College of Engineering, Mathematics and Physical Sciences

PhD in Computer Science

Interpreting multi-stable behaviour in input-driven recurrent neural networks

Submitted by **Andrea Ceni**, to the University of Exeter as a thesis for the degree of Doctor of Philosophy in Computer Science, May 2021.

Under the supervision of

Prof. **Lorenzo Livi**, University of Exeter, UK, and University of Manitoba, CA

Prof. **Peter Ashwin**, University of Exeter, UK

List of works

Part of the results presented in this thesis have been published in the following articles:

- Ceni, A., Ashwin, P., and Livi, L. (2020a). Interpreting recurrent neural networks behaviour via excitable network attractors. *Cognitive Computation*, 12(2):330–356.
  DOI: 10.1007/s12559-019-09634-2

- Ceni, A., Ashwin, P., Livi, L., and Postlethwaite, C. (2020b). The echo index and multistability in input-driven recurrent neural networks. *Physica D: Nonlinear Phenomena*, 412:132609.
  DOI: 10.1016/j.physd.2020.132609

Moreover, the candidate disseminated his original research through the following conferences and seminars:

1. 18-20 December 2018. Oral presentation at the international conference **Cognitive Computing - Merging Concepts with Hardware**, Herrenhaus, Hannover, Germany.

2. 29 January 2019. Oral presentation at the internal seminar of Applied Mathematics at **Department of Mathematics of University of Auckland**, New Zealand.

3. 19-23 May 2019. Oral presentation as part of a mini-symposium at the international conference at **SIAM Conference on Applications of Dynamical Systems (DS19)**, Snowbird, Utah, USA.

4. 21-29 September 2020. Oral presentation at the international conference at **Second Symposium on Machine Learning and Dynamical Systems**, Fields Institute, Toronto.

5. 11 March 2021. Oral presentation at the internal seminar of Applied Mathematics of the **University College Cork**, Ireland.

Abstract

Recurrent neural networks (RNNs) are computational models inspired by the brain. Although RNNs stand out as state-of-the-art machine learning models to solve challenging tasks as speech recognition, handwriting recognition, language translation, and others, they are plagued by the so-called vanishing/exploding gradient issue. This prevents us from training RNNs with the aim of learning long term dependencies in sequential data. Moreover, a problem of interpretability affects these models, known as the "black-box issue" of RNNs. We attempt to open the black box by developing a mechanistic interpretation of errors occurring during the computation. We do this from a dynamical system theory perspective, specifically building on the notion of Excitable Network Attractors. Our methodology is effective at least for those tasks where a number of attractors and a switching pattern between them must be learned. RNNs can be seen as massively large nonlinear dynamical systems driven by external inputs. When it comes to analytically investigate RNNs, often in the literature the input-driven property is neglected or dropped in favour of tight constraints on the input driving the dynamics, which do not match the reality of RNN applications. Trying to bridge this gap, we framed RNNs dynamics driven by generic input sequences in the context of nonautonomous dynamical system theory. This brought us to enquire deeply into a fundamental principle established for RNNs known as the echo state property (ESP). In particular, we argue that input-driven RNNs can be reliable computational models even without satisfying the classical ESP formulation. We prove a sort of input-driven fixed point theorem and exploit it to (i) demonstrate the existence and uniqueness of a global attracting solution for strongly (in amplitude) input-driven RNNs, (ii) deduce the existence of multiple responses for certain input signals which can be reliably exploited for computational purposes, and (iii) study the stability of attracting solutions w.r.t. input sequences. Finally, we highlight the active role of the input in determining qualitative changes in the RNN dynamics, e.g. the number of stable responses, in contrast to commonly known qualitative changes due to variations of model parameters.

Keywords: Recurrent neural networks, reservoir computing, black-box, echo state property, machine learning.

Contents

1 Introduction

Machine learning provides fundamental tools both for scientific research and for the development of technologies with significant impact on society. It provides methods that facilitate the discovery of regularities in data and that give predictions without explicit knowledge of the rules governing a system. However, a price is paid for exploiting such flexibility. Some of the most powerful machine learning methods, namely artificial neural networks, are typically black boxes where it is difficult to fully understand what the machine is doing or how it is operating. This poses constraints on the applicability and explainability of such methods. The difficulty of understanding the detailed processes in these models and of interpreting their decisions makes them unappealing, especially in fields as finance or medicine. Furthermore in scientific applications, it would be desirable to extract new knowledge from a trained neural network model. At present, deep neural network models are often exploited to deal with big data where humans are not able to find patterns. However, after the training session neural nets do not claim to have found a synthetic rule that characterises data as humans would do. If we are able to interpret the behaviour of a trained machine learning model then this transfer of knowledge from machine to human can happen. The black box issue of neural networks is certainly a broad problem that can be solved only with an effort from the whole research community. In this work we made an attempt in this direction focusing on recurrent neural networks, an important family of neural networks used for processing sequential data.

Artificial recurrent neural networks (RNNs) are widely used to solve tasks involving time-varying data, e.g. speech [39] and handwriting recognition [88], audio classification [57, 100] or time series forecasting [14]. RNNs are characterised by the presence of recurrent connections in a hidden layer, which allows generating a state–space representation that equips the network with short-term memory capability. RNNs are universal approximators of dynamical systems [33, 42], meaning that, given enough neurons in the hidden layer, it is possible to fine-tune the weights to achieve any desired level of accuracy. Nevertheless, training via back-propagation through time is difficult due to the vanishing/exploding gradient problem [54, 87]. This has led to the development of new and faster techniques for training RNNs, including a different paradigm known as reservoir computing [71, 72]. Echo state networks (ESNs) [52, 68] constitute an important example of reservoir computing, where a recurrent layer (called a reservoir) is composed of a large number of neurons with randomly initialised connections that are not fine-tuned via gradient-based optimisation mechanisms. The

main idea behind ESNs is to exploit the rich dynamics generated by the reservoir with an output layer, the read-out that is optimised to solve a specific task. Although their relatively simple architecture, compared to other stacked layered neural network models, ESNs emerge as powerful models both in machine learning and theoretical neuroscience. Therefore, we make use of ESNs as a prototypical example of RNNs for our theoretical investigations and numerical simulations.

We propose a novel methodology that provides a mechanistic interpretation of behaviour when solving a computational task. We do that from the point of view of dynamical systems theory; the state of an RNN is described by the set of real values of all the neurons composing the neural network. As such the dynamics of an RNN traces a trajectory in the high dimensional space which represents the space of all the possible states of the RNN, i.e. the phase space of the system. A subset of phase space in which a trajectory starts, and from which it never leaves, is called an invariant set. Sometimes invariant sets are regions of phase space where various trajectories (starting from other regions) converge to, and they are called attractors. Multi-stable systems are usually characterised by the coexistence of more than one attractor. The switching dynamics between attractors can be achieved by following particular trajectories that function as bridges between attractors, here called excitable connections. Our methodology uses mathematical constructs called excitable network attractors [4, 6, 20], which are invariant sets with a graph-like structure in phase space, where nodes correspond to stable attractors, and edges correspond to excitable connections between attractors. As the behaviour of recurrent neural networks depends both on training and on inputs to the system, we introduce an algorithm to extract network attractors directly from the trajectory of a neural network while solving tasks. Simulations conducted on a controlled benchmark task confirm the relevance of these attractors for interpreting the behaviour of recurrent neural networks, at least for tasks that involve learning a finite number of stable states and transitions between them.

Theoretical investigations on RNNs are usually accomplished under the assumption of a zero driving input or other tights constraints about the nature of the input signal. Those assumptions allow to consider an RNN as an autonomous dynamical system, and permit the application of standard tools from the theory of autonomous dynamical systems, such as fixed points, attractors, Lyapunov stability and bifurcation theory. Nevertheless, those are quite unrealistic hypothesis to assume, especially when dealing with challenging machine learning tasks. This fact motivated us

to go beyond the paradigm of autonomous dynamical systems theory and investigate RNNs from a nonautonomous dynamical systems perspective. Roughly speaking, nonautonomous systems are those where the equations ruling the dynamics change over time. Consequently, common notions as convergence, and stability need to be carefully defined. In particular, the so-called pullback attractor is the notion of attractor which shares more properties with the autonomous counterpart. Our investigation led us to question one of the fundamental principles commonly assumed to be necessary for reservoir computing, namely the Echo State Property (ESP). Informally the ESP embodies a request of reliability for a machine: the RNN internal state at present time must be uniquely determined by the whole history of the infinite past input sequence that we used to drive the RNN with. From the nonautonomus dynamical systems point of view the ESP is nothing but a request on the structure of the pullback attractor of the input-driven RNN, namely that the pullback attractor is formed by a unique trajectory in phase space. The ESP principle is formulated in terms of ranges of input values. This is problematic since there is a broad plethora of possible input sequences assuming values in a given set while having extremely different dynamics from each other. This in turn makes the ESP an unreasonably restrictive property to satisfy. A way to overcome this issue is to relate the ESP to a given input sequence or a family of them. We highlight some issues that arise in the process of shifting the ESP from set of values that input sequences can assume to set of possible input sequences. The ESP, for a compact set of input signals, ensures the existence and uniqueness of an input-driven response such that all initial conditions converge on it. Reliability of the system, in this sense, is meant as a guarantee of existence and uniqueness of an attractive response. On the other hand, when dealing with the ESP linked to a specific input sequence, any request on the pullback attractor cannot ensure the reliability of the system, as discussed in section 5.3.2. Therefore, in order to make the input-driven system reliable, we devise a suitable definition of nonautonomous attractor isolating the features needed to encode a "good" behaviour for the input-driven RNN. We show that under certain circumstances reliable, stable behaviour in input-driven RNNs is still possible without the classical ESP.

The thesis is structured as follows. Chapter 2 gives an overview on recurrent neural networks.

Chapter 3 provides some background material necessary to investigate RNN behaviour in the framework of dynamical systems theory. This chapter is a mixture of known facts and original

contributions, including published material from [20]. In particular, sections 3.1 and 3.2 are dedicated to fixed points, their linear stability analysis and bifurcation analysis w.r.t. low dimensional discrete-time RNNs, precisely of dimensions 1 and 2. Similar works can be found for instance in [10, 43, 111]. In section 3.3 we recall end extend the notion of network attractor [4, 6], and include original contributions (that have appeared in [20]) in section 3.3.1 in order to take into account input sequences and make numerical simulations feasible for large dynamical systems.

The rest of this thesis is mostly composed by original results developed by the candidate. In particular, chapter 4 outlines a methodology for characterising RNN dynamics by means of a weighted directed graph (or a sequence of graphs) encoding the behaviour of a RNN during the computation (or during training). Material from chapter 4 has been published in [20].

Chapter 5 describes results on RNNs driven by (deterministic) input sequences within the framework of the nonautonomous dynamical systems theory. Material from chapter 5 has been published in [21]. The ESP is investigated through the lens of nonautonomous attractors as well as the concept of uniform stability for fully input-driven RNNs. An input-driven fixed point theorem is proved and used for deducing very general results regarding fully input-driven RNNs, as for example the existence and uniqueness of a global attracting solution for strongly (in amplitude) input-driven RNNs or the existence of multiple responses for certain input signals.

Finally, Chapter 6 is dedicated to concluding remarks about the work done in this thesis, reflections on what has been achieved and some of the significant remaining open problems.

2 Recurrent neural networks

> The paradigm for intelligence was logical reasoning [...]. That has completely changed with these big neural nets.
>
> Geoffrey Hinton

Neural networks research is an extremely interdisciplinary field intersecting with several branches of knowledge as computer science, biology, psychology, engineering, philosophy and many others.

One of the more fascinating and challenging question of our time is to understand how the interaction of relatively simple units, i.e. the neurons, can lead to the emergence of cognitive functions such as memory, intuition, recognition, consciousness, and emotions. Most of the scientific knowledge achieved by humanity until nowadays has been based on a reductionist approach. Systems are divided in simpler sub-systems which obey to certain laws, then from the analysis of those sub-systems and their interactions we infer the behaviour of the whole system. On the other hand, complex systems theory is based on the holistic assumption that the whole is more than just the sum of all its components. This alternative approach provides a conceptual framework for the interpretation of those emergent properties that characterise complex systems that hardly can be inferred simply by investigating the basic units of the system.

Biological vs artificial NNs While in *theoretical neuroscience* (also called *computational neuroscience*) the focus is to understand the functioning of biological neural networks, for example how the brain encodes and process information, in *machine learning* the aim is to create and control artificial neural networks to solve more engineering-oriented tasks as speech recognition, computer vision, language translation etc. Artificial Neural Networks (NNs) are computational models inspired to a certain extent by biological brains. Intuitively, an NN consist of a collection of simple units (which are supposed to represent the neurons) nonlinearly interacting to each other by means of weighted connections (which are meant to represent the synapses).

Single neuron model According to the degree of adherence to the real complexity of the brain there exist various mathematical models of the neurons [35]. The Hodgkin–Huxley model describes

the functioning of a single neuron as a set of 4 nonlinear differential equations. It models the dynamics of the major ions channels involved in the determination of the electric potential of the cell membrane of a neuron and the mechanism of the generation of the action potential, i.e. the fundamental electrochemical spiking phenomenon responsible for the cell-to-cell communication. Simplifications of the Hodgkin–Huxley model lead to computationally less expensive models like the FitzHugh–Nagumo model or integrate-and-fire models [1, 22]. Reducing the whole state description of a neuron to a single real value (which can be interpreted as the membrane potential or the firing rate depending on the model) we get the simplest NN models. Analogously, rather than modelling the generation of an actual action potential, we might make simpler the interaction between neurons by modelling the output of a neuron as a nonlinear (sigmoidal) function of the sum of its inputs received by other neurons of the net, see the scheme in Figure 1.

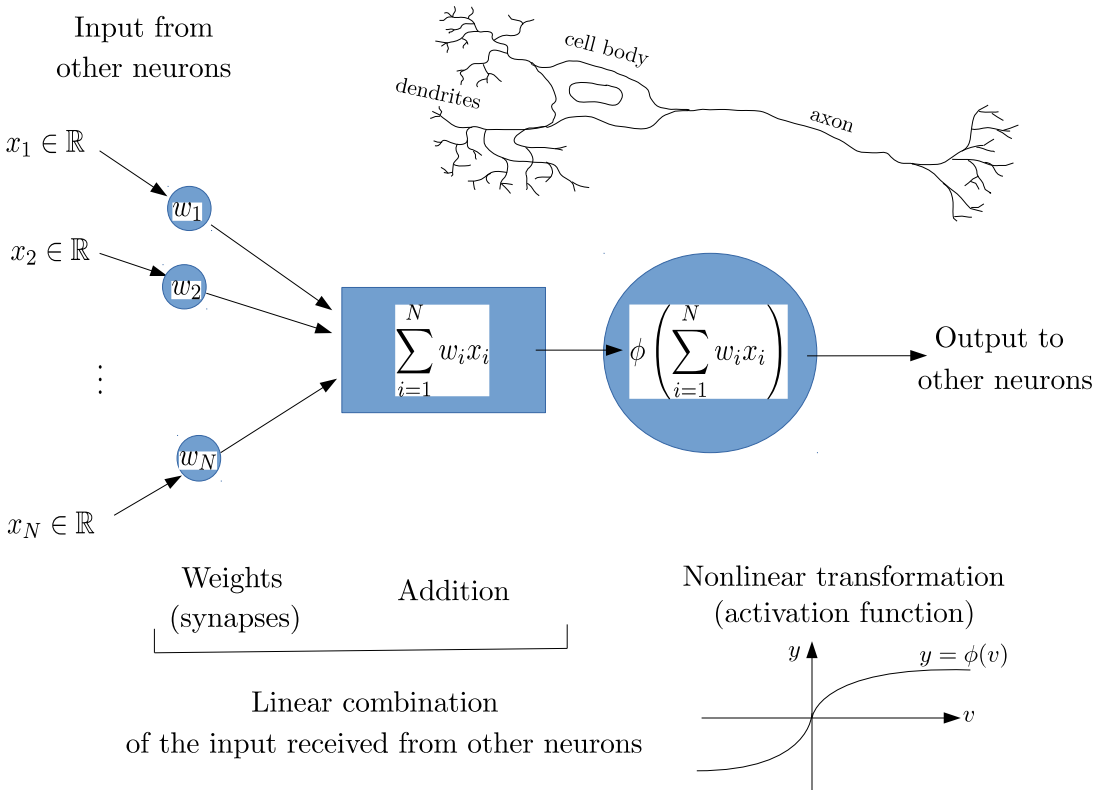


Figure 1: Sketch of one of the simplest and widely used single-neuron models: the McCulloch–Pitts (without bias).

Usually such a nonlinear function, called the *activation function*, is chosen with a sigmoid shape and to be a monotonically increasing, continuously differentiable and bounded function, e.g. the hyperbolic tangent [1]. The connections between neurons (i.e. synapses) are described as real values, called *weights*, representing the strength of those links; if the value is zero it means the connection does not exist, otherwise they can be inhibitory (negative values) or excitatory (positive values). Plasticity of the net is achieved by allowing those weights to adjust as learning proceeds, thus effectively increasing or decreasing the strength of the signal at a particular connection depending on the task to solve and the context. In this way, although neglecting the actual physiological dynamics of neurons and its generation of spikes, we may still gain some precious insides on the functioning of the network as a complex system.

2.1 Forward vs recurrent NNs

Among various distinctions between artificial neural network models, one of the most important is the distinction between *forward* NNs and *recurrent* NNs.

Forward NNs Often neurons are organised in layers in order to accomplish some particular aspects of the computation; hence the information flows from layer to layer until the output layer provide the final response to the input, see Figure 2.

For example, in a task of classification between pictures of cats and dogs the input layer might be a vector of real values encoding the picture, each real number between -1 (full white) and 1 (full black) may represent a specific shade of grey of a pixel of the picture. The output layer may be composed of 2 neurons respectively expressing the probability that a picture represents a cat or a dog, while the intermediate layers (also called *hidden layers*) might deal with the recognition of specific features as for example the raw shape of the animal or the texture of the hair etc.

The lack of loops in the architecture of these nets make them *forward* NNs. *Deep neural networks* have gathered massive interest in the last decade. These are usually forward NNs composed by a relatively large number of layers. Broadly, an NN might be considered deep if there are at least 2 hidden layers (plus one input, and one output).

[1] Also other choices have been demonstrated useful, as for example step functions and rectifiers [36].

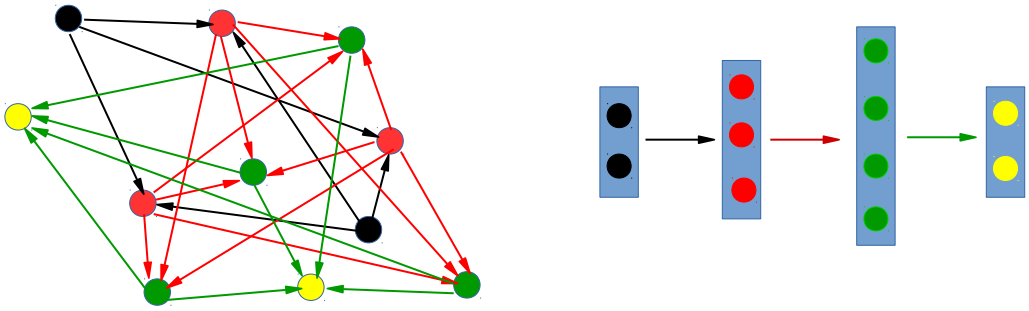


Figure 2: Two different representations of the same directed graph. There are no loops, black neurons compose the first layer, then the information is elaborated through other layers of neurons until yellow neurons produce the output.

Interestingly, in a similar sequential manner the information is processed in the visual cortex of the human brain [113]. Indeed, it has been observed that different regions of the brain (as in different layers of an artificial feedforward NN) process sequentially the visual stimulus coming from the optic nerve. At present, the state-of-the-art NNs for tasks as visual recognition are held by the so-called *convolutional neural networks* which exploit the deep forward architecture [64, 108]. A common interpretation in deep networks is that each layer is responsible and specialised for extracting a certain feature of the input [121]. Such example suggests that understanding artificial NNs can help interpreting neurobiological phenomena as well, besides the obvious other way round.

In forward NNs (or simply NNs) a vector u is provided as input. Such input is thus processed through a series of hidden layers in a sequential manner, see Figure 3, until the last layer, the output layer, provides the final response of the NN to the input u. In Figure 3 a NN with 3 hidden layers is sketched: firstly, the 3-dimensional input vector provided at time k, i.e. $u[k]$, is linearly transformed via the 4×3 matrix W_{in} and then processed component-wisely by means of the nonlinear sigmoidal function ϕ, hence obtaining $\phi(W_{in}u[k])$. Secondly, the resulting vector $\phi(W_{in}u[k])$ is used as input vector for the first hidden layer characterised by the 5×4 matrix $W_{1,2}$ and the nonlinear squashing action of ϕ, hence obtaining $\phi(W_{1,2}\phi(W_{in}u[k]))$. Then it passes through the second hidden layer getting $\phi\left(W_{2,3}\phi\left(W_{1,2}\phi\left((W_{in}u[k]\right)\right)\right)\right)$, where $W_{2,3}$ is a 3×5 matrix. Finally, the output layer reads

out the actual response of the NN by means of the mapping ψ. In this explanatory example, the NN results to be equivalent to the mapping $F : \mathbb{R}^3 \to \mathbb{R}^2$ defined as

$$F(u) := \psi\left(\phi\left(W_{2,3}\phi\left(W_{1,2}\phi\left((W_{in}u)\right)\right)\right)\right), \tag{1}$$

where the parameters of the model are the matrices $W_{in}, W_{1,2}, W_{2,3}$, and the function ψ.

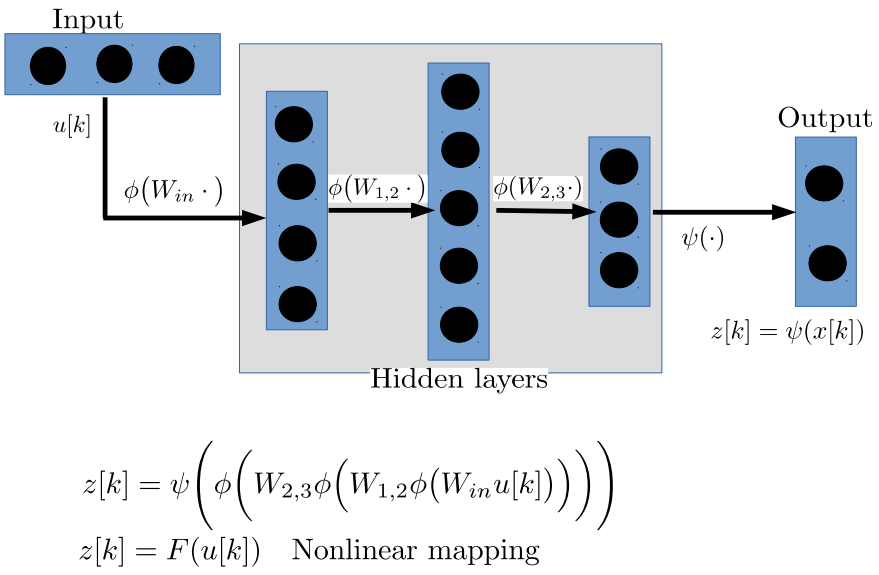


Figure 3: A sketch of a forward NN composed by 3 hidden layers with a total of 12 neurons and the output layer formed by 2 neurons. Although this forward NN is nothing but to a nonlinear function F mapping an input $u \in \mathbb{R}^3$ to an output $z \in \mathbb{R}^2$ there are 47 parameters to exploit (i.e. all the entries of the matrices $W_{in} \in \mathbb{R}^{4\times 3}$, $W_{1,2} \in \mathbb{R}^{5\times 4}$, $W_{2,3} \in \mathbb{R}^{3\times 5}$) plus any parameters of the function ψ.

From a strictly mathematical point of view, forward NNs are just nonlinear functions mapping from a domain, the input space, to a codomain, the output space. Nevertheless, the multiple composition of those simple actions of a linear combination plus a nonlinear transformation is enough to reproduce any possible real-valued continuous function on compact subsets, provided a sufficiently large number of neurons, that is parameters, whether they are distributed in width [27] or depth [58]: these constitute the statements of the *universal approximation theorems* for feedforward NNs.

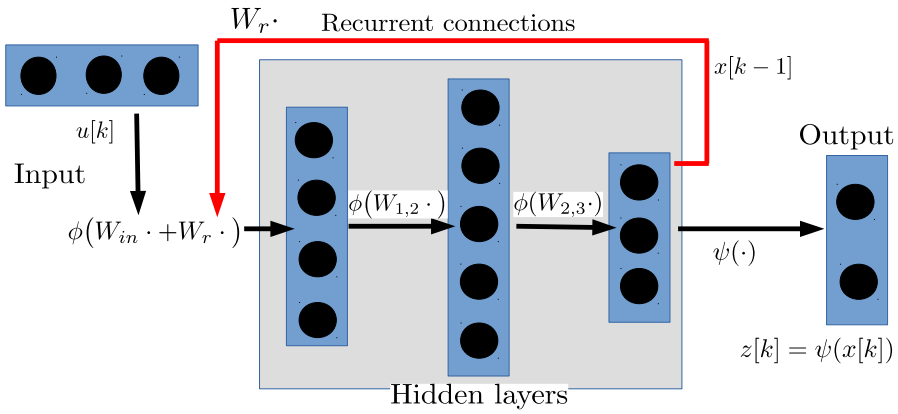
Remark 2.1 *Nonlinearity is the key to bring complexity in NNs. If the activation function ϕ is linear then the layered architecture is useless since it is equivalent to a single linear layer. For example, if $\phi(\cdot)$ is the linear transformation given by W_ϕ then* (1) *reads $F(u) := \psi(Hu)$, where $H := W_\phi W_{2,3} W_\phi W_{1,2} W_\phi W_{in}$.*

The last decade witnessed the raise of the exploitation of the so-called *deep neural networks*. Roughly speaking, the more deep the network is the more parameters available for training there are and the more potentially complex tasks the network is able to solve.

Recurrent NNs (RNNs) Forward NNs cannot be employed to deal with tasks involving time in them, e.g. wherever it is required working memory or decision making based on the context. In these tasks, the features of an input signal driving the dynamics should be somehow "memorised" and exploited at the present time in order to make a good choice. A forward NN cannot accomplish that. Once trained a forward NN cannot make decisions based on the context nor have a sort of memory of the past.

The key idea is to create a feedback in the system in order to promote a "recurrence" of the input signal in the net. The recurrence is not artificial at all if we consider that in the cortex of mammalian brains there are plenty of recurrent connections [8, 106]. Allowing the information of the output to flow back into the network, see Figure 4, the NN encodes an internal representation of the input history into its dynamics. The forward NN of Figure 3 is transformed into a recurrent NN introducing another matrix W_r modelling the recurrent connections, see Figure 4. In this way, at each time step k the NN produces an internal state $x[k]$ exploited for the computation of an output $z[k] = \psi(x[k])$. The computation of the current output is thus based on the current input value $u[k]$ and the previous internal state $x[k-1]$ which encodes the past history of the input signal $u[k-1], u[k-2], u[k-3], \ldots$.

From a purely mathematical perspective, the presence of a recurrent connection makes the NN a dynamical system; a considerably different object from a feedforward NN which is a function. Interestingly, RNNs might be interpreted as infinitely deep feedforward NNs by means of the trick of unfolding the recurrence in time, see Figure 5.



$$x[k] = \phi\Big(W_{2,3}\phi\Big(W_{1,2}\phi\big(W_{in}u[k] + W_r x[k-1]\big)\Big)\Big)$$

$$x[k] = G(x[k-1], u[k]) \quad \text{Nonautonomous dynamical system}$$

Figure 4: The same forward NN of Figure 3 is converted to a recurrent NN introducing a feedback loop connecting the resulting vector of the last hidden layer at time $k-1$ to the first hidden layer at time k by means of the matrix W_r. In this case we might consider the last hidden layer $x[k-1] \in \mathbb{R}^3$ as the internal state of the recurrent neural network which evolves according to the external input sequence $u[k]$. This makes the system a nonautonomous dynamical one described by a time-varying map $x[k] = G(u[k], x[k-1])$. Note that the current state $x[k]$ is somehow a function of all the past history of the input signal since $x[k-1] = G(u[k-1], x[k-2])$ and $x[k-2] = G(u[k-2], x[k-3])$ and so on and so forth until the initial condition $x[1] = G(u[1], x[0])$.

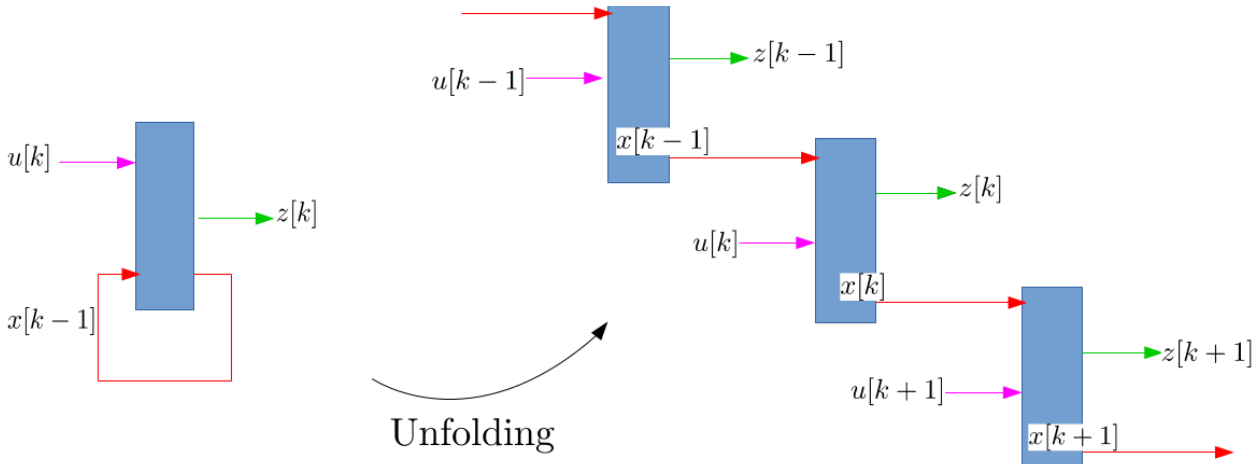


Figure 5: A blue rectangle represents a RNN machine. It produces an output $z[k]$ (green arrow) at each time step k according to the input $u[k]$ (magenta arrow) and the previous value of its internal state $x[k-1]$ (red recurrent arrow). A RNN can be interpreted as an infinitely deep feedforward NN which consecutively generates an output sequence driven by an input sequence. Each layer of this unfolded feedforward NN corresponds to the hidden state of the RNN at a given time step k, and it is linked with the next one via the recurrent connection (red arrow).

2.2 The state-space model

Despite the differences of goals and methods, both computational neuroscience and machine learning benefit of the discovers of each other field and share a common ground: the mathematical model of a recurrent neural network. A popular continuous-time model of a recurrent neural network is the following system [9], for $i = 1, \dots, N_r$, $\tau_i \dot{v}_i(t) = -v_i(t) + \sum_{j=1}^{N_r} w_{ij}\phi(v_i(t)) + J_i(t)$, of N_r first-order nonlinear differential equations coupled by means of the coefficients w_{ij}. Here, $v_i(t)$ is the membrane potential of the ith neuron, and $J_i(t)$ represents an external (w.r.t. the network) input stimulating the ith neuron. Such model is also known as *Additive Neural Network* [44]. This model is massively used in theoretical neuroscience, it can be derived through the Kirchoff's current law treating a neuron as an electrical RC circuit where the synaptic weights w_{ij} are interpreted as *conductances*, [35, 116], therefore linking to the actual biological functioning of a real neuron. Assuming all the neurons of the network to have the same RC time constant τ and forced by the same external input $J(t)$ we can simplify the additive model in matrix notation as follows

$$\tau \dot{v}(t) = -v(t) + W_r \phi(v(t)) + J(t), \tag{2}$$

where the entries of the recurrent matrix W_r are the coupling coefficients w_{ij}. The astonishingly high number of neurons in even a relatively small portion of the cerebral cortex makes the assumption of randomness of W_r interesting to study. Therefore, in the literature often the entries of W_r are assumed random i.i.d. variables according to a Gaussian or Uniform distribution, and in such a case the model is also called Random Neural Network, especially regarding networks of spiking neurons. Mean-field techniques [22, 76, 94, 96, 104] borrowed from Physics have been widely and successfully applied to understand some aspects of Random Recurrent Neural Networks in the thermodynamic limit of infinite neurons, i.e. $N_r \to \infty$.

Model (2) has been proved [33] to be able to approximate any nonlinear dynamical system to any desired degree of accuracy, provided that the network is equipped with an adequate number of hidden neurons and an appropriate W_r. The above statement is known in the NN community as the *universal approximation theorem for RNNs*, and it is testimony to the computing power of recurrent neural networks for machine learning applications. Morever, RNNs are capable to compute whatever is possible to compute with a Turing machine [102].

A mathematically equivalent formulation of model (2) is the following one in terms of $x \in \mathbb{R}^{N_r}$, the vector of firing rates of each neuron,

$$\tau \dot{x}(t) = -x(t) + \phi(W_r x(t) + \tilde{J}(t)), \tag{3}$$

with $\tilde{J}(t)$ representing an external input. A proof of this equivalence can be found in [80]. In model (3) we will assume $\tilde{J}(t)$ to have the following form

$$\tilde{J}(t) := W_{in} u(t), \tag{4}$$

so that we can rescale the external input driving the network, denoted as $u(t)$, by means of the matrix W_{in}. The discrete-time version of (3)-(4) reads

$$x[k+1] = (1-\alpha)x[k] + \alpha\phi(W_r x[k] + W_{in} u[k+1]), \tag{5}$$

where the scalar parameter $\alpha \in (0, 1]$ can be interpreted as $\dfrac{1}{\tau}$ (physically a frequency, representing the time scale of each neuron of the network) multiplied by the discrete time step of integration of (3) by means of the Euler method [53]. The discrete-time version (5) stands as a model from its own, and it is preferred in machine learning and engineering-oriented applications. After all, when

it comes to run a simulation of the continuous-time RNN model (3)-(4) we need to implement a discrete-time version of it in the calculator. In this thesis, we will deal with discrete-time RNNs.

We can view a discrete-time RNN, as the one in (5), as a mapping which takes as input the time series $u[k]$ and develops an internal state time series $x[k]$ according to the initial condition $x[0]$ and the time-varying rule:

$$x[k+1] = G(u[k+1], x[k]). \tag{6}$$

In response to the input signal $u[k]$ an output signal $z[k]$ is produced by the RNN which somehow encodes the solution to the task at hand

$$z[k+1] = \psi(x[k+1]). \tag{7}$$

The particular implementation of $\psi(\cdot)$ in (7) depends on the task at hand. For instance, in classification tasks $\psi(\cdot)$ might take the form of a softmax assigning probabilities to predicted classes; in forecasting the easiest choice is a linear deterministic function, i.e. $z[k+1] = \psi(x[k+1]) = W_o x[k+1]$, although nonlinear functions are also common. Moreover, depending on the specific task to solve it might be useful to feed back the output signal into the network, leading to the following discrete-time RNN model with leaky-integrator neurons,

$$x[k+1] = (1-\alpha)x[k] + \alpha\phi(W_{in}u[k+1] + W_r x[k] + W_{fb}z[k]), \tag{8}$$

$$z[k+1] = \psi(x[k+1]). \tag{9}$$

The model (8)-(9) is the one we will refer throughout this manuscript. It is a fairly general RNN model [13], specifically a state-space model [44] describing the evolution of state variable $x[k] \in X \subset \mathbb{R}^{N_r}$ in discrete time $k \in \mathbb{Z}$. $\phi(\cdot)$ is a component-wise activation function (e.g. hyperbolic tangent) and $u[k] \in U$ is the N_i-dimensional input sequence for $U \subset \mathbb{R}^{N_i}$. The sets X and U denote the state (or phase) and input spaces, respectively. The matrices $W_r \in \mathbb{R}^{N_r \times N_r}$ and $W_{in} \in \mathbb{R}^{N_r \times N_i}$ represent recurrent and input-to-network couplings. The output feedback matrix $W_{fb} \in \mathbb{R}^{N_r \times N_o}$ injects the last computed output into the state-update equation (8). The scalar $\alpha \in (0, 1]$ can be used as hyperparameter to control the RNN time-scale given by $1/\alpha$ [53, 110], and it is usually referred to as the *leaking rate*.

Referring to stacked layered RNNs, as the one of Figure 4, model (8)-(9) describes a RNN with just one hidden layer, fully recurrent and usually large.

2.3 Training

The aim of this section is to provide an informal description of what is meant by the term "training" in machine learning.

Although the universal approximation theorem proves an RNN is potentially able to reproduce any dynamical system, it is not known a procedure to constructively set the RNN's parameters allowing a given RNN to approximate a desired dynamical system. Such a procedure is the essence of what is called *training* of a RNN. The key idea is to define a *loss function* depending on the task to solve and vary the parameters of the model in order to achieve a sufficiently low minimum of the loss function. Roughly speaking, two main categories of learning algorithms exist: *supervised* and *unsupervised*. The former it is applicable whenever we possess a dataset labelled as input-output samples to exploit for training the model. The latter when our dataset is unlabelled. In this work we focus on supervised learning.

For example, pattern generation is a typical task where supervised learning is usually implemented. Assume to own the target signal $\{y[k]\}_k$ that we want the RNN to reproduce, also called *teacher signal*. A simple loss function might be defined as the sum of all the errors at each time step between the target and the actual output signal produced by the RNN, i.e. $\mathcal{E} = \sum_k \|y[k] - z[k]\|^2$, where $z[k]$ is the actual output of the RNN. Clearly, the output $z[k]$ is a function of all the parameters of the model (8)-(9), i.e. all the entries of W_{in}, W_r, W_{fb}, W_o, and that makes the loss function $\mathcal{E} = \mathcal{E}(W_{in}, W_r, W_{fb}, W_o)$ a function of all the parameters of the model. *In these terms, any algorithm which modifies the entries of those matrices with the purpose of minimising the loss function can be defined as a learning algorithm.* If the training session is successful then the RNN is set in a parameter configuration which produces a sufficiently accurate output signal.

In general, training is accomplished through an optimisation algorithm exploring the landscape of the space of the model's parameters. Gradient descent algorithms are by far the most employed optimisation algorithms; they are all variations or generalisations based on the *backpropagation algorithm*, a popular method proposed in the late eighties for training feedforward NNs [99]. Such a method can be applied to train also RNNs, and it takes the name of *backpropagation through*

time (BPTT), picturing to unfold trough time a RNN as an infinite feedforward NN, see Figure 5.

Unfortunately, learning long-term dependencies with gradient descent (e.g. with BPTT algorithms) is known to be problematic, as a consequence of the so-called vanishing/exploding gradient problem [87]. For this purpose, two types of methods have been proposed: (i) gating mechanisms [24, 45] and (ii) approaches based on unitary matrices and constant-slope activation functions [119]. Alternatively, in the early 2000s a new paradigm of computing with neural networks has been proposed [51, 73]: the *reservoir computing*.

2.3.1 Reservoir computing and Echo State Networks

Reservoir computers (RCs), a special class of RNNs, bypass the problem associated to BPTT as training targets the output layer weights only. Two similar models of RC have been independently proposed: the Echo State Network (ESN) implementing analog neuronal activation functions [51], and the Liquid State Machine making use of spiking neurons [73]. Both consider a large pool of neurons as a unique hidden layer characterised by randomly generated and sparse connections expressed by the recurrent matrix W_r of Equation (8), which is commonly dubbed the *reservoir*. The wide variety of response of the neurons embodies a high-dimensional representation of the input signal injected into the RC, and the output layer is trained in order to produce the desired target signal.

In the context of reservoir computing, all the reservoir $W_r \in \mathbb{R}^{N_r \times N_r}$, input $W_{in} \in \mathbb{R}^{N_r \times N_i}$, read-out $W_o \in \mathbb{R}^{N_r \times N_o}$, and in case $W_{fb} \in \mathbb{R}^{N_r \times N_o}$ matrices of Equation (8) are usually random with i.i.d. entries drawn from uniform or Gaussian distributions [70, 71]. However, in the literature it is possible to find reservoirs with different connection patterns, including deterministic topologies [98] and those exploiting the norm-preserving property of orthogonal matrices [79].

Although, there exist techniques as FORCE learning [106] where the properly recurrent connections, i.e. the entries of W_r, are modified, usually in reservoir computing training is achieved by modifying only the read-out matrix W_o. Since all the other matrices are essentially left untrained, it is convenient to rescale them at a pre-training level by means of three hyper-parameters as $\rho W_r, wW_{in}, \nu W_{fb}$, which are heuristically tuned according to the specific task to solve. The particular choice of this triplet directly affects the performance of a reservoir computer. In the literature, extensive analyses have been made on linking the hyperparameter ρ which directly tunes

the spectral radius of the reservoir with the so-called echo-state property (ESP) [34, 51, 74, 120]. The ESP guarantees the existence and uniqueness of a global attracting trajectory for any input sequence in a compact set. Such a property was conceived in order to make an ESN a reliable system producing a stable input-driven response, hence in rough terms to set the ESN to work well. All the three hyper-parameters ρ, w, ν, are directly related with the ESP. We will investigate deeply on the ESP in Chapter 5 from a nonautonomous dynamical systems theory perspective.

Some other relevant hyper-parameters directly affecting ESN performance include the leak rate α, the number of neurons N_r and sparseness[2] of their connections [12, 67].

2.3.2 Training ESNs with low-rank perturbation matrices

Generally, in reservoir computing the recurrent layer is randomly instantiated and the model is modified only offline at the hyper-parameter level. Nevertheless, this simple training protocol is not sufficient in many applications, e.g. when it is required to learn memory states. To this end, training mechanisms based on output feedback [78, 97] and online training [46, 106] have been proposed, with successful applications in physics [69, 101], complex systems modeling [17, 48], and neuroscience [15], just to name a few. In particular, feeding back the trained output to the recurrent layer effectively corresponds to perturb the randomly-initialised reservoir with a deterministic matrix with a specific structure learned during the training session. Often such a structured deterministic matrix turns out to have low rank, from which the name low-rank perturbation [62, 66, 95, 97, 115]. For example, in [77] they proposed to directly design the reservoir as $W_r = A + D$, where A is a random matrix and D is a deterministic, low-rank matrix encoding the batch of edits to be made in A in order to prepare the reservoir to solve the task of interest.

In Chapters 4 and 5, we use a supervised learning algorithm which exploits the feedback of the ESN output as a mechanism for training the recurrent layer. Specifically, we use the model of Equation (8) with no leakage, i.e. $\alpha = 1$, and with an additional term, namely ϵ, which represents an additive white Gaussian noise with spherical covariance matrix and unit standard deviation, as

[2] In biological large neuronal microcircuits a neuron is more unlikely to be connected to neurons located very far from its neighbourhood in the neural network [65], hence there are usually many neurons which are not connected to each other. A simple way to account this phenomenon is to consider a sparse recurrent matrix W_r.

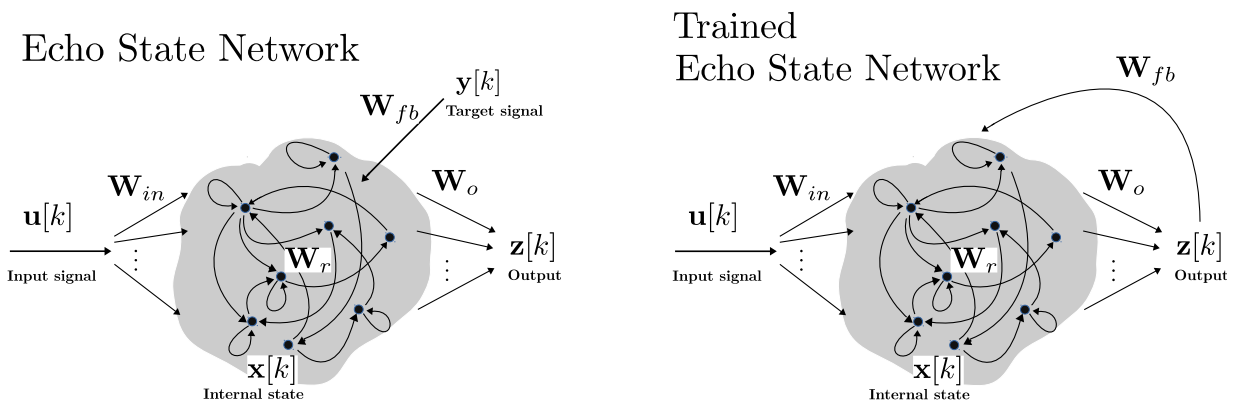


Figure 6: **Left:** Illustration of a ESN in the open-loop (training) phase (10)-(11). Training is supervised by means of the target signal y. **Right:** Illustration of an ESN (13) after the optimisation of the read-out matrix.

follows.

$$x[k] = \phi(W_r x[k-1] + W_{in} u[k] + W_{fb} y[k-1] + \epsilon), \tag{10}$$

$$z[k] = W_o x[k]. \tag{11}$$

The presence of noise in the training session is proved to be useful to the purpose of learning without overfitting [70].

Depending on whether training is performed batch or online, $y[k-1]$ in (10) takes the form of either the target signal or the output produced by the ESN (11), respectively. In batch mode, it is possible to distinguish two main phases; see Figure 6 for an illustration. First, the reservoir is fed with an auxiliary input, i.e. the target signal y. Therefore, we construct a matrix $X \in \mathbb{R}^{N \times N_r}$ containing the states $x[k]$ of the ESN generated in response to target and input signals. Finally, the weights W_o of the read-out are determined by solving a regularised least-squares problem, $W_o = \left(\left(X^\top X + \lambda^2 I\right)^{-1} X^\top y\right)^\top$, where I is an $N_r \times N_r$ identity matrix and $\lambda \geq 0$ is a regularisation parameter. Successively, the target signal is replaced by the generated output z; this "closed-loop phase" corresponds to the test phase of the trained ESN. An analysis of the stability of the transition from open- to closed-loop can be found in [97]. Reservoir computers are classically trained without feeding back the output, thereby the untouched recurrent part acts as a filter for the input signal, and the trained output layer is basically decoupled from such filter. Injecting the output back to

the reservoir could cause instabilities, but it might enable to represent dynamics beyond fading memory filters. Although interesting, these issues are not investigated in this thesis.

Definition 2.1 *We call the* trained reservoir *the following matrix*

$$M := W_r + W_{fb}W_o. \tag{12}$$

Once the read-out matrix W_o is optimised, by imposing $y[k] = z[k]$ and expanding in (10) with (11), we obtain:

$$\begin{aligned} x[k] =& \phi(W_r x[k-1] + W_{in}u[k] + W_{fb}W_o x[k-1] + \epsilon) = \\ =& \phi(Mx[k-1] + W_{in}u[k] + \epsilon). \end{aligned} \tag{13}$$

Remark 2.2 *The trained reservoir* (12) *is obtained by adding a matrix* $W_{fb}W_o \in \mathbb{R}^{N_r \times N_r}$*, which is low-rank* $N_o \ll N_r$ *relative to the randomly initialised reservoir matrix* W_r*. Therefore the reservoir is in some sense trained using output feedback connections.*

Note that the training procedure described here via equations (10)-(13) is applicable also for any value of the leaking rate $\alpha \in (0, 1]$. We reported the equations with $\alpha = 1$ just for clarity of exposition.

The ESN read-out matrix W_o is conventionally determined by solving a regularised least-squares problem. Nonetheless, we note that also online training schemes have been developed, e.g., the FORCE learning algorithm originally introduced in [106] and further extended in [28]. During the test phase, regardless of the adopted training mechanism, the state-update of ESNs is described by (13).

In this work, we consider batch training via ridge regression and analyse the trajectory generated during the test phase. It is worth nothing that our theoretical framework does not rely on a particular training method or a particular RNN architecture. We note that complicated (trained) RNN models may be described using a noisy nonautonomous dynamical system, which in our system is represented by (13). We focus on ESNs as they are the simplest RNN models to test our hypothesis. For this reason, the terms ESN and RNN are used interchangeably throughout the thesis.

2.4 Black-box issue

Generally, three main features prevent us obtaining a comprehensive understanding of the behaviour of RNN dynamics:

- nonlinearity;
- high dimensionality;
- strong dependence on the input.

Those three features coexist and influence each other. For example, the input signal injected into the network affects the nonlinearity of the system driving the activation functions of some neurons towards saturation. Furthermore, the amount of single neurons acting in a regime of strong nonlinearity promotes an effective reduction of dimensionality of the portion of phase space where the dynamics take place [10]. On the contrary, the input might drive some neurons to the linear case of small values of their activation functions leading to an actual enlargement of the dimensions exploited in the phase space.

As a result of the training session we end up with a massively large input-driven nonlinear dynamical systems with all its (millions or billions) of parameters finely tuned in order to accomplish a given task. Roughly speaking, we know how to make the RNN work but we do not know how it actually works, hence the name "black-box". Why the RNN made a particular decision instead of another? These kinds of questions not only prevent us from understanding, hence controlling and expecting, RNNs behaviour, but also preclude us from extracting new knowledge from trained RNNs [18]. A solution would be to provide a mechanistic interpretable model of the functioning of a trained RNN while it is solving the task. It would be possible to correlate some properties of the attractors and the excitable connections between them with certain good or bad behaviour of the trained model, thereby highlighting which are the hyperparameters that have more influence for the success of the training of a given task. For example, if the attractors sit too close to their basin boundaries then training with larger noise or enforcing regularisation can help to maintain the attractors sufficiently far from their basin boundaries. Moreover, possessing a mechanistic model of the functioning of a RNN might provide insights on new effective methods for training RNNs. It is an exciting, yet very hard problem: it cannot be solved in a few years of research and requires

focused research efforts from the entire community. In this manuscript we attempt to shed light on the shadow of the black-box issue of RNNs.

3 Nonlinear dynamics of RNNs

> The identification of attractors with computational objects (e.g., associative memories and input–output mappers) is one of the foundations of neural network paradigms.
>
> Simon Haykin

In this chapter we introduce some concepts of dynamical systems theory, indispensable to define and understand the key notion of excitable network attractor on which we build the next chapter.

The core idea of investigating RNNs as dynamical systems is to interpret the behaviour of RNNs as the evolution of their trajectories in the high-dimensional phase space of all the possible neuronal activation $x \in \mathbb{R}^{N_r}$. Given an initial condition of the network, the state-update equation (8) generates a trajectory which draws an orbit in phase space as time flows. Some orbits are time-invariant, that means regardless of the initial time the trajectories draw the same figure in phase space. Examples of time-invariant sets are fixed points, which model stationary states for the RNN, closed orbits, which represent periodic patterns of the activation functions, and multidimensional tori, which encode quasi-periodic dynamics. Furthermore, some trajectories may trace sets characterised by complex and fractal geometries, called *strange attractors*, a proxy of chaotic dynamics. Apart from different topologies, invariant sets of the dynamics can be stable or unstable. Trajectories perturbed from the location of the former remain in a certain neighbourhood, while the latter lead closer trajectories to deviate significantly from them. As a consequence, often many trajectories tend to converge towards particular regions of phase space regardless of their initial conditions, rendering significant to study the properties of these attracting regions. As a matter of fact, many common dynamical systems encountered in nature are dissipative [19, 105]. In such systems, the absence of any conservation law means that typical trajectories evolve towards an attracting set of dimension strictly less than the original phase space dimension; such a set (or a particular subset of it) is commonly called an *attractor*: formal definitions are discussed for example in [82]. It has been established that attractors are of relevance for the behaviour of RNNs, and modifications of attractors as parameters' model vary through the training session reflect *bifurcations* of the dynamical system. Informally speaking, bifurcations are qualitative changes of

the dynamics of the system; for instance, the appearance/disappearance of an attractor in phase space can happen in various ways that are studied in bifurcation theory [63]. Bifurcations might lead a system characterised by a unique attractor to the coexistence of more than one attractor, i.e. giving rise to multi-stable systems, see [89] for an extensive review on multistability. Multistability has been proposed to underlie functions like working memory and decision-making [30]. The *basin of attraction* of an attractor is the set of all initial conditions from which the system evolves toward the attractor. The shape of a basin of attraction and its connections with the basins of other attractors convey crucial information about the nonlinear dynamics of the system. From this perspective, a learning algorithm shapes the attractors, their basins, and their locations with the aim of encoding and retrieving information suitably for a desired task to solve.

The remainder of this chapter is structured as follows. Section 3.1 reviews notions of fixed points and their linear stability. Section 3.2 discusses bifurcations of fixed points for low-dimensional RNN maps. In Section 3.3 we introduce the concept of *network attractor* on which we will build on Chapter 4.

Notation: throughout this thesis we denote a fixed point with a superscript star *. Often, in chapters 3 and 4 we also indicate fixed points via bold letters, while on chapter 5 we dedicate bold letters for infinite sequences (of inputs or states) and leave only the superscription * to denote fixed points.

3.1 Fixed points

A fixed point $\mathbf{p} \in \mathbb{R}^{N_r}$ of an autonomous dynamical system

$$x[k] = F(x[k-1]) \tag{14}$$

is a solution of $F(\mathbf{p}) = \mathbf{p}$. Fixed points represent the simplest type of invariant sets for an autonomous dynamical system. Their individuation and analysis of stability gives precious insights on the overall dynamics. Generally they can be (Lyapunov) stable or unstable. In the first case a little perturbation $\delta x = x - \mathbf{p}$ from the position of the fixed point will bring the internal state of the system to evolve inside a ball $B_r(\mathbf{p})$ centred on the fixed point of a radius r, where $r = r(\|\delta x\|)$ is a function of the perturbation such that $r(0) = 0$ which is continuous in 0. In the second case there is no such a continuous function $r = r(\|\delta x\|)$ and so any perturbation, even the smallest infinitesimal, will lead the internal state of the system to move away from the fixed point. Often, a perturbation

gets damped as time evolve and the state of the system converges back to the fixed point; in such a case the fixed point is called an *asymptotically stable fixed point*. An asymptotically stable fixed point is the simplest kind of attractor.

Related to the notion of attractor is the notion of *limit set* [19], thus we introduce the following

Definition 3.1 *The* ω-limit set *of a point* x_0 *under the iterated map* (14) *is defined by*

$$\omega_F(x_0) := \bigcap_{n\in\mathbb{N}} \overline{\{F^h(x_0) \,|\quad h > n\}}. \tag{15}$$

The ω-limit set of a point x_0 is the set of limit points of the forward trajectory $\{F^h(x_0)\}_{h\in\mathbb{N}}$. By means of the ω-limit set we can define the *stable and unstable sets* of a fixed point as follows.

Definition 3.2 *The* stable set $W^s(\mathbf{p})$ *of a fixed point* $\mathbf{p}$ *under the iterated map* (14) *is defined by*

$$W^s(\mathbf{p}) = \big\{x \in \mathbb{R}^{N_r} \mid \omega_F(x) = \{\mathbf{p}\}\big\}. \tag{16}$$

The unstable set $W^u(\mathbf{p})$ *of a fixed point* $\mathbf{p}$ *is the stable set w.r.t. the inverse map of* (14).

Remark 3.1 *For a given fixed point* $\mathbf{p}$*, the stable set is the set of initial conditions whose long-time behaviour leads to the fixed point* $\mathbf{p}$*. Such set in general might be not empty and simultaneously have a zero Lesbegue measure, e.g. whenever the fixed point is a saddle. Whenever the fixed point is an asymptotically stable one then there exists a neighbourhood of* $\mathbf{p}$ *whose* ω*-limit set corresponds to* $\mathbf{p}$*. Therefore, in such a case the stable set* $W^s(\mathbf{p})$ *has interior and we can basically refer to the stable set as the* basin of attraction *of the attractor* $\mathbf{p}$*. If* $\mathbf{p}$ *is hyperbolic then the stable set* (16)*, i.e. the basin of attraction of* $\mathbf{p}$*, is a manifold and it is called the stable manifold of* $\mathbf{p}$*.*

The first step to analyse the stability of a fixed point is to compute the eigenstructure of $J_F(\mathbf{p})$, i.e. the Jacobian of the map F evaluated on the fixed point $\mathbf{p}$. It is known [105] that if a fixed point is hyperbolic (i.e., $J_F(\mathbf{p})$ has no eigenvalues on the unit circle in the complex plane), then the linear approximation provides a bona-fide characterisation of the non-linear behaviour around that fixed point. In fact, for an hyperbolic fixed point the map F is locally invertible and it turns out that the stable set and unstable set are regular manifolds and their tangent spaces are represented by the linear stable spaces of its Jacobian on the fixed point. Therefore, the eigenspaces associated to eigenvalues with a modulus greater than 1 indicate those directions in phase space where the

perturbation δx gets amplified. A fixed point with at least one eigenvalue with module greater than one is unstable. A fixed point with the whole spectrum of eigenvalues inside of the unit circle is is an asymptotically stable point, thus an attractor, on the contrary a fixed point with all the eigenvalues outside the unit circle is called a *repeller*. A fixed point with some eigenvalues outside and others inside the unit circle is called a *saddle*. Saddle points have a key role in the global dynamics of a system [107]; they attract initial conditions from their stable manifolds and funnel them towards their unstable manifolds which are locally approximated by the eigenvectors characterised by eigenvalues with module greater than 1.

3.1.1 Linear stability analysis of RNNs

Let us consider the following noise-free discrete-time dynamical system with inputs:

$$x[k] = G(u[k], x[k-1]), \tag{17}$$

where $G : \mathbb{R}^{N_i} \times \mathbb{R}^{N_r} \longrightarrow \mathbb{R}^{N_r}$ is related to (8) as follows:

$$G(u,x) = \begin{pmatrix} (1-\alpha)x_1 + \alpha\phi\big(\xi_1(u,x)\big) \\ (1-\alpha)x_2 + \alpha\phi\big(\xi_2(u,x)\big) \\ \vdots \\ (1-\alpha)x_{N_r} + \alpha\phi\big(\xi_{N_r}(u,x)\big) \end{pmatrix}, \tag{18}$$

where $\xi_i(u,x) := (W_r)_{(i)} \cdot x + (W_{fb})_{(i)} \cdot \psi(x) + (W_{in})_{(i)} \cdot u$ is the i-th neuronal pre-activation, and the subscript (i) denotes the ith rows of a matrix and $\cdot$ the usual dot product.

The first-order derivative w.r.t. the state space variable of ith component of the RNN state of (18) reads

$$\frac{\partial G_i}{\partial x_j}(u_0, x_0) = (1-\alpha)\delta_{ij} + \alpha\phi'\big(\xi_i(u_0,x_0)\big)(M(x_0))_{ij},$$

where δ_{ij} is the Kronecker delta, and

$$M(x) = W_r + W_{fb} D_x \psi(x), \tag{19}$$

denotes the "effective recurrent matrix", where $D_x\psi(\cdot)$ is the first-order derivative of $\psi(\cdot)$. Therefore, it is possible to write in compact form the first-order derivative matrix of G w.r.t. the state variable evaluated onto (u_0, x_0) as

$$D_x G(u_0, x_0) = (1-\alpha) I_{N_r} + \alpha S(u, x_0) M(x_0), \tag{20}$$

where

$$S(u_0, x_0) = \begin{bmatrix} \phi'\big(\xi_1(u_0, x_0)\big) & 0 & \dots & 0 \\ 0 & \phi'\big(\xi_2(u_0, x_0)\big) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \phi'\big(\xi_{N_r}(u_0, x_0)\big) \end{bmatrix}. \tag{21}$$

The presence of the potentially varying input at each time step complicates significantly the analysis. Moreover, bear in mind that the concept of a fixed point does not make any sense with a time-varying input. In fact, there cannot exist a fixed point for a dynamical system driven by nonconstant input signal. We will deal with proper input-driven RNNs in Chapter 5. For the moment, we can substantially simplify the problem by considering the case where the neural network is autonomous, i.e. the case where the input is constant. Note that, there are some tasks where the input signal is almost everywhere constant, or slightly varying around a constant value, or piecewise constant. In those cases, the determination of fixed points of the underlying autonomous systems might be fundamental for understanding the input-driven dynamics. Specifically, let's simplify the analysis assuming that the constant input is zero, that is we deal with the input-free map

$$x[k] = F(x[k-1]), \qquad \text{where } F(x) = G(0, x). \tag{22}$$

Now, assuming a linear readout $\psi(x) = W_o x$, and the activation function $\phi = \tanh$, then the Jacobian matrix evaluated onto x_0 reads $J_F(x_0) = (1-\alpha)I_{N_r} + \alpha D(x_0)M$, where $M = W_r + W_{fb}W_o$ is the trained reservoir as in (12), and

$$D(x_0) = \begin{bmatrix} 1 - \tanh^2(M_{(1)} \cdot x_0) & 0 & \dots & 0 \\ 0 & 1 - \tanh^2(M_{(2)} \cdot x_0) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 - \tanh^2(M_{(N_r)} \cdot x_0) \end{bmatrix} \tag{23}$$

is an $N_r \times N_r$ diagonal matrix representing the squashing action of $\phi(\cdot)$ along the saturating components of the internal state x_0.

By linearizing the network state-update $x[k] = F(x[k-1])$ around a given fixed point x^*, we obtain the linear system $\delta x[k+1] = J_F(x^*)\delta x[k]$, where $\delta x[k] = x[k] - x^*$. Therefore, the (linear) stability of an hyperbolic fixed point x^* is completely determined by the spectral radius of $J_F(x^*)$. If

it is strictly less than one, then x^* is an attractor; on the other hand, if even just one eigenvalue has norm larger than one, then the linearized map is expanding along the corresponding eigenvectors and x^* is unstable.

Attractors may appear, disappear or change their properties according to parameter variations or external input variations. The *saddle-node* bifurcation (or fold bifurcation) is probably the commonest phenomenon to generate new fixed points or annihilate them. A slow point arises in a certain region of phase space, then such a point undergoes a bifurcation and splits in a couple of fixed points: one stable and the other unstable. A *slow point* [107] is a point in phase space where the dynamics tend to significantly slow down but without being a fixed point. This slow-down dynamics is commonly observed on the onset of a bifurcation [25]. In Section 3.2 we talk in detail about the saddle-node bifurcation in the RNN dynamics.

The effect of the leak rate. We conclude observing that holds $x_i^* = \tanh(M_{(i)} \cdot x^*) \iff \alpha x_i^* = \alpha \tanh(M_{(i)} \cdot x^*) \iff x_i^* + \alpha x_i^* = x_i^* + \alpha \tanh(M_{(i)} \cdot x^*) \iff x_i^* = (1-\alpha) x_i^* + \alpha \tanh(M_{(i)} \cdot x^*)$, hence the number of fixed points and their positions in phase space do not change on varying $\alpha \in (0, 1]$. However, their linear stability properties are directly affected by α.

3.2 Bifurcation of fixed points in RNNs

In what follows, we perform a bifurcation analysis of one-dimensional RNN map and provide some sufficient conditions to design two-dimensional RNNs with a desired number of fixed points. Particularly, we focus on *fold bifurcations*[3] of low-dimensional RNN maps, which constitute the only codimension-1 bifurcation that generate new fixed points; as discussed by Tiňo et al. [111], it is the usual mechanism for creating new attractive fixed point in RNNs. Moreover, as shown by Beer [10], some fold bifurcations are responsible for reducing the dimensions where the actual dynamics takes place, dividing the RNN parameter space in different regions of effective dimensionality.

[3]We refer to Kuznetsov [63] for the terminology; the fold bifurcation is also known as *saddle-node*, *limit point* or *turning point* bifurcation.

3.2.1 RNN maps in one dimension

Here, we consider the following one-dimensional map,

$$x[k] = \tanh(mx[k-1] + w), \tag{24}$$

where $(m, w) \in \mathbb{R}^2$ are the bifurcation parameters; we simplify the state-update in (13) removing explicit reference to inputs and setting zero noise. Nevertheless, studying the map (24) is still useful to obtain insights about high-dimensional input-driven RNN dynamics. In fact, in the high-dimensional case (13) the activation function of the jth neuron is determined by

$$x_j[k] = \tanh\Big(m_{jj}x_j[k-1] \;+\; \sum_{s\neq j} m_{js}x_s[k-1] + (W_{in})_{(j)} \cdot u[k] + \varepsilon_j\Big). \tag{25}$$

Hence, the parameter w in (24) can be interpreted as the weighted sum of all incoming neurons, plus input and noise terms. Fixed points of the map $F(x) = \tanh(mx + w)$ are the solutions x^* of the equation $Q(x^*) = 0$, where

$$Q(x) := F(x) - x \;=\; \tanh(mx + w) - x. \tag{26}$$

It is not possible to find a closed-form expression for the fixed points. However, it is possible to state that: (i) for every $(m, w) \in \mathbb{R}^2$, there exists at least one fixed point; (ii) there can be one, two, or three fixed points. The proof of these statements follow straightforwardly from the fact that $\lim_{x\to\pm\infty} Q(x) = \mp\infty$ and because, if $m < 1$ then Q is monotonic. Otherwise, there exist two critical points, $x_l < x_r$, namely

$$x_{l,r} = \frac{1}{m}\left[\pm \tanh^{-1}\left(\sqrt{\frac{m-1}{m}}\right) - w\right], \tag{27}$$

such that the function $Q(x)$ folds. Moreover, since $Q(x^*) = 0 \iff x^* = \tanh(mx^* + w) \in (-1, 1)$ for every $(m, w) \in \mathbb{R}^2$, all fixed points lie in the open interval $(-1, 1)$. Critical points in (27) are solutions to $Q'(x) = 0$, or equivalently to $F'(x) = 1$.

Let us assume $m > 0$. We observe a fold bifurcation whenever a critical point assumes the zero value. The condition $Q(x_{l,r}) = 0$ gives rise to the following parametrization of the fold bifurcation curve,

$$w_\pm(m) := \pm\left[m\sqrt{\frac{m-1}{m}} \;-\; \tanh^{-1}\left(\sqrt{\frac{m-1}{m}}\right)\right], \qquad m \in [1, +\infty), \tag{28}$$

which possesses two symmetric branches ending on a cusp; see Figure 7 for an illustration. Crossing that curve in parameter space towards the region containing the semi-axis of $m > 1$, a new fixed point is formed (x_l or x_r, depending on the branch) and splits in a pair of fixed points, one stable and one unstable[4]. This curve delimits the boundary between a dynamic regime where two stable points coexist with an unstable one, and a regime where only one fixed point exists. Due to symmetry, in the $m < 0$ case, there are two bifurcation branches identifying a *flip* or *period doubling bifurcation*, which is analytically described by the curve $w_{\pm}(-m)$ with $m \leq -1$. Crossing that curve towards the region containing the semi-axis of $m < -1$, a stable fixed point loses stability and gives rise to a 2-periodic attracting trajectory surrounding it.

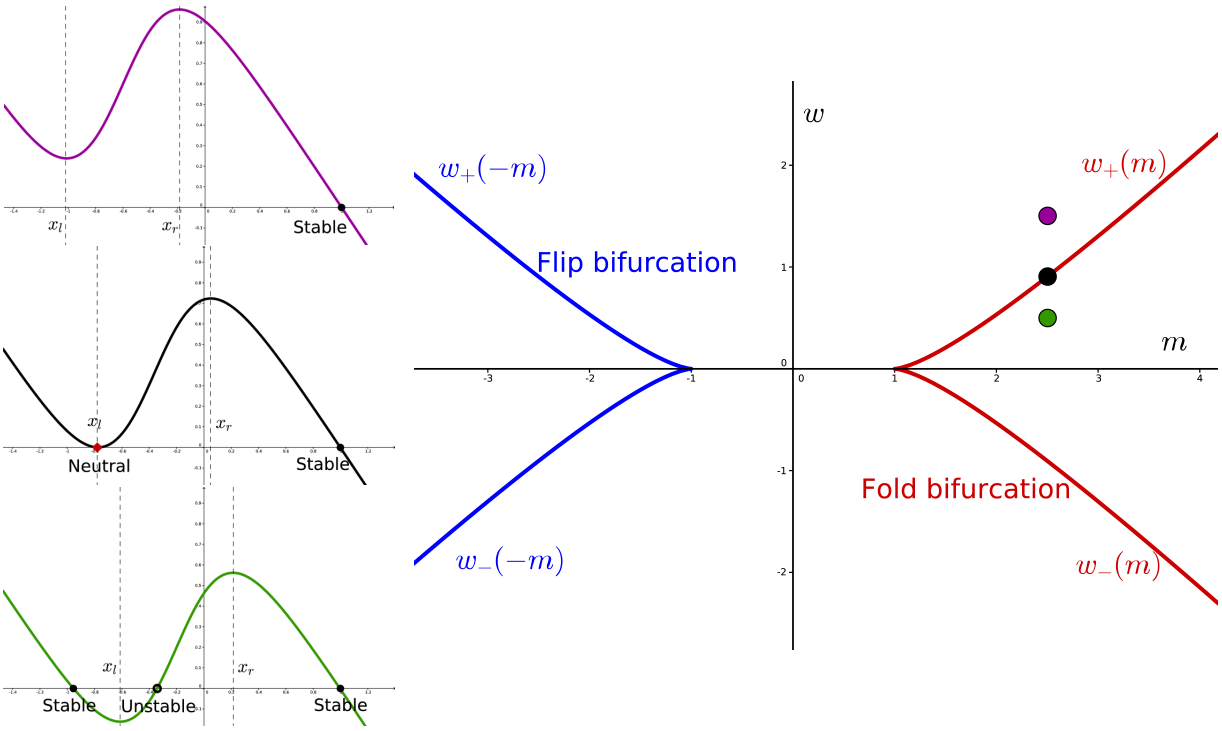


Figure 7: **Left:** three different folding configuration of the function $Q(x)$ in (26) with $m = 2.5$; purple $w = 1.5$, black $w \approx 0.9$, green $w = 0.5$. **Right:** bifurcation diagram of fixed points in one-dimensional RNN.

[4]The particular case of crossing that curve through the cusp gives rise to a pitchfork bifurcation of the origin.

3.2.2 RNN maps in two dimensions

Here, we consider a two-neuron reservoir. We denote the activation functions of these neurons as x and y, so that the RNN state evolution defines a trajectory $(x[k], y[k]) \in [-1, 1]^2, k = 1, 2, ...$, ruled by:

$$\begin{aligned} x[k] &= \tanh(ax[k-1] + by[k-1])\,, \\ y[k] &= \tanh(cx[k-1] + dy[k-1])\,. \end{aligned} \tag{29}$$

It is know that a fixed point can undergo only fold or flip bifurcations in one-dimensional, discrete-time dynamical systems [63]. Nevertheless, considering two or more dimensions, also *Neimark-Sacker* bifurcations could occur, where a stable point loses stability and an invariant curve surrounding it is created. These are the only possible codimension-1 bifurcations of a fixed point for a discrete-time dynamical system. Among them, only the fold bifurcation can generate new fixed points. In general, the origin point of the autonomous system defined by (22) is always a fixed point. Considering (29), the Jacobian matrix evaluated on the origin reads

$$W_r = \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

Therefore, training RNNs via (12) implies a qualitative change of the dynamics around the origin if some eigenvalue crosses the unit circle, i.e., fixed point $(0, 0)$ bifurcates. However, adding a low-rank matrix to the reservoir can induce global bifurcations far away from the origin as well; hence, we cannot rely on the local bifurcation of the origin to deduce the global attractor structure after training. As a counterexample, suppose $\lambda_1, \lambda_2 > 1$. The diagonal matrix $\mathrm{diag}(\lambda_1, \lambda_2)$ and the upper triangular matrix $\begin{pmatrix} \lambda_1 & \gamma \\ 0 & \lambda_2 \end{pmatrix}$ share the same spectrum $\{\lambda_1, \lambda_2\}$. Nevertheless, if the coupling is strong enough, that is, $|\gamma| > \dfrac{w_+(\lambda_1)}{y^*}$, where $w_+(\lambda_1)$ is the function (28) evaluated on λ_1 and y^* is the positive stable solution of $y[k] = \tanh(\lambda_2 y[k-1])$, then the fold bifurcation curve is crossed, making the dynamics for x variable trivial. As a consequence of this, the upper triangular matrix induces dynamics with just two attractors (plus two saddles and the origin is a repeller), while the diagonal matrix induces four attractors (plus four saddles and the repeller).

In order to count the number of fixed points of the map (29), we can draw its nullclines. Defining

function $N_{\alpha,\beta}(\eta) := \dfrac{1}{\beta}[-\alpha\eta + \tanh^{-1}(\eta)]$, the nullclines are given by

$$\begin{aligned} y &= N_{a,b}(x), \\ x &= N_{d,c}(y), \end{aligned} \tag{30}$$

and they represent the locus of points where x-dynamic and y-dynamic is stationary. As a consequence, the solutions of the algebraic nonlinear system (30) coincide with the set of fixed points. In the last part of this subsection, we show some sufficient conditions to control the number of fixed points assuming $a, d > 1$, i.e., when both nullclines are folding. We refer to Figure 8 for a graphical illustration.

- Case $bc \geq 0$.

  $$|N'_{a,b}(0)| < |N'_{d,c}(0)|^{-1}, \text{ i.e. } (1-a)(1-d) < bc \quad \Longrightarrow \quad \text{there are exactly 3 fixed points.} \tag{31}$$

  On the other hand, when $(1-a)(1-d) \geq bc$, there could exist 5, 7 or 9 fixed points. Critical points of the function $N_{\alpha,\beta}(\eta)$ are $\eta_\pm = \pm\sqrt{\frac{\alpha-1}{\alpha}}$. Therefore, if $(1-a)(1-d) \geq bc$ holds then

  $$\left|N_{a,b}\left(\sqrt{\frac{a-1}{a}}\right)\right| < \sqrt{\frac{d-1}{d}} \text{ or } \left|N_{d,c}\left(\sqrt{\frac{d-1}{d}}\right)\right| < \sqrt{\frac{a-1}{a}} \quad \Longrightarrow \quad \text{there are exactly 5 fixed points.} \tag{32}$$

  Top right graph of Figure 8 depicts a nullcline configuration where there are 5 intersections[5]. From that configuration, we can make the humps of $y = N_{a,b}(x)$ more pronounced by increasing the $\frac{a}{b}$ ratio, until a fold bifurcation occurs, giving rise to a new couple of pair of fixed points with a total of 9 fixed points, centre right graph of Figure 8. The case of 7 fixed points is obtained exactly at the onset of the fold bifurcation, centre left graph of Figure 8. Clearly, this last case is not structurally stable since arbitrarily small perturbations of the nullclines make the number of intersections change.

- Case $bc < 0$.

  $$\left|N_{a,b}\left(\sqrt{\frac{a-1}{a}}\right)\right| < \sqrt{\frac{d-1}{d}} \text{ and } \left|N_{d,c}\left(\sqrt{\frac{d-1}{d}}\right)\right| < \sqrt{\frac{a-1}{a}} \quad \Longrightarrow \quad \text{there is exactly 1 fixed point.} \tag{33}$$

[5]Note that it holds $N_{\alpha,\beta}\left(\pm\sqrt{\frac{\alpha-1}{\alpha}}\right) = -\frac{1}{\beta}w_\pm(\alpha)$.

From the above configuration, increasing the d/c ratio makes stretch the humps of $N_{d,c}$ until a fold bifurcation occurs producing exactly three fixed points. Beyond the fold bifurcation, a new pair of fixed points is generated.

$$\left|N_{a,b}\left(\sqrt{\frac{a-1}{a}}\right)\right| < \sqrt{\frac{d-1}{d}} \text{ and } \left|N_{d,c}\left(\sqrt{\frac{d-1}{d}}\right)\right| > 1 \quad \Longrightarrow \quad \text{there are exactly 5 fixed points.}^{6} \tag{34}$$

In both cases, a simple sufficient condition to ensure the existence of 9 fixed points, see Figure 8, is

$$\left|N_{a,b}\left(\sqrt{\frac{a-1}{a}}\right)\right| > 1 \text{ and } \left|N_{d,c}\left(\sqrt{\frac{d-1}{d}}\right)\right| > 1 \quad \Longrightarrow \quad \text{there are exactly 9 fixed points.} \tag{35}$$

The maximum number of nullcline intersections is 9, setting the maximum number of fixed points that can be generated in a two-dimensional RNN map. In general, this indicates that for a N-dimensional autonomous RNN of the type (22), there can be at most 3^N fixed points of which a maximum number of 2^N stable fixed points, 1 repeller and $3^N - 2^N - 1$ saddles. An interesting study of multistability in continuous-time RNNs with delayed coupling can be found in [23]. There, they provide some sufficient conditions ensuring the maximum number of 2^N asymptotically stable points for Hopfield-type RNNs.

[6] Due to symmetry, this condition holds also when inverting the role of $N_{d,c}$, $N_{a,b}$ and consequently (a, b), (d, c).

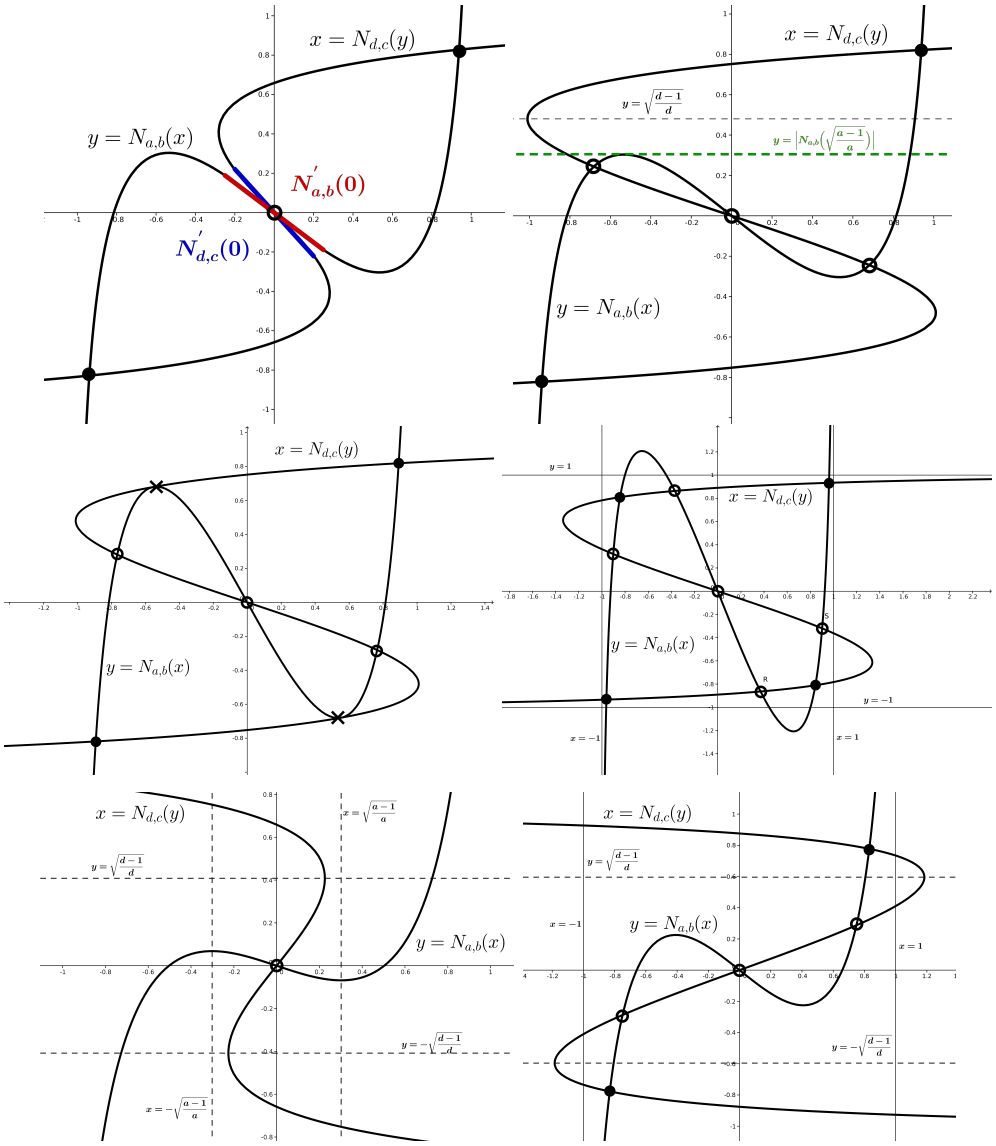


Figure 8: In all panels, filled points denote stable attractors, while circles denote saddles or repellers. **Top left:** geometric representation of the sufficient condition (31); nullcline slopes on the origin are highlighted with different colours. **Top right:** illustration of the condition (32); the black dashed line acts as an upper bound ensuring that the maximum height of the peak of the x-nullcline (represented by the green dashed line) does not cross further the y-nullcline. **Centre left:** special case of 7 fixed points. Crosses indicate neutral fixed points where fold bifurcations take place. **Centre right:** an example of nullclines configuration holding the condition (35) in the case of $bc > 0$. Both curves come out of the invariant square $[-1, 1]^2$ with their humps, that guarantees 9 intersections, i.e., 9 fixed points. **Bottom left:** depiction of the sufficient condition (33); the origin is the only fixed point and it is unstable. **Bottom right:** illustration of the sufficient condition (34).

3.3 Network attractors and finite state computation

The aim of this section is to provide a mathematical framework for modelling multi-stable dynamics between fixed points profitable for developing finite state computation machines. The idea to interpret a system whose dynamics in phase space is regulated by a number of local attractors as a general content-addressable memory can be traced back to the seminal work of Hopfield [47]. The idea is that different briefly presented stimuli would push the network into one of the different stimulus-specific basin of attraction states, hence maintaining an active representation of the stimulus even after its removal and therefore guiding actions in forthcoming choice situations. Hopfield analysis was based on the definition of a Lyapunov function representing the potential energy of the neural network. As time flows the Lyapunov function monotonically decreases, hence the system state tends towards a stable point where it corresponds a local minimum value of the energy. Nevertheless, Hopfield assumed no self-connections in the network, i.e. $(W_r)_{ii} = 0, \forall i$, and a perfectly symmetric network, i.e. $(W_r)_{ij} = (W_r)_{ji}, \forall i, j$, while in our context the recurrent matrix W_r is randomly generated according to a uniform distribution of values in $[-1, 1]$. Some interesting results involving Lyapunov functions of neural networks that relate with the Hopfield works can be found in [117]. Some other interesting works investigating networks-like models from the dynamical systems perspective are: [112] where the link between RNNs and (not necessarily finite) state machines is explored, the seminal work on dynamical recognizers of [90], and [109] where the author shows how to use fractal sets to encode context-free grammars.

More complex attractors consisting of networks of invariant sets in phase space have been proposed in the literature [84, 118]. Such models found renewed interest in neuroscience [81, 114] and other fields of research, as they provide a fundamental tool to describe dynamic processes occurring on transients that explore excitable connections. More relevant to our paper, we focus on networks of stable fixed points that are connected by excitable connections. Following [5, 6], we say that there exists an *excitable connection* for amplitude $\delta > 0$ from stable fixed point $\mathbf{p}_i$ to $\mathbf{p}_j$ whenever

$$B_\delta(\mathbf{p}_i) \cap W^s(\mathbf{p}_j) \neq \emptyset, \tag{36}$$

where $B_\delta(\mathbf{p}_i)$ stands for the closed ball centred on $\mathbf{p}_i$ with radius $\delta > 0$ and $W^s(\mathbf{p}_j)$ denotes the stable set of $\mathbf{p}_j$, which is the manifold corresponding to its basin of attraction whenever $\mathbf{p}_j$ is hyperbolic, see Definition 3.2.

Definition 3.3 *We define* excitability threshold *[5] (or just* threshold*) of the excitable connection from* $\mathbf{p}_i$ *to* $\mathbf{p}_j$*, and denote it as* $\delta_{th}(\mathbf{p}_i, \mathbf{p}_j)$*, the following nonnegative real number:*

$$\delta_{th}(\mathbf{p}_i, \mathbf{p}_j) := \inf\{\delta > 0 \ : \ B_\delta(\mathbf{p}_i) \cap W^s(\mathbf{p}_j) \neq \emptyset\}. \tag{37}$$

Remark 3.2 *The quantity* (37) *can be informally interpreted as the fact that the fixed point* $\mathbf{p}_i$ *is* $\delta_{th}(\mathbf{p}_i, \mathbf{p}_j)$ *away from the basin of* $\mathbf{p}_j$*; see Figure 9 (left picture) for a visual explanation.*

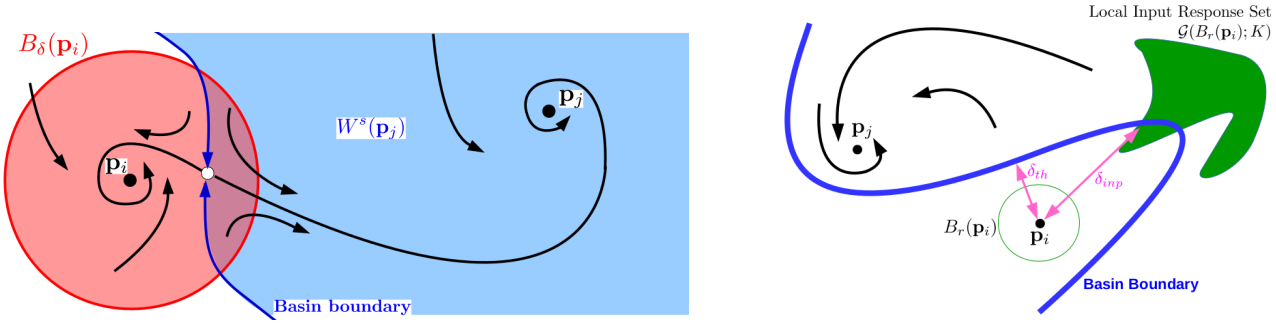


Figure 9: **Left:** Depiction of an excitable connection from $\mathbf{p}_i$ to $\mathbf{p}_j$. The red area is a closed ball centred on $\mathbf{p}_i$ of radius δ. The blue area represents the stable manifold of $\mathbf{p}_j$, i.e., its basin of attraction. The open circle represents a saddle whose stable manifold (blue curves) denotes the boundary of the basin of attraction of $\mathbf{p}_j$. Some points of $B_\delta(\mathbf{p}_i)$ converge to $\mathbf{p}_i$ itself, while those points beyond the basin boundary converge towards $\mathbf{p}_j$, as suggested by the black arrows. **Right:** Representation of the activation of an excitable connection through the action of the input which allows to accomplish the switch from the stable point $\mathbf{p}_i$ to the stable point $\mathbf{p}_j$. Firstly, the internal state of the RNN stands nearby the stable point $\mathbf{p}_i$, in the neighbourhood $B_r(\mathbf{p}_i)$. Then, a control input $u[k+1] \in K$ drives the current internal state $x[k]$ to the next state $x[k+1]$ inside the local input response set, represented as the green subregion. Finally, if the state falls beyond the basin boundary then the internal state converges to the stable point $\mathbf{p}_j$. Excitability threshold $\delta_{\text{th}}(\mathbf{p}_i, \mathbf{p}_j)$, in (37), is computed along the direction where the distance is shortest in order to escape from the basin of attraction of $\mathbf{p}_i$ and converge to $\mathbf{p}_j$. Nevertheless, input can potentially drive the dynamics towards alternative dimensions for the purpose of achieving the switch from $\mathbf{p}_i$ to $\mathbf{p}_j$. The input-driven excitability threshold $\delta_{\text{inp}}(\mathbf{p}_i, \mathbf{p}_j)$, in (40), is computed considering the subspace exploited by the input to solve the task.

ENAs and finite state computation. In [7], it is constructively proved that for any given directed graph G with N vertices it is possible to design an $N \times N$ matrix W_r such that the input-free continuous-time RNN of (2) has an Excitable Network Attractor (ENA) with threshold δ which is a realisation of G embedded in phase space $\mathbb{R}^N$; see definition 3.4 for the definition of an ENA. Such a result basically gives us a way to construct by hand a RNN model (2) able to reproduce any finite state computation machine. Essentially, if we want to realise a graph with N vertices then it suffices (but not needed) a RNN of dimension N, i.e. one for each vertex of the graph. Of course, N dimensions are not necessary in general, as a counter-example the nullcline configuration of the centre right picture of Figure 8 defines an ENA which is a realisation of a 4-vertices graph but the RNN is only 2-dimensional.

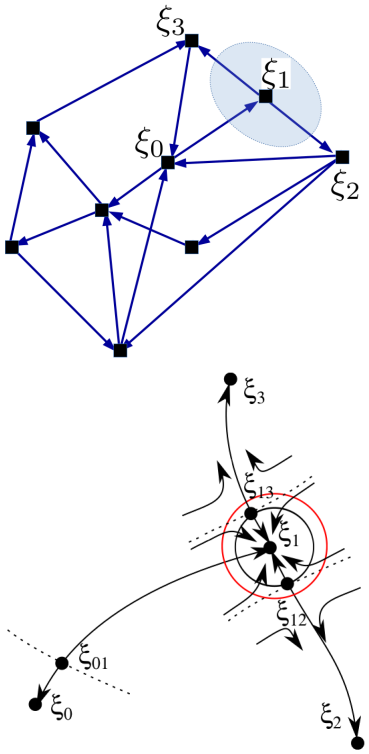


Figure 10: **Top:** an ENA is an invariant set of a dynamical system with a graph-like structure. Nodes of the graph corresponds to the local attractors (black squares in the picture). The directed (blue) edges corresponds to the excitable connections between the local attractors. **Bottom:** an example of phase portrait around the attractor ξ_1. The attractor ξ_1 is a stable point. The saddle ξ_{01} funnel trajectories around its stable manifold and then towards the stable point ξ_1. In absence of stimuli, the dynamics get stuck on ξ_1. The excitability threshold of ξ_1 is measured by the radius of the black circle in the picture. As soon as input (or noise) has an amplitude greater than the excitability threshold and a direction such that it drives the state of the system beyond the stable manifold (dashed line) of the saddle ξ_{12} then the system switch towards the attractor ξ_2. Analogously, if the direction of input (or noise) lead to cross the stable manifold (dashed line) of the saddle ξ_{13} then the system switch to the attractor ξ_3.

Definition 3.4 *A set $X_{exc} \subset \mathbb{R}^{N_r}$ is called an* excitable network attractor *(ENA) for amplitude $\delta > 0$ if there exists a collection of fixed points $\{\mathbf{p}_i\}_{i=1}^{M}$, such that*

$$X_{exc}(\{\mathbf{p}_i\}_{i=1}^{M}, \delta) := \bigcup_{\substack{i,j=1 \\ i \neq j}}^{M} \left\{F^h(B_\delta(\mathbf{p}_i))\right\}_{h \geq 0} \cap W^s(\mathbf{p}_j), \tag{38}$$

where $F(B_\delta(\mathbf{p}_i)) := \{F(x) \mid x \in B_\delta(\mathbf{p}_i)\}$ is based on (22) *(see [5, 6]).*

The autonomous dynamics on such a set converges to one of the stable fixed points; hence, external stimuli as inputs or noise are necessary to get interesting dynamics. If the external input (or noise) is large enough, then the state will escape from the current basin of attraction and switch to a different one, until another sufficiently large input will lead to another change of basin. This peculiar dynamics are referred to as *metastable* and have been proposed to explain the execution of cognitive processes [92].

3.3.1 Accounting for the inputs

The excitability threshold (37) is defined as the (Euclidean) distance between a given stable fixed point, $\mathbf{p}_i$, and the basin of attraction of another fixed point, say $\mathbf{p}_j$. Such a quantity measures the minimum distance in phase space necessary to escape from the basin of $\mathbf{p}_i$ and converge towards $\mathbf{p}_j$. Nevertheless, if the dynamical system is high dimensional, then there will be a large number of possible escaping directions that could be exploited by inputs. Therefore, excitability thresholds (37) alone may not be representative of nonautonomous systems driven by inputs. For this purpose, in order to take into account the action of inputs on the dynamics, we introduce the notion of *input-driven excitability threshold* of an excitable connection. Considering K as a compact subspace of $\mathbb{R}^{N_i}$, we define $\mathcal{G}(\mathbf{x}_0; K) := \bigcup_{\mathbf{u} \in K} G(u, x_0)$, where G is the function in (18) defining the nonautonomous dynamical system which describes the trained neural network. The set $\mathcal{G}(\mathbf{x}_0; K)$ contains all states reachable from x_0 under the action of input values $u \in K$. Importantly, the presence of noise has the effect to perturb away the internal state from the exact location of the stable point in the deterministic counterpart of the dynamics. Therefore, instead of $\mathcal{G}(\mathbf{p}_i; K)$ we rather observe the following set.

Definition 3.5 *We call* local input response set *of the stable point $\mathbf{p}_i$ the subset of phase space*

defined by

$$\mathcal{G}(B_r(\mathbf{p}_i); K) := \bigcup_{x \in B_r(\mathbf{p}_i)} \mathcal{G}(x; K), \tag{39}$$

where the radius r can be shrunk according to the amplitude of the noise and K is a subset of admissible input values for the task at hand.

Remark 3.3 *Continuity of G implies that, if $r \to 0^+$, then the monotonically decreasing sequence of sets $\{\mathcal{G}(B_r(\mathbf{p}_i); K)\}_{r \geq 0}$ converges to $\mathcal{G}(\mathbf{p}_i; K)$ regardless of $K \subset \mathbb{R}^{N_i}$. Furthermore, we note that $\mathcal{G}(B_r(\mathbf{p}_i); K)$, as a subspace of $\mathbb{R}^{N_r}$, is compact if $K \subset \mathbb{R}^{N_i}$ is compact.*

If K represents all possible inputs of a particular task[7], then we can drop it and denote (39) simply as $\mathcal{G}(B_r(\mathbf{p}_i))$. Therefore, the subregion of the phase space represented by the local input response set $\mathcal{G}(B_r(\mathbf{p}_i))$, encodes the action exercised by the input when the internal state of the RNN is nearby the stable point $\mathbf{p}_i$.

Definition 3.6 *We define* input-driven excitability threshold *of the excitable connection from $\mathbf{p}_i$ to $\mathbf{p}_j$, and denote it as $\delta_{inp}(\mathbf{p}_i, \mathbf{p}_j)$, the following nonnegative real number:*

$$\delta_{inp}(\mathbf{p}_i, \mathbf{p}_j) := \inf\{\delta > 0 \ : \ \mathcal{G}(\mathbf{p}_i) \cap B_\delta(\mathbf{p}_i) \cap W^s(\mathbf{p}_j) \neq \emptyset\}. \tag{40}$$

Remark 3.4 *The excitability threshold in* (40) *has a similar meaning to the one defined in* (37)*, although it considers only the subspace exploited by the action of inputs nearby $\mathbf{p}_i$, where the excitable connection starts from, see Figure 9. Finally, from the fact that $\mathcal{G}(\mathbf{p}_i) \cap B_\delta(\mathbf{p}_i) \cap W^s(\mathbf{p}_j) \subseteq B_\delta(\mathbf{p}_i) \cap W^s(\mathbf{p}_j)$, it follows that $\delta_{th}(\mathbf{p}_i, \mathbf{p}_j) \leq \delta_{inp}(\mathbf{p}_i, \mathbf{p}_j)$ holds for all pairs of fixed points.*

[7]For example, in the flip-flop task that we will introduce in section 4.1 we have $K = \{(0,0), (1,0), (-1,0), (0,1), (0,-1)\}$

4 Towards a graph-based characterisation of input-driven RNN dynamics

> A fundamental problem in RNNs is to design attractors and their basins of attraction such that the dynamics of the networks (possibly driven by external inputs) leads to transitions among the attractors so that some computation is performed.
>
> Xufei Wang - Edward K. Blum

The high-dimensional and non-linear nature of RNNs complicates interpretability of their internal dynamics, which are characterised by complex, input-dependent spatio-temporal patterns of activity [94, 107]. This poses constraints on understanding the behaviour of RNNs: they are usually viewed as black-boxes from which it is hard to extract useful knowledge about their inner workings. As highlighted by recent research efforts [18, 55, 83], similar interpretability issues affect many other machine learning methods. Furthermore, an increasing societal need to develop accountability and explainability of decision making by AI [38] is driving the development of methodologies for explaining the behaviour of such methods.

The aim of this chapter is to develop effective models that capture the essential dynamical behaviour of RNNs on computational tasks as input-driven responses of a dynamical system, while neglecting microscopic details of the RNN dynamics in phase space (i.e., the space of all possible neuron activations). To this end, we hypothesise that RNNs can undertake computations by exploiting (i) transient dynamical regimes and (ii) excitable connections to switch between different stable attractors, depending on input and current state. The RNN behaviour depends on the task at hand: for example, long transients are mostly exploited in time series forecasting problems, while switching between attractors is mostly exploited in classification problems or tasks requiring to learn a finite set of memory states. These mechanisms are not mutually exclusive and can be exploited synergistically to realise more complex computations.

In order to test our hypothesis, we develop a theoretical framework based on network attractors

of dynamical systems. An attractor is a subset of the state space of a dynamical system (i.e., the phase space), where the state will asymptotically converge for "usual" choices of initial conditions. Network attractors are special kinds of attractor which can be thought of as directed graphs, where nodes correspond to local attractors and directed edges corresponds to particular trajectories that connect the basins of attraction of those local attractors. Since for this chapter, the local attractors of interest are all stable fixed points, we will use the term local attractor synonymously with stable fixed point. A stable fixed point is the simplest possible attractors: this is a point in phase space where all variables of the dynamical system assume constant values, and all close enough initial conditions converge to this point. However, fixed points need not be stable: they can be partially stable (saddle points), or totally unstable (repellers). According to the nature of the trajectories connecting local attractors it is possible to consider both heteroclinic network attractors composed of heteroclinic connections between saddles (i.e. partially stable fixed points), and excitable network attractors (ENAs) composed by connections that are excitable, i.e. that require a small initial perturbation to move between stable attractors [5, 118].

Heteroclinic network attractors have already been suggested as models to describe transient computation [92, 93], here, conversely, we focus attention on ENAs which have the advantage of being more robust to perturbations. We focus particularly on ENAs between stable fixed points [5], although ENAs can be theoretically conceived between any kind of local attractor: limit cycles, limit tori, strange attractors, etc. More precisely, we show how to construct ENAs describing RNN behaviour for tasks that involve switching between a finite set of attractors. Interestingly, from a directed graph (representing the underlying network attractor of a dynamical system) it is possible to reverse engineer [4] a set of ordinary differential equations ruling, in our case, the RNN behaviour. This, in turn, opens the way to a more analytic description of how RNNs solve computational tasks.

The original contributions of this chapter can be summarised as follows:

- We provide a theoretical framework to describe the behaviour of RNNs by means of ENAs. The proposed theoretical framework is general and covers several types of computational tasks. We present a specific instance of such a framework applied to describe RNN behaviour on tasks requiring to learn a finite set of memory states;
- We use bifurcation analysis to manually design (low-dimensional) ESNs that give rise to ENAs

able to solve the flip-flop task, a benchmark task we found in the literature [107] that we use in this chapter to validate our hypothesis. This allows us to justify our choice of modelling framework and suggests its validity in the context of RNNs;

- As ESNs are driven by inputs and subject to training via perturbation matrices, bifurcation analysis alone is not sufficient to explain changes to their behaviour. In fact, inputs and perturbation matrices can affect ESN behaviour in a non-trivial way. To this end, we introduce an algorithm to extract the ENA describing the ESN behaviour for the task at hand. The algorithm analyses an ESN trajectory and constructs a directed graph encoding the underlying ENA on which the dynamics takes place. The vertices (nodes) of such a graph are associated with the fixed points of the dynamics and directed edges describe excitable connections between them. We apply this algorithm to an ESN that is trained to solve the flip-flop task and show that the resulting ENA is able to explain how ESNs perform computations in a detailed and mechanistic way;

- We define a notion of excitability threshold for this high-dimensional, non-linear dynamical system driven by inputs. We propose a method for computing this excitability threshold that accounts for inputs and can directly be applied to trajectories generated by a neural network while solving a task;

- Our simulations suggest three important findings. First, as already noted by recent research [2, 107], the dynamics of high-dimensional RNNs takes place in a much lower-dimensional phase space region that is determined by the structure introduced with training and inputs. Here, we observe that the dynamics is indeed low-dimensional, but highlight the fact that additional dimensions are used occasionally to switch between stable states according to control inputs. This suggests that, for example, a simplistic use of methods based on explained variance to reduce dimensions needs to be avoided. Second, we show how ENA models describing RNN behaviour can be exploited to provide a mechanistic interpretation of errors occurring during the computation undertaken by RNNs. Finally, we show how excitability thresholds of the extracted ENAs allow us to assess the robustness of RNNs to noise-induced perturbations.

The remainder of this chapter is organised as follows. In section 4.1 is described the flip-flop

task. Section 4.2 shows how to design low-dimensional ESN models that give rise to ENAs able to solve the flip-flop task. In section 4.3 we propose an algorithm for automatically extracting ENAs directly from an ESN trajectory generated while solving a task. In section 5.5, we present and discuss results of the simulations. Finally, section 4.5 draws conclusions and points to future research directions. Appendix A provides details of the procedure used to determine fixed points.

4.1 Flip-flop task

In this chapter, we focus on the flip-flop task with two bits [107], since it provides a controlled workbench to test our hypothesis. Roughly speaking, the flip-flop task requires to generate an output signal that resembles a square wave, see Figure 11. The output can assume 2 values, namely $+1$ or -1, and the switch from an output value to another is driven by input pulses. Therefore, in the n-bit flip-flop task there is an n-dimensional input signal where each component is zero apart from some random instantaneous pulses of amplitude $+1$ and -1 (independently generated for any of the n components). The RNN must learn to generate an n-dimensional output signal where each component can assume values of $+1$ or -1, and switch between these two values according to the sign of the input pulses. More precisely, the input to discrete-time RNN is assumed to be a discrete temporal sequence with values from $\{+1, 0, -1\}^{N_i}$, where N_i is the positive integer dimension of the input. In general, we consider discrete time k varying input, denoted as N_i-dimensional vector $u[k]$, and output, denoted as N_o-dimensional vector $z[k]$, where N_o in the dimension of the output, of a system related by equations (10), (11), (12), (13) with evolving internal state, namely a N_r-dimensional vector denoted as $x[k]$, and connection weights that are optimised during the training phase. We consider response to inputs where, independently for any k, $u[k]$ is $(0, 0)$ with probability $1 - p$ or otherwise chosen to be one of the four inputs $(\pm 1, 0)$ or $(0, \pm 1)$ with equal probability. This generates a sequence of inputs that remain at $(0, 0)$ for an exponentially distributed period of time, but occasionally takes one of the four values $(0, \pm 1)$, $(\pm 1, 0)$. The target to be learned by the RNN is a two-dimensional vector $y[k]$ that can assume four possible configurations: $(1, 1)$, $(1, -1)$, $(-1, 1)$, and $(-1, -1)$. The trained system (11), (12), (13) is considered to successfully accomplish the task if the network is able to reliably and accurately reproduce all 20 possible actions described in Table 1.

Note that inputs $u[k]$ in (13) play the role of control inputs. Recall that we use RNNs that are

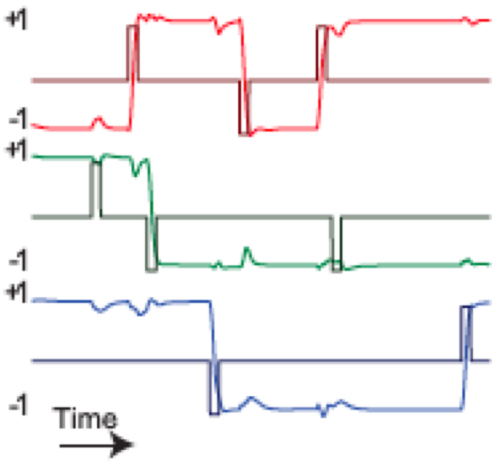


Figure 11: Example inputs and outputs for the 3-bit flip-flop task. Sample input and output to the 3-bit flip-flop network. The three inputs and outputs are shown in pairs (dark red/red, dark green/green, and dark blue/blue, respectively). Brief input pulses come in with random timing. For a given input-output pair, the output should transition to $+1$ or stay at $+1$ for an input pulse of $+1$. The output should transition to -1 or stay at -1 if the corresponding input pulse is -1. Finally, all three outputs should ignore their nonmatching input pulses. Caption and figure taken from [107].

Table 1: The two-bit flip-flop task. Depending on the output $z[k-1]$ and input $u[k]$, we expect the output $z[k]$ to become as given in the table, where N.C. indicates "no change" from the current output value.

Outputs\Inputs	$(0,0)$	$(1,0)$	$(-1,0)$	$(0,1)$	$(0,-1)$
$(1,1)$	N.C.	N.C.	$(-1,1)$	N.C.	$(1,-1)$
$(1,-1)$	N.C.	N.C.	$(-1,-1)$	$(1,1)$	N.C.
$(-1,1)$	N.C.	$(1,1)$	N.C.	N.C.	$(-1,-1)$
$(-1,-1)$	N.C.	$(1,-1)$	N.C.	$(-1,1)$	N.C.

ESNs subjected to supervised training with a perturbation matrix obtained by injecting the output into the ESN dynamics. This choice does not limit the validity of our hypothesis: we claim that our results are general and can be used to describe the behaviour of more general discrete-time, input-driven RNNs.

4.2 Designing low-dimensional ESNs to solve flip-flop tasks

In this section, starting from ESN models we show how to manually design ENAs that realise computations needed for the flip-flop task. This step allows us to justify the choice of the modelling framework adopted here and its validity in the context of RNNs. For this purpose, we rely on bifurcation theory; see section 3.2 for technical notions. A bifurcation [63] is a qualitative change in the solution set (phase space topology) on changing a parameter of a dynamical system. On varying the parameters of the model, if such qualitative changes appear, then we say the system has undergone a bifurcation. For this reason, the notion of bifurcation plays an important role in training RNNs and in describing their behaviour. Training the recurrent layer of RNNs corresponds to shaping their phase space topology, possibly inducing bifurcations that lead to a qualitative change in behaviour. We argue that adding a low-rank perturbation matrix to the reservoir (12) can induce bifurcations that lead to the creation of attracting regions in ESN phase space useful to solve the task at hand. Clearly, this can also happen with more sophisticated training algorithms. Let us assume the origin as the only global attractor for the autonomous system $x[k] = (1 - \alpha)x[k-1] + \alpha\phi(W_r x[k-1])$ associated with (10). Then, the bifurcation induced by (12) leads to a transition from such a trivial dynamic to one where the origin becomes unstable[8] and repels trajectories towards other attracting regions in phase space, where the nonautonomous dynamics actually takes place. We note that the flip bifurcation, see section 3.2, is detrimental for the flip-flop task considered here, as it gives period rather than fixed point attractors.

In Section 4.2.1, we provide a low-dimensional example of an ESN that is able to solve the flip-flop task; the reservoir of such ESN is formed by two neurons. In Section 4.2.2, instead we design an ESN model with a reservoir formed by $2k$ neurons which is able to solve the flip-flop task with k bits. For both examples we show there is an underlying ENA that explains their dynamics.

4.2.1 A minimal-dimension example to solve the two-bit flip-flop task

In order to solve the k-bit flip-flop task, one needs to learn 2^k memory states (stable fixed points) and the related switching patterns dictated by control inputs. Here, we show how to design an ESN with two neurons in the recurrent layer, giving rise to an ENA able to solve the flip-flop task with

[8]The origin could in principle remain stable and other attractors appear elsewhere: an example can be found in [120].

$k = 2$ bits. According to the analysis of Section 3.2, we know that it is possible to obtain, with k neurons, up to 3^k fixed points, 2^k of which are stable, $3^k - 2^k - 1$ are saddles, and 1 (the origin) is a repeller. Therefore, we set the trained reservoir weights (12) according to condition (35). In particular,

$$M(b) := \omega_r \begin{bmatrix} 1 & b \\ b & 1 \end{bmatrix} \tag{41}$$

with $\omega_r = 3$ scaling the reservoir weights, and b acting as tuning parameter. As shown in Figure 12, for every $0 \leq b < 0.47$, the autonomous system $x[k] = \tanh\big(M(b)x[k-1]\big)$ has four stable attractors located near the vertices of the invariant square $[-1, 1]^2$.

The input is injected into the ESN via the following weight matrix, $W_{in} = \omega_{in} I_2$, i.e. a scaled 2×2 identity matrix, setting the scaling factor (called a hyperparameter in machine learning) to $\omega_{in} = 6$. Let us set the remaining parameters in (13) as $\boldsymbol{\epsilon} = 0$ and $\alpha = 1$, and let the ESN output (11) be the identity mapping. Let us assume that, at time k, the system is in state $\mathbf{p} = (p_1, p_2)$, which is close to one of the four attractors, i.e., $\mathbf{p}_1 \approx (1, 1)$, $\mathbf{p}_2 \approx (-1, 1)$, $\mathbf{p}_3 \approx (-1, -1)$, $\mathbf{p}_4 \approx (1, -1)$. The possible, non-null inputs at time-step k are $\mathbf{u}[k] \in \{(1, 0), (0, 1), (-1, 0), (0, -1)\}$. The action of the input pulse is encoded in a vector $\boldsymbol{\Delta} \in [-1, 1]^2$, representing the difference between the state before and after the occurrence of such a pulse, whose components are

$$\Delta_j \approx \tanh\big(3p_j + 6u_j\big) - \tanh\big(3p_j\big), \quad j = 1, 2, \tag{42}$$

considering $\omega_r = 3$, $\omega_{in} = 6$ and, for the sake of simplicity, $b = 0$, which corresponds to the case of two independent neurons. Hence, the switching mechanism between attractors is ruled by the signs of both p_j and u_j; the former encoding the position and the latter the direction where to move. Therefore, the state changes if and only if p_j and u_j assume different signs for some $j \in \{1, 2\}$. In fact, if they have the same sign, then (42) is null because $\tanh\big(\mathrm{sgn}(p_j)[3 + 6]\big) \approx \tanh\big(\mathrm{sgn}(p_j)3\big)$ due to saturating activation functions. While, if they have different sign, then (42) becomes close to -1 or 1, according to $\mathrm{sgn}(p_j)$, because $\tanh\big(\mathrm{sgn}(p_j)[3 - 6]\big) = -\tanh\big(\mathrm{sgn}(p_j)3\big)$. Clearly, it is not necessary that $|\omega_r - \omega_{in}| = \omega_r$ holds, as in our set up. Although we assumed $b = 0$ for clarity of explanation, the conclusion is the same for each $b \in [0, 0.47)$. However, when the parameter approaches the bifurcation value $b \approx 0.47$, the escaping dynamics from certain fixed points become significantly slower.

Given a set of stable fixed points, a $\delta > 0$ gives rise to a specific ENA; see definition of ENA

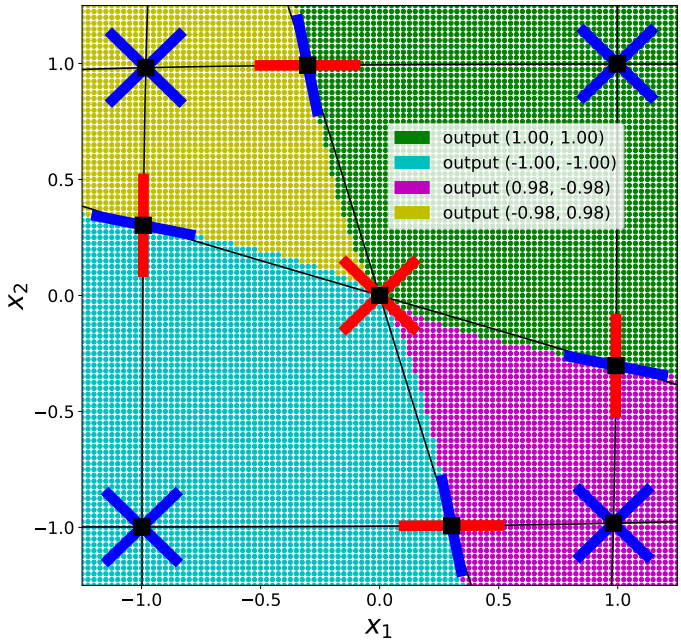


Figure 12: Basins of attraction, nullclines, fixed points and corresponding eigenvectors of the linearized two-dimensional system (41), with $b = 0.2$. Black curves represent the nullclines (see section 3.2) whose reciprocal intersections determine fixed points (black squares). Red segments indicate eigenvectors of the linearized system for real eigenvalues larger than one. Blue line segments represent eigenvectors of real eigenvalues in $(0, 1)$. Particularly important are blue lines of saddles, which represent local linear approximations of the boundary of the basins of attraction. The whole phase space is divided in four basins of attraction associated to the four stable points and their boundaries. These boundaries between these basins coincide with the stable manifolds of the saddles. The legend shows output values produced at every attractor, which, in this specific example, correspond with the internal state, i.e. the phase space coordinates of the attractors. Points on the plane have been coloured according to the attractor to which they apparently converged to after 100 steps.

in (38). For instance, a large δ will potentially activate all excitable connections between the attractors. However, in order to properly solve the flip-flop task, some of the connections need not to be active, namely, the diagonal connections between $\mathbf{p}_1$ and $\mathbf{p}_3$, and between $\mathbf{p}_2$ and $\mathbf{p}_4$. Due to symmetry, there are only three different excitability thresholds that an excitable connection can

have, that is, $\delta_{th}(\mathbf{p}_2, \mathbf{p}_1), \delta_{th}(\mathbf{p}_1, \mathbf{p}_2), \delta_{th}(\mathbf{p}_1, \mathbf{p}_3)$. In the particular configuration shown in Figure 12, it holds that $\delta_{th}(\mathbf{p}_2, \mathbf{p}_1) < \delta_{th}(\mathbf{p}_1, \mathbf{p}_2) < \delta_{th}(\mathbf{p}_1, \mathbf{p}_3)$. Therefore, the underlying ENA supporting the nonautonomous ESN dynamics is defined as $X_{\text{exc}}(\{\mathbf{p}_i\}_{i=1}^4, \delta)$, for every $\delta_{th}(\mathbf{p}_1, \mathbf{p}_2) < \delta < \delta_{th}(\mathbf{p}_1, \mathbf{p}_3)$. We note that $\delta_{th}(\mathbf{p}_1, \mathbf{p}_3) \approx \sqrt{2}$, regardless of $b \in [0,\, 0.47)$. We can reduce the size of the basin of attraction of $\mathbf{p}_2$ and $\mathbf{p}_4$ by increasing the value of b, which in turn reduces the excitability threshold of the corresponding connections, until at $b \approx 0.47$ a bifurcation occurs making them disappear[9]. However, the basins of attraction of the other two stable points, $\mathbf{p}_1$ and $\mathbf{p}_3$, become bigger ($\delta_{th}(\mathbf{p}_1, \mathbf{p}_2) \approx 2 - \delta_{th}(\mathbf{p}_2, \mathbf{p}_1)$), so increasing their excitability thresholds. Therefore, when the ESN state is close to $\mathbf{p}_1$ (or $\mathbf{p}_3$), the input needs to be very precise to throw back the state to the narrow basins of $\mathbf{p}_2$ and $\mathbf{p}_4$. Moreover, it must have a large amplitude compared to what is needed to escape from such narrow basins. Nevertheless, setting ω_{in} large enough and depending on b (until a value of $\omega_{in} = 6$, which is enough for every $0 < b < 0.47$), the system is able to reproduce the flip-flop dynamics without errors.

Unfortunately, it does not seem to be possible to design a two-dimensional ESN for the two-bit flip-flop where the excitability of each attractor can be tuned by changing the location of the nearby saddles. Nevertheless, as we will show in the next section, this becomes possible by including two additional dimensions in the model.

4.2.2 A $2k$-dimensional model for k-bit flip-flop tasks

In this section, we propose a $2k$-dimensional model able to solve k-bit flip-flop tasks. For the sake of clarity, but without loss of generality, we show the $k = 2$ bit case. The key insight is to model the switching dynamics between two fixed points in a two-dimensional space and then build a model formed by two independent systems with two-dimensional switching dynamics.

Unlike the previous example, here we want to tune the excitability of all connections. This can be obtained by imposing the condition in (34) and tuning the reservoir weights to change the position of the saddles. Therefore, we define

$$B := \begin{bmatrix} 1.1 & 4 \\ -s & 4 \end{bmatrix}, \tag{43}$$

[9] A bifurcation where both saddles collide with the corresponding stable points $\mathbf{p}_2$ and $\mathbf{p}_4$, simultaneously annihilating all six fixed points.

where s is a real parameter. Hence, for $0 \leq s < 2.15$, the autonomous dynamical system defined by $x[k] = \tanh\big(Bx[k-1]\big)$ has one repeller (the origin), two attractors, namely $\mathbf{p}_1 \approx (1,1)$, $\mathbf{p}_2 \approx (-1,-1)$, and two saddles. By increasing s within the $[0, 2.15)$ interval, we can make the saddles closer to the respective stable points, see Figure 13, hence decreasing the excitability thresholds of these attractors, until a saddle-node bifurcation occurs approximately at $s = 2.15$, annihilating attractors with saddles.

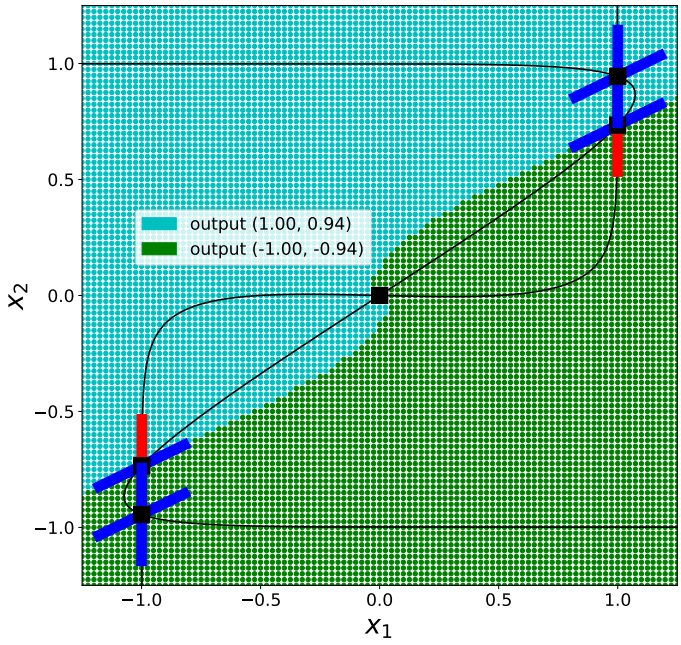


Figure 13: Basins of attraction, nullclines, fixed points and eigenvectors of the linearized two-dimensional autonomous system having (43) as reservoir matrix, with $s = 2$. Black curves represent the nullclines (see section 3.2) whose reciprocal intersections determine fixed points (black squares). Red segments indicate real eigenvectors associated with real eigenvalues larger than one. Blue line segments represent real eigenvectors of real eigenvalues in $(0, 1)$. Points of the plane have been coloured according with the attractor on which they converged after 100 steps.

Let us define the input matrix as follows:

$$W_{in} := \omega_{in} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix},$$

where ω_{in} is a positive real parameter. For instance, setting $s = 2$ and $\omega_{in} = 1$, the two-dimensional dynamical system defined by $x[k] = \tanh\big(Bx[k-1] + W_{in}u[k]\big)$ is able to accomplish the switching

mechanism between $\mathbf{p}_1$ and $\mathbf{p}_2$ according to inputs $(1,0)$, $(-1,0)$, and ignore other inputs, i.e., $(0,1)$, $(0,-1)$. Of course, replacing zeros in W_{in} with relatively small values (compared to ω_{in}) does not change the results.

The complete system able to solve the two-bit flip-flop dynamics can be obtained by defining the reservoir as the following four-dimensional block diagonal matrix,

$$M := \begin{bmatrix} B & O \\ O & B \end{bmatrix}, \tag{44}$$

where O denotes a 2×2 matrix with all zeros. Starting from a given initial condition, for every $0 \leq s < 2.15$, the state of the four-dimensional autonomous dynamical system $x[k] = \tanh\big(Mx[k-1]\big)$ converges to one of these four attractors $(\mathbf{p}_1, \mathbf{p}_1)$, $(\mathbf{p}_1, \mathbf{p}_2)$, $(\mathbf{p}_2, \mathbf{p}_1)$, $(\mathbf{p}_2, \mathbf{p}_2)$. In order to produce a two-dimensional output (11) suitable for the task at hand, we define $W_o := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$, which basically corresponds to a projection onto the first and third component, i.e., $(x_1, x_2, x_3, x_4) \longmapsto (x_1, x_3)$. Finally, we define the input matrix as

$$W_{in} := \omega_{in} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

We considered the case of a 4-dimensional system composed by two independent, two-dimensional systems. Nevertheless, the dynamics remains qualitatively the same even if the coupling between these two-dimensional systems is weak, i.e., if the zero matrices in (44) are replaced by matrices with relatively small (absolute) values. With those definitions in mind and, as before, $\boldsymbol{\epsilon} = 0$ and $\alpha = 1$, the ESN ruled by (13) and (11) correctly implements the flip-flop task with two bits.

As the dynamics of systems (x_1, x_2) and (x_3, x_4) are independent from each other, the set of fixed points of the overall dynamics turns out to be the Cartesian product of the set of fixed points of (x_1, x_2) and the fixed points of (x_3, x_4) system. This gives rise to a large number of fixed points: there are 4 stable points, 8 saddles with 1 unstable directions, 8 saddles with 2 unstable directions, 4 saddles with 3 unstable directions and 1 repeller. However, the set of fixed points where the nonautonomous flip-flop dynamics take place is formed by 4 stable fixed points and 8 saddles with only one unstable direction; see Section 4.4.1 for a detailed example. Every stable fixed point is

close to a pair of saddles and, due to symmetry, they are all at the same distance $\delta_{th}(\mathbf{p}_1, \mathbf{p}_2)$. This quantity defines the excitability thresholds of the connections needed in the flip-flop task, and therefore implicitly defines the ENA ruling the behaviour of the four-dimensional ESN.

4.3 Extracting the effective ENAs from a ESN trajectory

In this section, we describe the proposed algorithm to extract an ENA from an ESN trajectory. The proposed algorithm, which is schematically illustrated in Figure 14, takes the trained reservoir matrix M (12) and a trajectory of the nonautonomous ESN (13) as input and produces a weighted directed graph, representing the ENA describing the ESN behaviour, as output. The algorithm is composed of two main steps. In Section 4.3.1, we describe the procedure to find fixed points of the ESN dynamics (corresponding to the vertices of the graph). Successively, in Section 4.3.2, we show how to determine the excitable connections and related thresholds between stable fixed points (corresponding to the weighted directed edges).

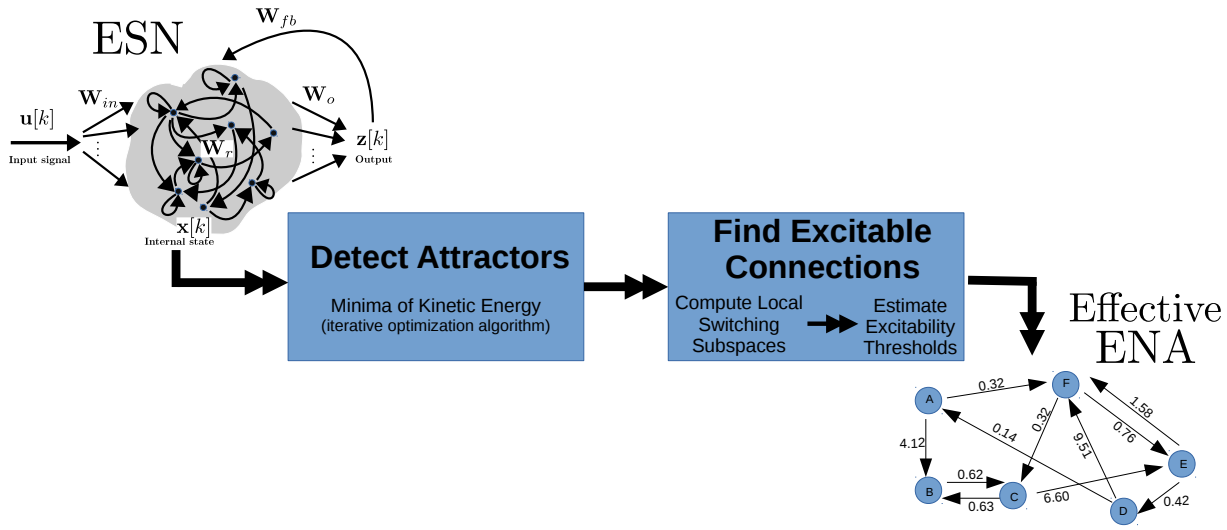


Figure 14: Illustration of the method applied in section 4.3 to extract ENAs from trajectories.

4.3.1 Finding fixed points of the dynamics

The optimisation algorithm we have used to find fixed points is based on Sussillo and Barak [107]; see [37] for an open-source Tensorflow toolbox for finding fixed points in arbitrary RNN architectures and [56] for an alternative method to identify fixed points. The key idea is to define a scalar function

whose minima correspond to fixed points of the ESN dynamics.

Definition 4.1 *We call* velocity field *of the autonomous system* (22) *the vector field* $Q : \mathbb{R}^{N_r} \longrightarrow \mathbb{R}^{N_r}$*, defined as*

$$Q(x) := F(x) - x, \tag{45}$$

with F *being the input-free and noise-free map of* (13)*.*

Therefore, the velocity field takes the following form:

$$Q(x) = \alpha\big[\tanh(Mx) - x\big], \tag{46}$$

where $Q(x[k])$ is the vector that needs to be added to the current state $x[k]$ to obtain the next one. In fact,

$$x[k+1] = F(x[k]) \iff x[k+1] - x[k] = F(x[k]) - x[k] \iff x[k+1] = x[k] + Q(x[k]). \tag{47}$$

Definition 4.2 *We define* kinetic energy *of the autonomous system* (22) *to be the following scalar function,*

$$q(x) := \frac{1}{2}\|Q(x)\|^2. \tag{48}$$

Note that fixed points $\mathbf{x}^* \in \mathbb{R}^{N_r}$ satisfy $Q(\mathbf{x}^*) = 0$ if and only if $q(\mathbf{x}^*) = 0$. Fixed points are hence identified as the global minima of (48). We use the quasi-Newton algorithm BFGS [85] to minimise (48). In order to speed-up the optimisation by several orders of magnitude, we explicitly provided the gradient of (48) to BFGS, which reads:

$$\nabla q(x_0) = J_Q(x_0)^T Q(x_0) \;=\; \alpha\big(D(x_0)M - I_{N_r}\big)^T Q(x_0), \tag{49}$$

where I_{N_r} is an $N_r \times N_r$ identity matrix, $J_Q(x_0)$ denotes the Jacobian matrix of the velocity field (46) and $D(x_0)$ is a diagonal matrix defined in (23), both evaluated on x_0.

The initial conditions for minimising (48) are determined by randomly sampling states from a trajectory of the nonautonomous ESN (13). The convergence of the BFGS algorithm depends on a tolerance. In fact, the algorithm may converge to similar solutions that, depending on chosen tolerance, are numerically different. As these solutions represent fixed points of the dynamics, we aggregate them in a meaningful way and return a non-redundant set of fixed points. For this purpose, as post-processing step, we run a clustering algorithm on the set of all solutions and return only cluster representatives; details provided in Appendix A.

4.3.2 Determining excitable connections between attractors

Once stable fixed points have been identified, we determine the excitable connections between them. As discussed before (and in more detail in section 3.2), the ESN training (12) induces bifurcations that generate new fixed points; and possibly also other attracting regions in phase space that, however, are not explicitly modelled in this work. The ESN is driven by control inputs that allow us to correctly perform switches between stable states. As a consequence, we first need to understand how such inputs affect the dynamics from a geometric point of view. This is done below by introducing the notion of local switching subspace (LSS), which is strictly related to the notion of local input response set, (39), introduced in Section 3.3. Then, we describe how we determine all excitable connections that are relevant for the task under consideration and compute their excitability thresholds. The method we propose is based on a grid of points lying in the LSS, accounting for the input action on the dynamics. By simulating the autonomous dynamics with initial conditions taken from such a grid, we are able to approximate input-driven excitability thresholds (40) and also to quantify how likely it is that the RNN uses such connections while solving the task.

Local switching subspaces

Definition 4.3 *Let us consider an ESN trajectory* (13)*,* $x[0], x[1], x[2], \ldots, x[k], \ldots$*, obtained with inputs* $u[1], u[2], \ldots, u[k+1], \ldots$*. Moreover, let us denote with* $\mathcal{K} := \{k_i\}_{i \in \mathbb{N}}$ *the set of indices for which* $u[k_i] \neq \mathbf{0}$*. We define* pulse difference vector *(PDV) a vector containing the difference between pre- and post-input states, namely*

$$x[k_i] - x[k_i - 1], \quad k_i \in \mathcal{K}. \tag{50}$$

The PDVs (50) convey relevant information about the action of inputs on ESN state and we exploit such information to determine the excitable connections and related thresholds.

Remark 4.1 *We remind the reader that the number of non-zero inputs is controlled by the parameter p of the exponential distribution (see section 4.1), which in turn determines the (average) number of PDVs available for the following analysis.*

In order to compute only those connections that are actually used by the ESN while solving the task, we need to focus on the action of inputs in the neighbourhood of each attractor. To this

end, we consider an Euclidean ball $B_r(\mathbf{p})$ of radius $r \geq 0$ centred on an attractor $\mathbf{p}$, and call *local* PDVs of $\mathbf{p}$ the finite set of PDVs that originate inside $B_r(\mathbf{p})$. Therefore, referring to (39), the local PDVs of $\mathbf{p}$ is a collection of vectors originating in $B_r(\mathbf{p})$ and ending in $\mathcal{G}(B_r(\mathbf{p}))$.

Definition 4.4 *Let $\mathbb{L}(\mathbf{p})$ be the vector space obtained by means of principal components analysis of the local PDVs* (50) *of the attractor $\mathbf{p}$, retaining only $l \ll N_r$ principal components. We define the* local switching subspace *(LSS) of $\mathbf{p}$, and we denote it as $\mathbf{p} + \mathbb{L}(\mathbf{p})$, the affine space composed by attaching the vector space $\mathbb{L}(\mathbf{p})$ over $\mathbf{p}$; see Figure 15, top panel.*

Estimation of excitability thresholds The idea is to sample the LSS of a stable point $\mathbf{p}_i$ and describe it as a grid of points $G(\mathbf{p}_i)$. Then, we consider those points on the grid as initial conditions for the autonomous system (22), which is then iterated for a sufficiently large number of steps to ensure convergence to nearby attractors. Finally, tracing the evolution of these initial conditions allows us to estimate excitability thresholds and other relevant quantities.

The proposed algorithm for finding excitability thresholds is graphically illustrated in Figure 15. The algorithm is based on a hypercube $H(\mathbf{p}_i)$ centred on attractor $\mathbf{p}_i$ that is contained within the LSS $\mathbf{p}_i + \mathbb{L}(\mathbf{p}_i)$. The length of the hypercube is such that $H(\mathbf{p}_i)$ contains the projection of $\mathcal{G}(B_r(\mathbf{p}_i))$ on $\mathbf{p}_i + \mathbb{L}(\mathbf{p}_i)$. In order to estimate excitability thresholds, we consider a mesh with a pre-defined density of points on the hypercube $H(\mathbf{p}_i)$, thus obtaining a grid of points, $G(\mathbf{p}_i) = \{\mathbf{g}_j^i\}$. To simplify the notation, we use a single index j to enumerate points of the grid. Through the ω-limit set (15) of the grid $G(\mathbf{p}_i)$, that is:

$$\omega_F(G(\mathbf{p}_i)) = \bigcup_j \omega_F(\mathbf{g}_j^i),$$

we can compute the following nonnegative integer $c(\mathbf{p}_i) := |\omega_F(G(\mathbf{p}_i))| - 1$, which counts the number of excitable connections originating from $\mathbf{p}_i$ which are allowed to be activated through input. Indeed, $\omega_F(G(\mathbf{p}_i))$ is composed of a set of stable fixed points, $\{\mathbf{p}_{i_0}, \mathbf{p}_{i_1}, \mathbf{p}_{i_2}, \ldots, \mathbf{p}_{i_{c(\mathbf{p}_i)}}\}$, determining the endpoints of the $c(\mathbf{p}_i)$ different excitable connections.

Remark 4.2 *Note that if the grid is sufficiently dense then the attractor itself is always included in such a set of fixed points. However we do not count this as an excitable connection. In what follows, we assume that the attractor is indexed by i_0, i.e., $\mathbf{p}_{i_0} = \mathbf{p}_i$.*

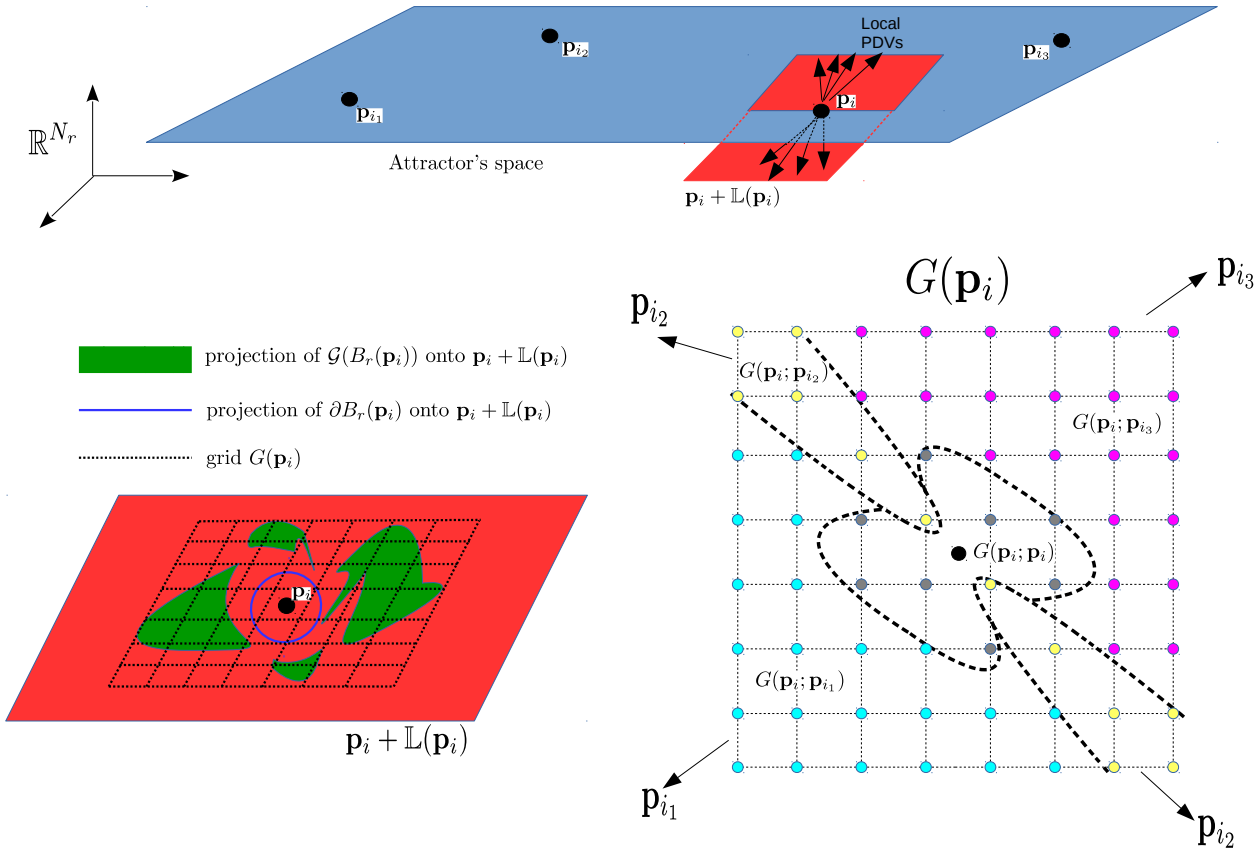


Figure 15: **Top:** in blue, the low-dimensional space containing the attractors; black arrows represent local PDVs (50) originating from the attractor $\mathbf{p}_i$ which in turn define the LSS of $\mathbf{p}_i$, sketched in red. **Bottom left:** representation of the local input response set $\mathcal{G}(B_r(\mathbf{p}_i))$, in green, and the grid $G(\mathbf{p}_i)$, dashed line, in the LSS of $\mathbf{p}_i$, in red. **Bottom right:** illustration of the partition of a two-dimensional grid $G(\mathbf{p}_i)$. Every colour represents a subset (51). Dashed black lines track the boundary of the basins corresponding to those attractors reachable from $\mathbf{p}_i$ through excitable connections, which can be enabled by inputs allowing exploration of the hypercube centred on $\mathbf{p}_i$.

With these definitions in mind, we are ready to compute thresholds of excitable connections. Let us denote with

$$\sigma_i : \{1, \dots, |G(\mathbf{p}_i)|\} \longrightarrow \{i_0, i_1, \dots, i_{c(\mathbf{p}_i)}\},$$

an indexing function such that $\mathbf{p}_{\sigma_i(j)} = \omega_F(\mathbf{g}^i_j)$. Through the preimage $\sigma_i^{-1}(\cdot)$, we obtain a partition

of points of the grid into the following subsets:

$$G(\mathbf{p}_i;\mathbf{p}_{i_t}) := \{\mathbf{g}_j^i \in G(\mathbf{p}_i) \ \mid \ j \in \sigma_i^{-1}(i_t)\},\ t = 0, 1, \ldots, c(\mathbf{p}_i). \tag{51}$$

The points of the grid belonging to the subset defined by (51) are all destined to converge to $\mathbf{p}_{i_t}$. Therefore, we estimate the input-driven excitability threshold (40) of the connection from $\mathbf{p}_i$ to $\mathbf{p}_{i_t}$ as follows:

$$\widetilde{\delta}_{\text{inp}}(\mathbf{p}_i, \mathbf{p}_{i_t}) = \min\left\{\|\mathbf{g}_j^i - \mathbf{p}_i\| \ \ : \ \ \mathbf{g}_j^i \in G(\mathbf{p}_i;\mathbf{p}_{i_t})\right\}, \quad t = 1, \ldots, c(\mathbf{p}_i). \tag{52}$$

The excitability thresholds (52) represent geometric properties of the attractors and related basins in phase space learned through training. In order to determine the effective excitability (accounting for inputs) of each connection outgoing from $\mathbf{p}_i$, we exploit the local topology of the LSS of $\mathbf{p}_i$ by means of the grid partition (51) induced by $\sigma_i(\cdot)$. To this end, we define the ratio of initial conditions taken from the grid that converged to attractor $\mathbf{p}_{i_t}$ as follows:

$$\nu_{i,i_t} := \frac{|G(\mathbf{p}_i;\mathbf{p}_{i_t})|}{|G(\mathbf{p}_i)| - |G(\mathbf{p}_i;\mathbf{p}_i)|} \ \in [0, 1]. \tag{53}$$

In the limit of infinite number of points in the grid, the quantity (53) converges to the ratio of volumes between the portion of the hypercube belonging to the basin of $\mathbf{p}_{i_t}$ and the portion of the hypercube that does not belong to the basin of $\mathbf{p}_i$. Therefore, dense grids give ratios (53) providing accurate information about how the LSS is distributed between basins of attraction of stable fixed points. Finally, merging both (52) and (53) into a single expression, we define *effective excitability* of the connection from $\mathbf{p}_i$ to $\mathbf{p}_{i_t}$ as follows:

$$\beta_{i,i_t} := \frac{\nu_{i,i_t}}{\widetilde{\delta}_{\text{inp}}(\mathbf{p}_i, \mathbf{p}_{i_t})}. \tag{54}$$

Note that this quantity takes into account both the distance between the attractor $\mathbf{p}_i$ and the basin's boundary, and the volume of the basin itself. A low value for β_{i,i_t} indicates that it is difficult to activate the connection from $\mathbf{p}_i$ to $\mathbf{p}_{i_t}$ during the task. This may be due (i) to the small volume occupied by the basin of the attractor $\mathbf{p}_{i_t}$ in the LSS of $\mathbf{p}_i$ or (ii) to a very high excitability threshold associated with such a connection. On the other hand, $\beta_{i,i_t} \gg 1$ necessarily implies that such a connection has a low excitability threshold, since $\nu_{i,i_t} \in [0, 1]$. As a consequence, the distance between the basin of attraction of $\mathbf{p}_{i_t}$ and $\mathbf{p}_i$ is small, thus the connection can be easily activated during the task.

Remarks on computational complexity. There are three parameters controlling the complexity and, accordingly, the accuracy of the search in the grid: the dimension ζ_1 of the hypercube, the length ζ_2 of the hypercube's edge, and the number of points ζ_3 on each edge determining the density of the grid. ζ_3 and ζ_2 have a linear and polynomial impact on the computational complexity of the algorithm, respectively. However, ζ_1 is more critical as it increases exponentially the number of points in the grid and, accordingly, the overall complexity. In the simulations we always set $\zeta_1 = l$, that is, the dimension of the hypercube is equal to the dimension of the LSS which is low in our case. More generally, the dimension of LSSs depends on the complexity of the inputs and their impact on the dynamics, and this will need to be assessed on a case-by-case basis.

4.4 Simulations

In this section we discuss the results of our simulations and relate this to our theoretical framework. In Section 4.4.1, in order to evaluate the correctness of the algorithm proposed in Section 4.3, we apply it to the manually-designed, low-dimensional ESN maps discussed in Section 4.2.1 and Section 4.2.2. Section 4.4.2 applies our method to high-dimensional trained ESNs. We show that, even though the ESN is high-dimensional, the dynamics generated by the trained reservoir (12) is effectively low-dimensional. We also show the usefulness of ENAs for giving a mechanistic interpretation of prediction errors occurring during task execution. Finally, in Section 4.4.3, we show how ENA models can be used to assess the robustness to noise of trained ESNs. For all simulations, we use $p = 0.1$ as parameter of the exponential distribution governing the occurrence of input pulses and set the tolerance of the kinetic energy (48) for detecting minima to 10^{-6}.

4.4.1 Evaluation on manually designed low-dimensional ESNs

For the grid search algorithm, we used $\zeta_1 = 2, \zeta_2 = 4$, and $\zeta_3 = 223$.

Minimal-dimension model The LSS determined as described in Section 4.3.2 corresponds to the whole 2D phase space. As a consequence, excitability thresholds (37) match the corresponding input-dependent counterparts (40). For each attractor, the Euclidean distances from the two closest saddles are consistent with the estimation of excitability thresholds provided by our method. As discussed in Section 4.2.1 and graphically represented in Figure 16 bottom-centre panel, we find three excitability thresholds in the computed ENA: $0.49, 1.39$, and 1.42. In Figure 16, bottom-right

panel, we show that undesired connections (i.e., connections corresponding to state transitions that are not defined in the flip-flop task) between fixed points $(0.97, -0.97)$ and $(-0.97, 0.97)$ have very low effective excitability values (54), indicating that it is unlikely to use such connections during the task execution. The low effective excitability is due to the small volume ratio (53) of the basins associated with the two attractors. On the other hand, larger thresholds characterise the undesired connections from $(-1, -1)$ to $(1, 1)$, reflecting a lack of symmetry between the basin volumes of $(1, 1), (-1, -1)$ and $(0.97, -0.97), (-0.97, 0.97)$.

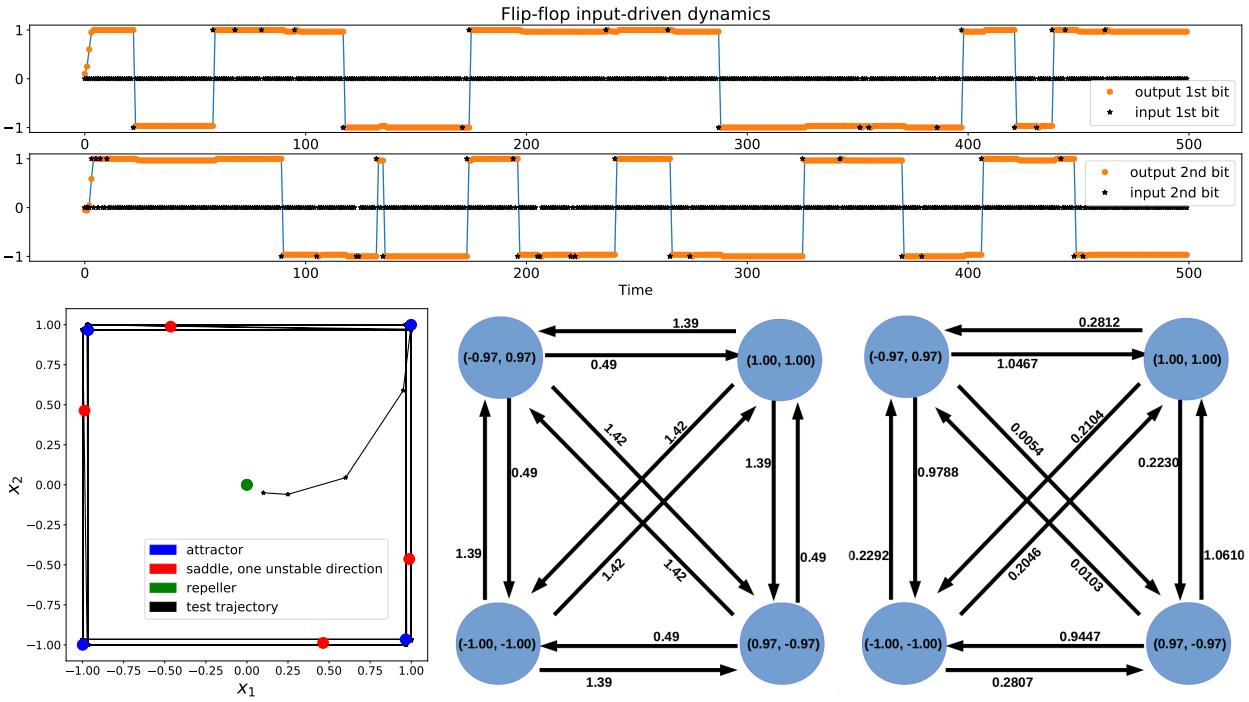


Figure 16: **Top:** output produced by the two-dimensional ESN defined in Section 4.2.1. **Bottom left:** fixed points found by the optimisation algorithm with 100 initial conditions. Length of the test trajectory was 1000 steps. **Bottom centre:** extracted ENA with edges weighted by excitability thresholds (52). **Bottom right:** extracted ENA with edges weighted according to (54). Node labels represent output values (11) produced on the attractors.

Four-dimensional model The first two principal components obtained via principal component analysis are related to all identified fixed points and produce a cumulative variance ratio larger than 0.96; Figure 17 shows a trajectory of the map described in Section 4.2.2 and related fixed points projected on the plane spanned by the two principal components. For every attractor, the computed

LSS is a two-dimensional plane; this is expected, since it is actually the plane depicted in Figure 13. Furthermore, we note that none of these planes is aligned with the one where the attractors lie, stressing the importance of defining reference frameworks local to each attractor that take the action of inputs into account. Figure 17, middle-right panel, shows how the input moves the states out of the plane, using additional dimensions for the switches. The computed ENA, weighted with excitability thresholds (52) and effective excitability (54), is shown in Figure 17, bottom-left and bottom-right panels, respectively. Symmetries of the dynamical system are clearly present in the resulting directed graphs. All desired connections are characterised by an excitability threshold of $\simeq 0.83$, while undesired ones have higher thresholds equal to $\simeq 1.19$. We note that the presence of undesired connections does not imply that the ESN actually uses such connections during the task execution. To this end, the effective excitability thresholds (54), shown in the bottom-right panel of Figure 17, provide us with a more realistic picture of the behaviour under the action of inputs. Note that the effective excitability thresholds are very low for the undesired connections, implying that the LSS used by inputs is mostly occupied by basins corresponding to attractors adjacent to the end-point of desired connections.

4.4.2 Application of the proposed method to high-dimensional trained ESNs

We now consider implementation using ESNs (13) with a reservoir composed of 500 neurons. Experimenting with ESNs with different reservoir sizes confirm that one can get ESNs can be successfully trained independent of the precise number of neurons, as long as there are enough. We choose 500 neurons for hardware and computing time considerations. For the grid search algorithm, we use $\zeta_1 = 3, \zeta_2 = 18$,and $\zeta_3 = 12$. The state-update (13) is configured without leakage, $\alpha = 1$ and standard deviation of noise $\boldsymbol{\epsilon}$ is set to 10^{-4} during training. The entries of matrices W_{in}, W_{fb}, and W_r are i.i.d. drawn from a uniform distribution in $[-1, 1]$; the sparseness of W_r is 95%. Moreover, the reservoir matrix was rescaled to obtain a spectral radius equal to 0.9. Finally, the training set length is always 50000 time-steps.

Low-dimensional dynamics Figure 18, top panel, shows the output produced by an ESN achieving high prediction performance. The extracted ENA, shown in the bottom-right panel, reveals that undesired connections have excitability thresholds (40) significantly higher than those of desired connections. This means that, in the LSS of every stable point, basins corresponding

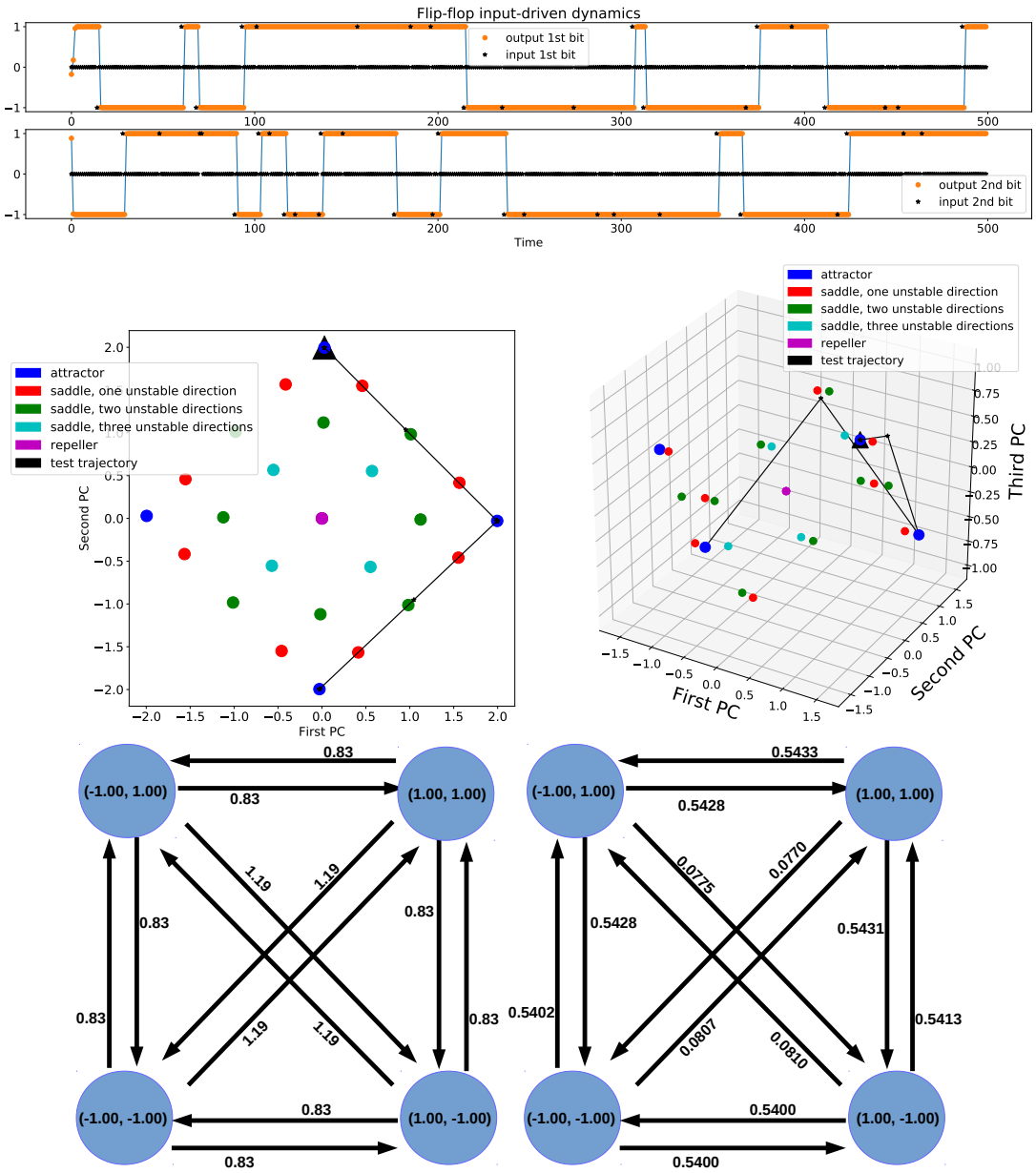


Figure 17: **Top:** output of the four-dimensional ESN defined in Section 4.2.1. **Centre left:** fixed points found by the optimisation algorithm with 200 initial conditions, depicted on the space spanned by the first two principal components. In black, it is shown a trajectory starting on the attractor (marked by a black triangle); to improve readability, the trajectory is limited to two switches only. **Centre right:** fixed points and trajectory depicted in the centre-left picture are embedded in the space spanned by the three principal components. **Bottom left:** ENA with edge weights representing excitability thresholds (52). **Bottom right:** ENA with edge weights representing the effective excitability of connections (54). Node labels represent output values produced by the ESN on the attractors.

to attractors adjacent to the end-point of undesired connections stand relatively far away from the stable point compared to basins of attractors adjacent to the end-point of desired connections. The fixed points of the dynamics lie in a two-dimensional subspace of the 500-dimensional phase space (the cumulative variance ratio is close 1). It is observed that trajectory spends most of the time close to such a plane. Hence, based on a principal component analysis of the trajectory, we can claim that the dynamics is two-dimensional. Nevertheless, during the task execution, the trajectory is occasionally driven away from such a 2D plane by the inputs in order to achieve the switches between attractors, and this feature is crucial to understand how the trained neural network behaves while solving the task. The fast transient dynamics outside such a 2D plane are ruled by the excitable connections between attractors. As can be observed by the top plot of Figure 18 those transitions look very sharp in the output space. It is likely that temporarily escaping from the 2D plane where the attractors reside allows the network to avoid to assume intermediate output real values between -1 and $+1$. The cumulative variance ratio for the identification of the LSS of every attractor, as described in Section 4.3.2, revealed that the switching between stable fixed points takes place in a three-dimensional subspace of the phase space, highlighting that the overall dynamics of ESNs is effectively low-dimensional after training. It is worth stressing that such LSSs are usually not aligned with the standard coordinate system of the original phase space and the subspaces where attractors lie, suggesting that inputs operate in phase space regions that are disjoint with respect to the low-dimensional linear subspace of the attractors; this is consistent with results reported in [107].

Computation accuracy and spurious attractors Various measures of accuracy exist for quantifying the performance on prediction tasks. For instance, the mean-squared-error (MSE) is typically adopted in tasks involving continuous targets. The MSE is a real-valued scalar that informs us about how close the computed output is to the target one. However, it is not possible to infer additional insights by looking only at the MSE; for instance, it is not possible to provide a mechanistic description of errors in the computation, i.e., why and how they occur. Here, we show how the effective ENA extracted from the ESN trajectory can be used to describe how the computation takes place in phase space and, in particular, how to diagnose the nature of prediction errors.

The top panel in Figure 19 shows the output produced by an ESN achieving low MSE of the

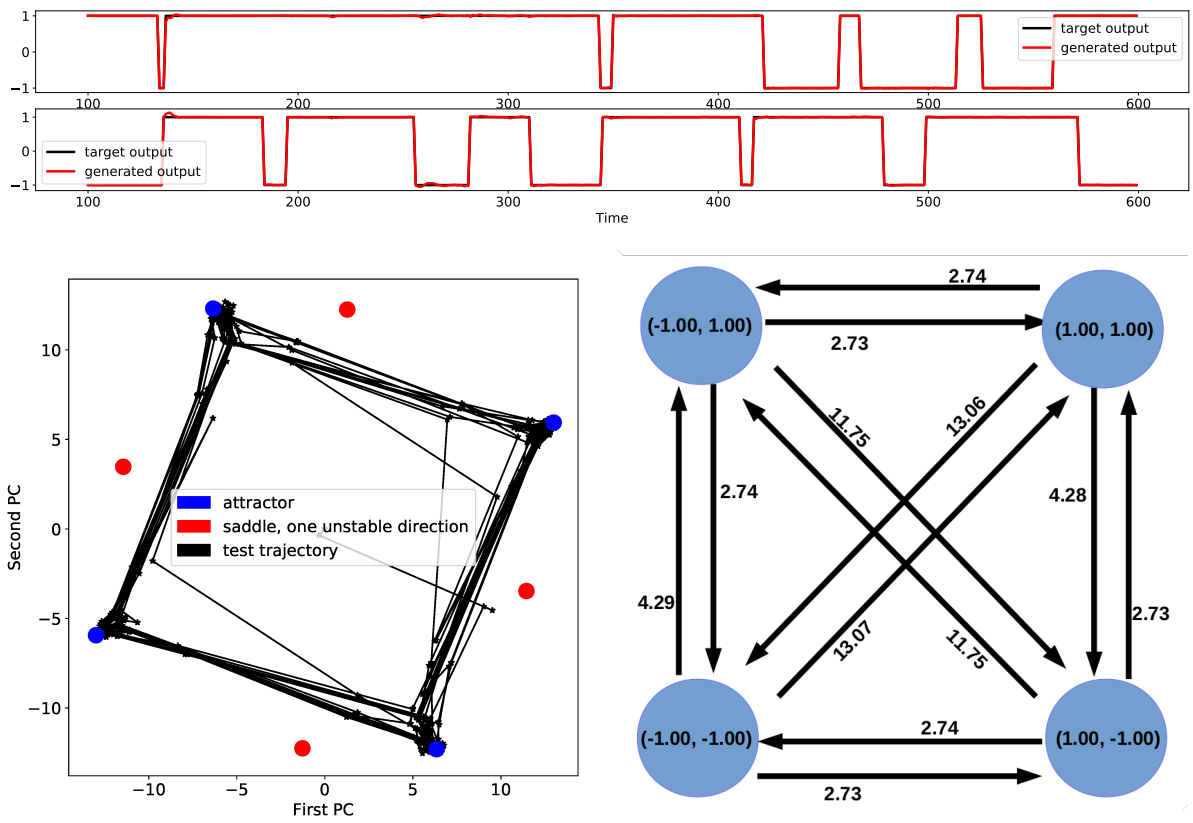


Figure 18: **Top:** output produced by a trained ESN and target output. **Bottom left:** fixed point found by means of the optimisation algorithm with 500 initial conditions. The plane in the picture represents the space spanned by the first two principal components determined over all 500 solutions returned by the optimisation algorithm. **Bottom right:** extracted ENA with edge weights representing excitability thresholds (52). Node labels represent output values produced by the ESN (11) on the attractors.

order of 10^{-3}. The small errors observable around time-step 13200 can be explained by looking at the ENA model depicted in the bottom panels of Figure 19, which is represented with excitability thresholds (52), left panel, and with effective excitability values (54), right panel. The directed graphs (whose topology is clearly identical) reveal the presence of two extra stable fixed points in the ESN phase space, which are generated during training. Nodes of these graphs are coloured according to output values produced by the ESN. The activation of the related excitable connections brings the ESN to operate in the proximity of such superfluous states, producing inaccurate output values and hence explaining the origin of such errors. Notably, the directed edge – in the graph

on the right-hand side – connecting the cyan with the yellow node has a relatively high value of effective excitability. This means that, in the LSS of the related attractor (cyan), whose output is $(-1.03, -1.07)$, it is relatively easy to transition to the basin of the other attractor (yellow). This, in turn, produces an output value equal to $(-0.84, -0.40)$, which is significantly different from the target output, i.e., $(-1, -1)$. The prediction error, visible to the naked eye in the top panel around time-step 13200, is indeed due to the activation of that excitable connection.

Both of these spurious attractors act as a sort of surrogates for the correct ones (i.e., those producing lower prediction errors). In fact, once the internal state switches to a spurious attractor, the ESN still behaves consistently. More precisely, spurious attractors are associated with higher effective excitability values on connections ending up in the correct attractors – see the bottom-right panel of Figure 19. The existence of these spurious attractors and the unravelling of their roles in the ESN computation could not easily be inferred by looking only at the MSE or plotting the outputs produced by the ESN. This demonstrates the importance of ENA models for describing the ESN behaviour.

4.4.3 Noise tolerance and effective excitability of ENAs

Here, we aim to provide a further example of the importance of ENAs for characterising the computation of ESNs. In particular, we ask the following question: given a set of ESNs trained on the same task and achieving the same performance during training, which one will be more robust to noise during test? Typical performance measures, such as MSE, cannot be used to answer such a question and provide useful insights. We show that an ENA model of the computation, weighted with effective excitability values (54), allows us to assess robustness to noise of ESNs.

As a perturbation, we consider the usual white Gaussian noise term $\boldsymbol{\epsilon}$ in the state-update (13), corresponding to perturbations directly applied to all neuron pre-activation values. We stress that such perturbations are applied only during the test phase; training is implemented as described in the previous section. By increasing the noise standard deviation, the ESN trajectory gets increasingly perturbed neglecting the possibility to reach the proximity of attractors, hence affecting the accuracy of the resulting output values.

We note that: (i) ESNs yielding ENAs with higher effective excitability values are less robust to noise perturbations than ESNs giving rise to ENAs with low effective excitability values (see [6]),

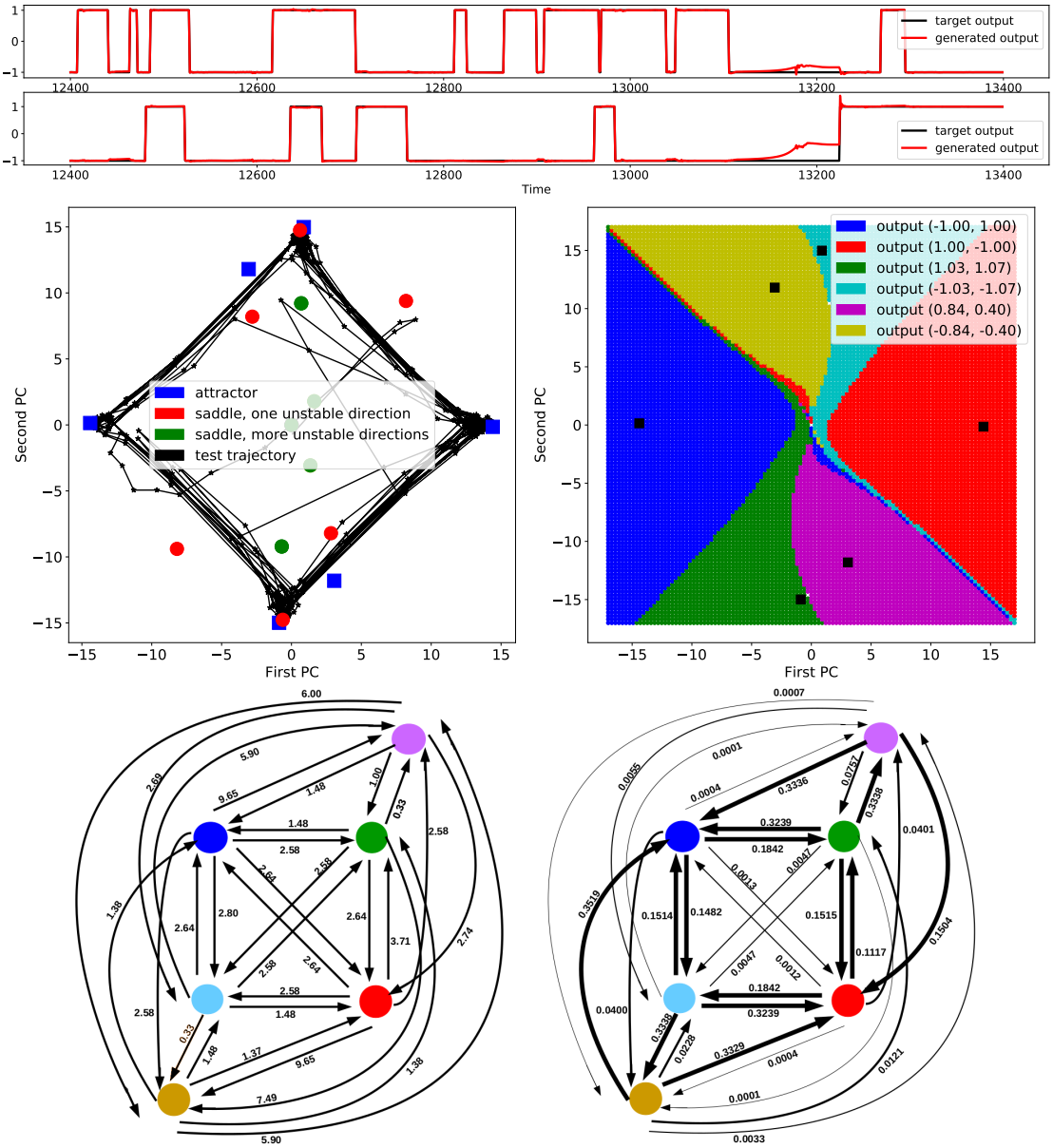


Figure 19: **Top:** output of the trained ESN showing an incorrect switch around step 13200. **Center left:** fixed points found by the optimisation algorithm plotted in their two-dimensional subspace together with the trajectory. **Center right:** attractor plane divided by basins of attraction of every stable fixed point. The figure is drawn assuming a transient of 300 time-steps. Every basin is coloured depending on the attractor where the ESN converges to without applying any input. **Bottom left:** extracted ENA weighted with estimation of input-driven excitability thresholds (52). **Bottom right:** extracted ENA weighted with effective excitability value (54). Nodes are coloured according to the colours used in the center-right figure. The thickness of the arrows is scaled according to the effective excitability value.

and (ii) ESNs producing ENAs with balanced edge weights on desired outgoing connections are more robust to noise perturbations than ENAs with unbalanced outgoing weights. Regarding (i), high excitability values imply the existence of connections with low excitability thresholds. That is, the basin of the attractor corresponding to the end-point of such a connection is very close to the attractor associated with the origin of the connection, resulting in unnecessary switches that induce errors. Concerning (ii), for a given attractor, unbalanced outgoing connection weights could be a symptom of unbalanced distribution of space between basins of attraction in the LSS or the existence of some basins significantly closer to the attractor than other basins. In the latter case, a reasoning similar to (i) applies. On the other hand, if there is indeed an asymmetric distribution of volumes between basins then basins of attractors corresponding to connections possessing large volume ratios will fill most of the LSS, leaving little space for basins of those attractors related to connections with small volume ratios. Therefore, if the ESN state is close to an attractor with unbalanced outgoing connection weights, but with similar excitability thresholds, then under noise perturbations it is more likely to switch towards an attractor reachable through a connection with high effective excitability even when such a connection should not be activated. For these reasons, an ENA, where each node is characterised by balanced weights on outgoing connections and very low effective excitability values on undesired connections, provides a prototypical example of ESNs whose behaviour is robust to noise.

To quantify the robustness of this behaviour to noise, we selected two ESNs that solve the two-bit flip-flop task with very high and comparable accuracy – MSE during training is $\simeq 10^{-4}$. We test them on the same input series composed of 10^5 time-steps and recorded their MSEs by considering different noise instances with increasing standard deviation. Directed graphs representing the extracted ENAs are shown in Figure 20. Results for increasing noise standard deviation are divided in two columns: on the left column, we report results obtained by the ESN that is least tolerant to noise. The directed graph on the left-hand side of Figure 20 shows the presence of two undesired connections. The edge weights of desired connections are not balanced, especially those outgoing from attractors with output values $(1, 1)$ and $(-1, -1)$. Conversely, the directed graph on the right-hand side does not contain undesired connections and possess balanced outgoing weights. The absence of undesired connections indicates that, in the LSS of every attractor, the basins of the attractors corresponding to undesired connections do not exist or, alternatively, they are very

small and hence not detectable with the grid density used in our simulations; therefore, they are not relevant for describing the ESN behaviour according with the numerical precision we considered in our simulations. Finally, we highlight that a performance breakdown for the ESN on the left-hand side panel is observed starting from noise standard deviation of 8×10^{-2}. On the other hand, the ESN on the right-hand side is significantly more robust to noise, denoting a performance breakdown for noise standard deviation of 1.4×10^{-1}.

4.5 Beyond autonomous dynamical systems

Despite the exciting results we found in this chapter, the methodology we proposed of extracting the underlying excitable network attractor from a RNN is affected by some drawbacks. Probably the most severe one is the impossibility to directly apply the notion of excitable network attractor to RNN driven by properly time-varying input signals. In fact, when an external input is allowed to effectively change the equations ruling the dynamics at each time step then familiar concepts like fixed points do not make sense anymore, and attractors need to be carefully defined. As a matter of fact, the input stream injected into the RNN is a time-varying complex signal in a countless number of tasks in machine learning. As a consequence, an effective excitable network attractor might not be extracted from a trajectory as described in this chapter. It follows that a comprehensive theory of RNN dynamics cannot be fitted into the framework of autonomous dynamical systems theory. In the following chapter we try to overcome this challenge, and we investigate RNN dynamics from the perspective of nonautonomous dynamical systems theory.

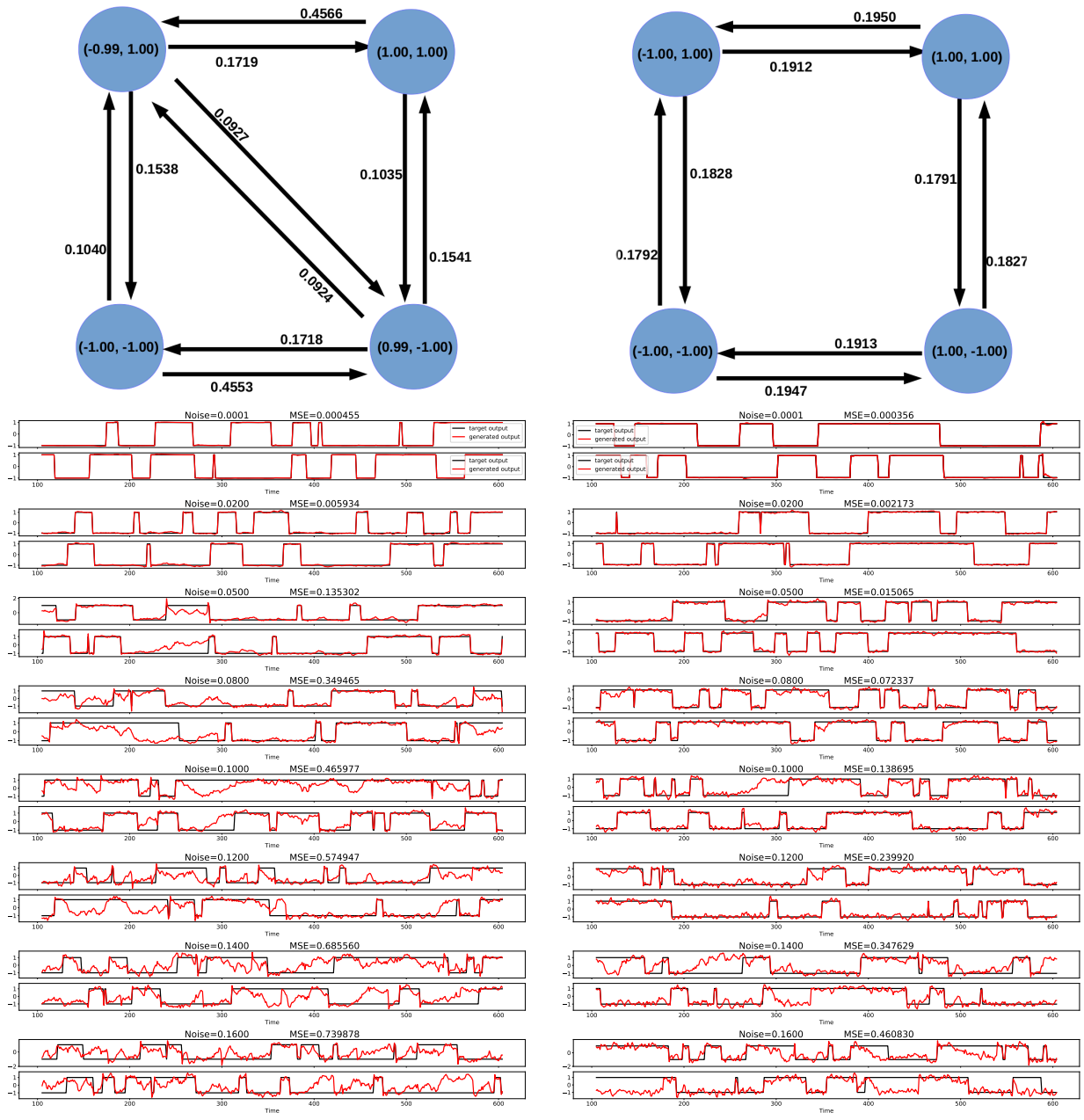


Figure 20: **Left:** results corresponding to a trained ESN giving rise to an ENA with undesired connections and unbalanced outgoing weights (54). **Right:** results obtained with a trained ESN giving rise to an ENA with balanced outgoing weights for all excitable connections. From top to bottom, in both columns, ESN outputs and related MSEs obtained with increasing noise standard deviation: 10^{-4}, $2 \cdot 10^{-2}$, $5 \cdot 10^{-2}$, $8 \cdot 10^{-2}$, 10^{-1}, $1.2 \cdot 10^{-1}$, $1.4 \cdot 10^{-1}$, and $1.6 \cdot 10^{-1}$.

5 RNNs as multi-stable input-driven dynamical systems

> Science is built up with facts, as a house is with stones. But a collection of facts is no more a science than a heap of stones is a house.
>
> Henri Poincaré

Recurrent neural networks (RNNs) are input-driven (i.e. nonautonomous) dynamical systems [75] whose behaviour depends both on model parameters and on inputs to the system. In order to describe responses of a (trained) RNN to the full range of possible inputs, it is necessary to go beyond the theory of autonomous (input-free) dynamical systems and consider more general nonautonomous dynamical systems [60], where the equations ruling the dynamics change over time. The theory of nonautonomous dynamical systems is much less developed than that of autonomous systems, and notions such as convergence (and hence attractors) need to be carefully defined [61].

Starting with the work of Jaeger [51], several authors have proposed that a successfully trained RNN should have the so-called *echo state property* (ESP) [120]. The idea of "echo state" gave rise to a training paradigm of RNNs called *reservoir computing* [71] and a class of RNNs known as *echo state networks* (ESNs) [52]. An ESN is an RNN that is relatively easy to train, since optimisation is restricted to the output layer and recurrent connections are left untouched after initialisation.

If an RNN possesses the ESP this means that, for a given input sequence, it will asymptotically produce the same sequence of states and will "forget" any internal state, ending up following a unique (though possibly very complex) trajectory in response to that input [74]. This trajectory represents the solution of the specific problem encoded in the input sequence that is fed to the network and so the system can be seen as acting as a *filter* that transforms the input sequence into a unique sequence of output [40]. The presence of the ESP has been historically associated with reliable RNN behaviour. Accordingly, the loss of ESP has been directly associated with the loss of reliable behaviour in RNNs, implying that correct computation is not possible without ESP.

In this chapter, we analyse the ESP through the lens of nonautonomous dynamical system theory and introduce a generalisation of the ESP, which has been published in [21], that accounts only for local behaviour in phase space. To this end, we introduce an "echo index" to characterise the number of simultaneously stable responses of a driven RNN. We achieve this by defining a

simple, yet non-trivial class of attracting solutions for input-driven systems that we call *uniformly attracting entire solutions* and count how many of these solutions coexist in phase space under the action of an input sequence. We show that echo index 1 and the classical notion of ESP do not always coincide. More precisely, driven RNNs having echo index 1 are allowed to produce reliable behaviour, in the ESP spirit, only in a phase space subset thus eliminating those regions producing unreliable behaviour but having zero probability of occurring.

The presence of multiple attractive solutions may be useful when the RNN is used to perform context-dependent computations, see [109, 112]. For example, consider the addition modulo N, i.e. the addition in a modular arithmetic. Given an initial integer value n_0 and an input sequence of integers $n_1, n_2, \ldots$, assuming values in $\{0, 1, \ldots, N-1\}$, the neural network needs to compute online the sum $N_k := \sum_{i=0}^{k} n_i \pmod N$ as time k runs. Therefore, in this case the system has to respond in N different ways depending on the initial state.

The remainder of this chapter is structured as follows. Background material is introduced in Section 5.1. In Section 5.2 we discuss about pullback attractors of RNNs. Remarkably, we rigorously prove in Proposition 5.5 that the *natural association* of [74] corresponds to the system's pullback attractor [60]. In Section 5.3, we describe our main contribution: a generalisation of the ESP for driven RNNs. Moreover, we point out some limitations of pullback convergence for RNNs, and in particular in Section 5.3.2 we link pullback convergence with uniform forward convergence in RNNs. Our main theoretical results are discussed in Section 5.4: (1) in Theorem 5.9 we provide a sufficient condition for the existence and uniqueness of a uniformly attracting entire solution in a phase space subset; (2) in Theorem 5.11, we discuss a mechanism giving rise to the occurrence of multistability in systems driven by low amplitude inputs; (3) in Proposition 5.14 we prove that forcing RNNs with large-amplitude inputs will induce echo index 1 (and the ESP); finally, (4) in Section 5.4.4 we discuss stability of the echo index w.r.t. perturbations on the input sequences and show, in Theorem 5.15, how this depends on the metric imposed on the space of input sequences. In Section 5.5 we report some numerical experiments. Section 5.5.1 provides an example of multi-stable RNN dynamics possessing echo index 2. In Section 5.5.2, we show how the echo index might depend, fixing the RNN model parameters, on the particular input driving the dynamics, illustrating a bifurcation from echo index 1 to echo index 2. This offers new insights on the fact that changes in behaviour may also occur for reasons that are not related to changes in model parameters (i.e.

via training). In Section 5.5.3, we train an RNN to solve the context-dependent task illustrated in [46, Figure 5] and give an interpretation of the related multistable, nonautonomous RNN dynamics based on the modeling framework developed here.

Notation: in this chapter we dedicate bold letters for sequences (of inputs or states), and we denote with superscript stars * fixed points.

5.1 Nonautonomous dynamics of recurrent neural networks

In this section, we investigate RNN dynamics using tools from nonautonomous dynamical systems theory. In Section 5.1.1, we highlight the input-driven nature of RNN dynamics and, successively in Section 5.1.2, we introduce the skew product formalism for nonautonomous dynamical systems [60]. Finally, in Section 5.1.3, we introduce the fundamental notion of *pullback attractor*.

5.1.1 Input-driven dynamical systems

In the remainder, we will assume to deal with an already trained RNN and thus do not consider the effects of training on the dynamics. The equations (8)-(9) ruling the behaviour of a (trained) RNN can be seen as a special case of an input-driven dynamical system of the form:

$$x[k+1] = G(u[k+1], x[k]), \tag{55}$$

where the map $G : U \times X \to X$ is defined as

$$G(u, x) = (1-\alpha)x + \alpha\phi(W_r x + W_{in} u + W_{fb}\psi(x)). \tag{56}$$

The action of inputs $u[k]$ driving the dynamics of $x[k]$ gives an explicit time-dependence of (8) and means this system is nonautonomous. A given input sequence $\mathbf{u} = \{u[k]\}_{k\in\mathbb{Z}}$ induces a sequence of maps $\{f_k\}_{k\in\mathbb{Z}}$, where $f_k(\cdot) := G(u[k+1], \cdot) : X \to X$ is the map ruling the update of the RNN state at time k, i.e. $x[k+1] = f_k(x[k])$.

Note that, when considering the map (56), as long as the neuronal activation function ϕ and the readout function ψ are both regular of class C^1, then the map G will be too. Moreover, for all $u \in U$, the map $G(u, \cdot) : X \to X$ is a local diffeomorphism whenever matrix $M(x)$ (19) is invertible. In addition, if ϕ is bounded with image $(-L, L)$ then the state space X can be assumed to be the compact hypercube $[-L, L]^{N_r} := \{(x_1, \ldots, x_{N_r}) \in \mathbb{R}^{N_r} \mid x_i \in [-L, L],\ i = 1, \ldots, N_r\}$, e.g. if $\phi = \tanh$ then $L = 1$, see Proposition 5.2.

We make the following standing assumptions:

Assumption 5.1 *(i) G is continuously differentiable in all arguments, i.e $G \in C^1(U \times X, X)$;*

(ii) for all $u \in U$, the map $G(u, \cdot) : X \to X$ is a local diffeomorphism onto its image;

(iii) $U \subset \mathbb{R}^{N_i}$ is compact and $X \subset \mathbb{R}^{N_r}$ is usually the compact closure of a N_r-dimensional Cartesian product of real intervals.

Remark 5.1 *(ii) implies that the preimage of any zero measure set is also zero measure, hence given any input sequence $\{u[k]\}_{k\in\mathbb{Z}}$ assuming values in U, for any $Z \subset X$ with $\lambda(Z) = 0$, then $\lambda(f_k^{-1}(Z)) = 0$ for all $k \in \mathbb{Z}$, where λ denotes Lebesgue measure on $X \subset \mathbb{R}^{N_r}$. This is weaker than assuming that $G(u, \cdot)$ is invertible for fixed u, but it means that phase space volume cannot "suddenly collapse".*

5.1.2 The cocycle formalism

There are two ways to describe nonautonomous systems [60]: the *process* and the *skew product* (also called *cocycle*) formalism. In this thesis, we use the cocycle formalism that is convenient when describing the input evolution as a shift in sequence space.

Let (X, d_X) and (U, d_U) be compact metric spaces. Time evolution will be parametrised through the ring of integers $\mathbb{Z}$ or a subset of it. We write $\mathbb{Z}^+ := \{k \in \mathbb{Z} \,:\, k \geq 1\}$ and $\mathbb{Z}_0^- := \{k \in \mathbb{Z} \,:\, k \leq 0\}$. Let $\mathbb{T}$ be one of the sets $\mathbb{Z}$, $\mathbb{Z}^+$, or $\mathbb{Z}_0^-$, we consider the set $U^{\mathbb{T}} := \{\mathbf{u} = \{u[k]\}_{k\in\mathbb{T}} \,:\, u[k] \in U,\ \forall k \in \mathbb{T}\}$ of all input sequences assuming values in set U, and we will denote $\mathcal{U} = U^{\mathbb{Z}}$, $\mathcal{U}^+ = U^{\mathbb{Z}_0^+}$ and $\mathcal{U}^- = U^{\mathbb{Z}_0^-}$. Moreover, given $\mathbf{u} \in \mathcal{U}$, we will denote with $\mathbf{u}^+$ and $\mathbf{u}^-$ the projection of $\mathbf{u}$ to $\mathcal{U}^+$, and $\mathcal{U}^-$, respectively. The set $\mathcal{U}$ is usually equipped with the product topology induced by the metric

$$d_{\text{prod}}(\mathbf{u}, \mathbf{v}) := \sum_{k\in\mathbb{Z}} \frac{d_U(u[k], v[k])}{2^{|k|}}, \tag{57}$$

which renders $(\mathcal{U}, d_{\text{prod}})$ a compact metric space. Another metric suitable for our purposes is the uniform metric, defined as:

$$d_{\text{unif}}(\mathbf{u}, \mathbf{v}) := \sup_{k\in\mathbb{Z}} d_U(u[k], v[k]). \tag{58}$$

From the fact that U is compact, it follows that $(\mathcal{U}, d_{\text{unif}})$ is a compact metric space.

The dynamics on this space of input sequences is described by means of the *shift operator*, which is a map $\sigma : \mathcal{U} \to \mathcal{U}$ defined as follows:

$$\mathbf{u} := \{u[k]\}_{k\in\mathbb{Z}} \longmapsto \sigma(\mathbf{u}) := \{u[k+1]\}_{k\in\mathbb{Z}}. \tag{59}$$

The composition σ^n defines a discrete dynamical system on the metric space $(\mathcal{U}, d_{\mathcal{U}})$ [60]. Setting $\sigma^0(\mathbf{u}) := \mathbf{u}$ we have that $\{\sigma^n\}_{n\in\mathbb{Z}}$ represents a group of homeomorphisms on $\mathcal{U}$, which expresses the sequential forward or backward shift in time of all input sequences. Moreover, defining $p : \mathcal{U} \to U$ as the projection mapping

$$\mathbf{u} := \{u[k]\}_{k\in\mathbb{Z}} \longmapsto p(\mathbf{u}) := u[0], \tag{60}$$

we get in U the current value of input sequence $\mathbf{u}$ as $p(\sigma^n(\mathbf{u})) = u[n]$.

We describe the dynamics in response to input using a cocycle map [60, Definition 2.1, page 28] for RNNs.

Definition 5.1 *The nonautonomous dynamical system* (55) *can be described using a* cocycle mapping, $\Phi : \mathbb{Z}_0^+ \times \mathcal{U} \times X \longrightarrow X$, *defined as follows:*

$$\Phi(0, \mathbf{u}, x_0) := x_0, \qquad \forall \mathbf{u} \in \mathcal{U}, \forall x_0 \in X, \tag{61}$$

$$\Phi(n, \mathbf{u}, x_0) := G(p(\sigma^n(\mathbf{u})), \Phi(n-1, \mathbf{u}, x_0)), \qquad \forall n \geq 1, \forall \mathbf{u} \in \mathcal{U}, \forall x_0 \in X \tag{62}$$

Definition 5.1 leads to the following (63) to hold. We state the result without proof since it is well-known in the literature under the name of *cocycle property.*

Lemma 5.1 *The set* $\{\Phi(n, *, \cdot)\}_{n\in\mathbb{Z}_0^+}$ *forms a semigroup of continuous functions from* $\mathcal{U} \times X$ *to* X. *In particular, relations* (61)-(62) *imply the* cocycle property*:*

$$\Phi(m+n, \mathbf{u}, x_0) \;=\; \Phi(n, \sigma^m(\mathbf{u}), \Phi(m, \mathbf{u}, x_0)), \tag{63}$$

for any $m, n \in \mathbb{Z}_0^+$, $x_0 \in X$ *and* $\mathbf{u} \in \mathcal{U}$.

Note that (55) implies that the forward map is always defined, i.e. given a point $x_0 \in X$ and any input $\mathbf{u} \in \mathcal{U}$, the forward trajectory is uniquely defined by (55). On the other hand, it is possible that a backward trajectory does not exist, or may not be unique if one does exist. For example, although the one-dimensional map $G(u, x) := \tanh(\mu x + u)$ is invertible for any fixed $\mu \in (0, 1)$, every backward trajectory constructed from a $x \neq 0$ leads outside the compact set $[-1, 1]$ in a finite

number of backward steps, thus making impossible to obtain a further preimage of tanh. This means it may not be possible to extend the cocycle mapping backward in time. Trajectories that are well-defined in the infinite past play an important role in the nonautonomous dynamics [74], as expressed in the next definition.[10]

Definition 5.2 *An* entire solution *for the system in* (55) *with input* $\mathbf{u} := \{u[k]\}_{k\in\mathbb{Z}} \in \mathcal{U}$ *is a bi-infinite sequence of states* $\{x[k]\}_{k\in\mathbb{Z}}$ *that satisfies* (55) *for all* $k \in \mathbb{Z}$*. In other words,*

$$\Phi(s, \sigma^m(\mathbf{u}), x[m]) = x[m+s]$$

for all $m \in \mathbb{Z}$ *and* $s \in \mathbb{Z}_0^+$*.*

Assuming the existence of an entire solution $\{x[k]\}_{k\in\mathbb{Z}}$ for input $\mathbf{u} \in \mathcal{U}$, and exploiting the forward definition of the cocycle mapping, we can write the past evolution as follows:

$$x[-n] = \Phi(m-n, \sigma^{-m}(\mathbf{u}), x[-m]), \quad \forall n \in \mathbb{Z}_0^+, \forall m \in \mathbb{Z}_0^+ \text{ with } m \geq n. \tag{64}$$

Such a relation expresses the fact that the point $x[-n]$ is the resulting state of the system if we start from $x[-m]$ and drive the dynamics with the sequence of input values $u[-m+1], \ldots, u[-n]$.

5.1.3 Pullback attractors

In this section we introduce some basic definitions of the theory of nonautonomous dynamical systems, including the one of pullback attractor. In fact, the ESP notion first introduced in [51] was formulated using left-infinite sequences and a pullback argument. The following definition extends the notion of entire solution.

Definition 5.3 *Consider a nonautonomous system defined by a cocycle mapping as in Definition 5.1 with input sequence* $\mathbf{u} \in \mathcal{U}$*. A family of nonempty compact sets* $\mathbf{A} = \{A_n\}_{n\in\mathbb{Z}}$ *is called an* invariant nonautonomous set *for input* $\mathbf{u}$ *if*

$$\Phi(s, \sigma^m(\mathbf{u}), A_m) = A_{s+m}.$$

for all $m \in \mathbb{Z}$ *and* $s \in \mathbb{Z}_0^+$*.*

[10] We do not consider invariant sets or entire solutions in terms of the skew product formalism [60, Definition 2.19], but rather in terms of the process [60, Definition 2.14] induced by a given input sequence $\mathbf{u}$.

Entire solutions are invariant nonautonomous sets where each A_n is a single point. Invariant nonautonomous sets turn out to be composed by entire solutions [60, Lemma 2.15]. Replacing "=" with "$\subseteq$" in Definition 5.3, we obtain the definition of a positively invariant family of sets.

Definition 5.4 *A family of nonempty compact sets* $\mathbf{B} = \{B_n\}_{n\in\mathbb{Z}}$ *is called a* positively invariant nonautonomous set *for input* $\mathbf{u}$ *(or simply* $\mathbf{u}$-positively invariant*) if*

$$\Phi(s, \sigma^m(\mathbf{u}), B_m) \subseteq B_{s+m}.$$

for all $m \in \mathbb{Z}$ *and* $s \in \mathbb{Z}_0^+$*. As a special case, a nonempty compact set* B *is called a* positively invariant set *for input* $\mathbf{u}$*, if* $\Phi(s, \sigma^m(\mathbf{u}), B) \subseteq B$*, for all* $m \in \mathbb{Z}$ *and* $s \in \mathbb{Z}_0^+$*.*

Note that the assumption that G is well-defined as a map to X implies that X itself is positively invariant. Invariant sets play an important role for understanding the behaviour of a dynamical system. The most relevant ones are those invariant sets that attract the surrounding trajectories. Since, in nonautonomous systems, the equations ruling the dynamics change with time, the notion of attraction can be formulated in various ways, leading to definitions of *forward attraction* and *pullback attraction*. It is interesting to note that pullback attractors share more properties with their autonomous counterparts [61].

Based on [60, Definitions 3.3 and 3.4], we introduce the notion of (global) pullback attractor as follows.[11] Let h be Hausdorff semi-distance for the metric space (X, d_X); see Appendix B for details.

Definition 5.5 *An invariant nonautonomous set* $\mathbf{A} = \{A_n\}_{n\in\mathbb{Z}}$ *for input* $\mathbf{u}$ *consisting of nonempty compact sets is called a* (global) pullback attractor *of the system* (55) *driven by* $\mathbf{u}$ *if*

$$\lim_{k\to\infty} h(\Phi(k, \sigma^{-k+n}(\mathbf{u}), X), A_n) = 0, \qquad \forall n \in \mathbb{Z}. \tag{65}$$

It is easy to show from this that if $\mathbf{A} = \{A_n\}_{n\in\mathbb{Z}}$ is a pullback attractor for input $\mathbf{u}$ then, for any $m \in \mathbb{Z}$, $\{A_{n+m}\}_{n\in\mathbb{Z}}$ is a pullback attractor for $\sigma^m(\mathbf{u})$.

Manjunath and Jaeger [74] introduce the notion of *natural association* as follows.

[11] Note that as X is bounded, all bounded subsets of X are pullback attracted to A_n [60, Definition 3.4] if and only if X is.

Definition 5.6 *[74, Definition 4] Consider an input sequence* $\mathbf{v} := \{v[k]\}_{k\in\mathbb{Z}} \in \mathcal{U}$. *The sequence of sets* $\{X_n(\mathbf{v})\}_{n\in\mathbb{Z}}$ *defined by*

$$X_n(\mathbf{v}) := \bigcap_{m\leq n} \Phi(n-m, \sigma^m(\mathbf{v}), X), \qquad n \in \mathbb{Z}, \tag{66}$$

is called the natural association *of the process induced by input sequence* $\mathbf{v}$.

Remark 5.2 *In Proposition 5.5, we rigorously prove that the nonautonomous set* $\{X_n(\mathbf{v})\}_{n\in\mathbb{Z}}$ *of Definition 5.6 is the unique invariant compact nonautonomous set* $\{A_n\}_{n\in\mathbb{Z}}$ *which satisfies equation* (65) *for the input sequence* $\mathbf{v}$. *For this reason, from now on we will call the nonautonomous set* $\{X_n(\mathbf{u})\}_{n\in\mathbb{Z}}$ *of* (66) *the global pullback attractor for input* $\mathbf{u}$.

5.2 Pullback attractors of input-driven RNNs

In this section the state space is assumed to be a complete metric space and is denoted as (Y, d_Y), with $Y \subseteq \mathbb{R}^{N_r}$ closed and d_Y is the Euclidean distance. Hence, we temporarily relax the hypothesis of compactness and deal with a state space that is not necessarily bounded. This will serve us to apply results from the literature and so justify the compactness hypothesis of phase space when dealing with RNNs of the type (8)-(9).

For the purpose of this section, it will be useful to introduce the definition of pullback absorbing set. In this regard, we wish to recall from Section 5.1.2 that $\sigma : \mathcal{U} \longrightarrow \mathcal{U}$ (the shift operator) defines an autonomous dynamical system acting on a compact metric space $(\mathcal{U}, d_{\mathcal{U}})$ (the space of all admissible input sequences), and $\Phi : \mathbb{Z}_0^+ \times \mathcal{U} \times Y \longrightarrow Y$ (the cocycle map) describes the nonautonomous dynamics on a complete metric space (Y, d_Y).

Definition 5.7 *[60, Definition 3.17] A nonempty compact subset* $B \subseteq Y$ *is called* pullback absorbing *for a family of inputs* $\mathcal{V} \subseteq \mathcal{U}$ *if*

$$\forall\, \mathbf{u} \in \mathcal{V},\, \forall\, \textit{bounded}\ D \subseteq Y,\, \exists\, N = N(\mathbf{u}, D) \in \mathbb{Z}_0^+ \,:\, \Phi(n, \sigma^{-n}(\mathbf{u}), D) \subseteq B\ \forall n \geq N.$$

Analogously, a nonempty compact subset $B \subseteq Y$ *is called* forward absorbing *for a family of inputs* $\mathcal{V} \subseteq \mathcal{U}$ *if*

$$\forall\, \mathbf{u} \in \mathcal{V},\, \forall\, \textit{bounded}\ D \subseteq Y,\, \exists\, N = N(\mathbf{u}, D) \in \mathbb{Z}_0^+ \,:\, \Phi(n, \mathbf{u}, D) \subseteq B\ \forall n \geq N.$$

While, a nonempty compact subset $B \subseteq Y$ is called uniformly forward absorbing *for a family of inputs $\mathcal{V} \subseteq \mathcal{U}$ if*

$$\forall \text{ bounded } D \subseteq Y, \exists\, N = N(D) \in \mathbb{Z}_0^+ \,:\, \forall\, \mathbf{u} \in \mathcal{V},\, \Phi(k, \mathbf{u}, D) \subseteq B \;\forall k \geq N.$$

Remark 5.3 *Informally, a positively invariant set (see Definition 5.4) is a region in which the internal state, if it enters it, becomes trapped forward in time. On the other hand, an absorbing set is a region in which the internal state is guaranteed to enter it, whether it is in a pullback, forward or uniform sense.*

Remark 5.4 *Note also that a uniformly forward absorbing set is in particular both a pullback absorbing and a forward absorbing one.*

Below we prove, under very few assumptions, that for a generic RNN with leaky-integrator neurons (56), if the image of the activation function ϕ is $(-L, L)$ then the hypercube $[-L, L]^{N_r}$ of phase space is a uniformly forward absorbing, positively invariant set for all input sequences assuming values in a given compact space.

Proposition 5.2 *Let us consider a compact subspace $U \subset \mathbb{R}^{N_i}$ as the set of admissible input values and $\mathcal{U} := U^{\mathbb{Z}}$ as the set of admissible input sequences. Let us consider $Y := \mathbb{R}^{N_r}$ equipped with the Euclidean distance. For all $\mathbf{u} = \{u[k]\}_{k\in\mathbb{Z}} \in \mathcal{U}$, the input-driven dynamics of a leaky RNN with feedback of the output* (8)-(9) *are ruled by*

$$x[k] = (1-\alpha)x[k-1] + \alpha\phi(W_r x[k-1] + W_{in}u[k] + W_{fb}\psi(x[k-1])). \tag{67}$$

If ϕ, ψ are upper semi-continuous functions and ϕ is non-decreasing with image $(-L, L)$, then the set $L \cdot I^{N_r} := [-L, L]^{N_r}$ is positively invariant (see Definition 5.4) for all input sequences in the family $\mathcal{U}$, and it is uniformly forward absorbing for inputs $\mathcal{U}$ (see Definition 5.7).

Proof. The proof is divided in two parts. First we prove that $L \cdot I^{N_r}$ is a $\mathcal{U}$-positively invariant set, then we show that $L \cdot I^{N_r}$ is a uniformly forward absorbing set.

(i) *Positively invariant.*

Let be given an initial condition $x[0]$ such that $\|x[0]\|_\infty \leq L$. Where $\|\cdot\|_\infty$ gives the maximum between all absolute values of the components of a given vector. Thanks to the triangle inequality applied on (56), we have

$$\|x[1]\|_\infty \leq (1-\alpha)\|x[0]\|_\infty + \alpha\|\phi(W_r x[0] + W_{in}u[1] + W_{fb}\psi(x[0]))\|_\infty \leq (1-\alpha)L + \alpha L = L.$$

Analogously, if at any time step it holds that $\|x[N]\|_\infty \leq L$, then it will be $\|x[k]\|_\infty \leq L,\ \forall k \geq N$.

(ii) *Uniformly forward absorbing.*

We will prove that

$$\forall \alpha \in (0,1),\ \forall x[0] \in Y,\ \exists N = N(\alpha, x[0])\ :\ \forall \mathbf{u} \in \mathcal{U}, \quad \|x[k]\|_\infty \leq L,\ \forall k \geq N,$$

which implies that $L \cdot I^{N_r}$ is a uniformly forward absorbing set for the family $\mathcal{U}$.

The case of $\alpha = 1$ brings trivially to the thesis. Thus let us suppose that $\alpha \in (0,1)$. Let be given the initial condition $x[0]$, where we assume $\|x[0]\|_\infty > L$, otherwise the argument of (i) brings to the thesis.[12] Now, since ϕ, ψ are upper semi-continuous functions and ϕ is non-decreasing then the function $\nu : U \times [-R, R]^{N_r} \longrightarrow \mathbb{R}_{\geq 0}$ defined as $\nu(u,x) := \|\phi(W_r x + W_{in} u + W_{fb}\psi(x))\|_\infty$ is upper semi-continuous. Hence, defined $R := \|x[0]\|_\infty$ and since U is compact, there exists a maximum value

$$\eta := \max_{u \in U,\ x : \|x\|_\infty \leq R} \|\phi(W_r x + W_{in} u + W_{fb}\psi(x))\|_\infty.$$

Exploiting recursively the triangle inequality on (67) the following holds

$$\begin{aligned}
\|x[k]\|_\infty \quad &\leq \quad (1-\alpha)\|x[k-1]\|_\infty + \alpha\eta \quad \leq \quad (1-\alpha)^2\|x[k-2]\|_\infty + \alpha\eta(1-\alpha) + \alpha\eta \quad \leq \\
&\leq \quad (1-\alpha)^k\|x[0]\|_\infty + \eta\alpha\sum_{j=0}^{k-1}(1-\alpha)^j \quad = \quad (1-\alpha)^k\|x[0]\|_\infty + \eta\left[1 - \alpha\sum_{j=k}^{\infty}(1-\alpha)^j\right],
\end{aligned}$$

where the last equality holds true in virtue of the geometric series limit $\alpha\sum_{j=0}^{\infty}(1-\alpha)^j = 1$. Now the following inequalities are equivalent,

$$(1-\alpha)^k\|x[0]\|_\infty + \eta\Big[1 - \alpha\sum_{j=k}^{\infty}(1-\alpha)^j\Big] \leq L \iff (1-\alpha)^k\|x[0]\|_\infty \leq L - \eta + \eta\alpha\sum_{j=k}^{\infty}(1-\alpha)^j \iff$$

$$\|x[0]\|_\infty \leq \eta\alpha\sum_{j=k}^{\infty}(1-\alpha)^{j-k} + \frac{L-\eta}{(1-\alpha)^k} \iff \|x[0]\|_\infty \leq \eta\,\underbrace{\alpha\sum_{j=0}^{\infty}(1-\alpha)^j}_{=1} + \frac{L-\eta}{(1-\alpha)^k} \iff$$

$$\|x[0]\|_\infty - \eta \leq \frac{L-\eta}{(1-\alpha)^k}.$$

[12] Note that, although for all $x[s]$ such that $\|x[s]\|_\infty > L$ it holds that $\|x[s+1]\|_\infty \leq (1-\alpha)\|x[s]\|_\infty + \alpha L < \|x[s]\|_\infty$ for all $s \geq 0$, it is not guaranteed that $\|x[k]\|_\infty \leq L$ eventually for some k.

Note that, by the boundedness hypothesis of ϕ, it holds that $\eta \leq L$. If $\eta = L$ then we conclude from the last inequality that

$$\|x[0]\|_\infty \leq \eta = L,$$

which is in contradiction with the assumption of $\|x[0]\|_\infty > L$. Accordingly, $\eta < L$ must hold, leading to the following inequality:

$$(1-\alpha)^k \leq \frac{L-\eta}{\|x[0]\|_\infty - \eta} \quad \Longleftrightarrow \quad k \geq \frac{\ln(L-\eta) - \ln(\|x[0]\|_\infty - \eta)}{\ln(1-\alpha)}.$$

Therefore, after a number of time steps given by $N(\alpha, x[0]) := \dfrac{\ln(L-\eta) - \ln(\|x[0]\|_\infty - \eta)}{\ln(1-\alpha)}$, the internal state of a leaky ESN (56) will surely lie inside the hypercube $L \cdot I^{N_r}$. □

For the sake of clarity, we report here below without proof [60, Theorem 3.20] but using our notation.

Theorem 5.3 *[60, Theorem 3.20] Let $U \subset \mathbb{R}^{N_i}$ be compact and $(\mathcal{U}, d_{\mathcal{U}})$ be the compact metric space of admissible input sequences, where $\mathcal{U} := U^{\mathbb{Z}}$ and $d_{\mathcal{U}}$ as defined in* (57)*. Let $\sigma : \mathcal{U} \longrightarrow \mathcal{U}$ be the shift operator. Let (σ, Φ) be the skew product flow on a complete metric space (Y, d_Y), and Φ defined as Definition 5.1. If there exists a nonempty compact subset $B \subset Y$ which is* pullback absorbing *and* positively invariant *for $\mathcal{U}$, then there exists a unique pullback attractor $\mathbf{A} = \{A_{\mathbf{u}}\}_{\mathbf{u}\in\mathcal{U}}$ with fibres in B uniquely determined by*

$$A_{\mathbf{u}} := \bigcap_{m\geq 0} \overline{\bigcup_{s\geq m} \Phi(s, \sigma^{-s}(\mathbf{u}), B)}, \qquad \forall\, \mathbf{u} \in \mathcal{U}. \tag{68}$$

In addition, since $(\mathcal{U}, d_{\mathcal{U}})$ is a compact metric space the subset $A(\mathcal{U}) := \overline{\bigcup_{\mathbf{u}\in\mathcal{U}} A_{\mathbf{u}}} \subset B$ uniformly (in $\mathcal{U}$) attracts every bounded set of the phase space, that is

$$\lim_{k\to\infty} \sup_{\mathbf{u}\in\mathcal{U}} h(\Phi(k, \mathbf{u}, D), A(\mathcal{U})) = 0, \qquad \forall \text{ bounded } D \subseteq X. \tag{69}$$

Thanks to Proposition 5.2 we can apply Theorem 5.3 on a RNN taking values with phase space the whole $\mathbb{R}^{N_r}$ and use the set $B = [-L, L]^{N_r}$ in order to construct the pullback attractor. Moreover, since $\mathcal{U}$ is compact the second part of Theorem 5.3 implies that the entire nonautonomous dynamics of a RNN is uniformly attracted to a closed subset inside $B = [-L, L]^{N_r}$. This justifies our assumption of considering the whole space as $X = [-L, L]^{N_r}$ whenever referring to the RNN nonautonomous dynamics even with leaky neurons.

Therefore, pullback attractor has component sets made by

$$A_{\mathbf{u}} := \bigcap_{m\geq 0} \overline{\bigcup_{s\geq m} \Phi(s, \sigma^{-s}(\mathbf{u}), X)} \qquad \forall\, \mathbf{u} \in \mathcal{U}. \tag{70}$$

Note that under this formalism the resulting set $A_{\mathbf{u}}$ actually depends only by the left-infinite sequence $\mathbf{u}^- = \{\ldots, u[-2], u[-1], u[0]\}$. Furthermore, since $\mathcal{U} = U^{\mathbb{Z}}$ is shift-invariant, i.e. $\sigma^n(\mathcal{U}) = \mathcal{U}$ for all $n \in \mathbb{Z}$, we can equivalently write (70) as follows

$$A_{\sigma^n(\mathbf{u})} := \bigcap_{m\geq -n} \overline{\bigcup_{s\geq m} \Phi(s+n, \sigma^{-s}(\mathbf{u}), X)} = \bigcap_{m'\leq n} \overline{\bigcup_{s'\leq m'} \Phi(n-s', \sigma^{s'}(\mathbf{u}), X)} \qquad \forall\, \mathbf{u} \in \mathcal{U}, \tag{71}$$

where the last equality is obtained transforming indices as $m' = -m$ and $s' = -s$.

As a consequence, for any fixed input sequence $\mathbf{v} \in \mathcal{U}$, from (71) we get the component sets of the pullback attractor with regard to such input sequence:

$$\mathbf{A}(\mathbf{v}) = \{A_{\sigma^n(\mathbf{v})}\}_{n\in\mathbb{Z}}. \tag{72}$$

Proposition 5.5 below implies that (72) is exactly the natural association (66) defined in Section 5.1.2, for a given input sequence $\mathbf{v} \in \mathcal{U}$.

Lemma 5.4 *Assume the hypotheses of Theorem 5.3 hold. If there exists a $\mathcal{U}$-positively invariant set $B \subset X$, then*

$$\Phi(N+1, \mathbf{v}, B) \subseteq \Phi(N, \sigma(\mathbf{v}), B) \qquad \forall N \in \mathbb{Z}_0^+, \forall \mathbf{v} \in \mathcal{U}. \tag{73}$$

Proof. The base case of $N = 0$, which reads $\Phi(1, \mathbf{v}, B) \subseteq B$, holds thanks to the fact that B is $\mathcal{U}$-positively invariant. Therefore by induction, the result follows from the *cocycle property* (63),

$$\Phi(N+1, \mathbf{v}, B) = \Phi\big(N, \sigma(\mathbf{v}), \Phi(1, \mathbf{v}, B)\big) \subseteq\ \Phi\big(N, \sigma(\mathbf{v}), B\big).$$

□

Proposition 5.5 *Let $U \subset \mathbb{R}^{N_i}$ be compact and $(\mathcal{U}, d_{\mathcal{U}})$ be the compact metric space of admissible input sequences, where $\mathcal{U} := U^{\mathbb{Z}}$ and $d_{\mathcal{U}}$ as defined in* (57)*. Let $\sigma : \mathcal{U} \longrightarrow \mathcal{U}$ be the shift operator. Let (σ, Φ) be the skew product flow on a complete metric space (X, d_X), $X \subseteq \mathbb{R}^{N_r}$ closed and d_X the Euclidean distance, and Φ defined as Definition 5.1. Let us be given an input sequence $\mathbf{v} \in \mathcal{U}$, yielding the subset $\mathcal{V} \subseteq \mathcal{U}$ of input sequences $\mathcal{V} := \{\ldots, \sigma^{-2}(\mathbf{v}), \sigma^{-1}(\mathbf{v}), \mathbf{v}, \sigma^1(\mathbf{v}), \sigma^2(\mathbf{v}), \ldots\}$. If there exists a nonempty compact subset $B \subset X$ which is* pullback absorbing *and* positively invariant *for*

$\mathcal{V}$, then there exists the (global) pullback attractor $\mathbf{A}(\mathbf{v}) := \{A_n\}_{n\in\mathbb{Z}}$ of the dynamics driven by $\mathbf{v}$ and it has component sets

$$A_n := \bigcap_{m\leq n} \Phi(n-m, \sigma^m(\mathbf{v}), B), \qquad \forall n \in \mathbb{Z}. \tag{74}$$

In addition, if $\mathcal{V}$ is compact in $(\mathcal{U}, d_{\mathcal{U}})$ then $A(\mathcal{V}) := \overline{\bigcup_{n\in\mathbb{Z}} A_n} \subset B$ uniformly (in time) attracts the whole phase space driven by $\mathbf{v}$, that is

$$\lim_{k\to\infty} \sup_{n\in\mathbb{Z}} h(\Phi(k, \sigma^n(\mathbf{v}), X), A(\mathcal{V})) = 0. \tag{75}$$

Proof. The proof is a direct application of Theorem 5.3. We need to prove that component sets of (68) coincide with component sets of (74). Let be given a $\mathbf{v} \in \mathcal{U}$. Fixed a $n \in \mathbb{Z}$, let us define a sequence of sets $B_s := \Phi(n+s, \sigma^s(\mathbf{v}), B)$, for $s \geq -n$. Note that, B_s turns out compact since image of a compact B through $\Phi(n+s, \sigma^s(\mathbf{v}), \cdot) \ : \ X \longrightarrow \ X$, a continuous map $\forall n \in \mathbb{Z}$, $s \geq -n$, $\forall \mathbf{v} \in \mathcal{U}$, see Lemma 5.1. It is known that, for a nonincreasing sequence of closed sets $\{B_s\}_{s\geq -n}$, the limit set $L(n) := \bigcap_{m\geq -n} \bigcup_{s\geq m} B_s$ exists and it also holds that $B_m = \bigcup_{s\geq m} B_s$. Therefore, it is sufficient to prove that $B_{s+1} \subseteq B_s$ for having that $L(n) := \bigcap_{m\geq -n} B_m$. Relation $B_{s+1} \subseteq B_s$ holds thanks to Lemma 5.4. Concluding, component sets (68) can be written as $L(n) := \bigcap_{m\geq -n} B_m$, which reads

$$A_n := \bigcap_{m\geq -n} \Phi(n+m, \sigma^m(\mathbf{v}), B) = \bigcap_{m'\leq n} \Phi(n-m', \sigma^{m'}(\mathbf{v}), B),$$

that is (74) of thesis.

To conclude, (75) follows from (69) by noting that we are interested in the subset of input sequences given by $\mathcal{V} = \{\ldots, \sigma^{-2}(\mathbf{v}), \sigma^{-1}(\mathbf{v}), \mathbf{v}, \sigma^1(\mathbf{v}), \sigma^2(\mathbf{v}), \ldots\}$. Therefore, the supremum $\sup_{\mathbf{u}\in\mathcal{V}}$ can be expressed with the supremum $\sup_{n\in\mathbb{Z}}$ of (75) of thesis. □

Remark 5.5 *Note that, in general, for a given $\mathbf{v} \in \mathcal{U}$ the set $\mathcal{V} = \{\sigma^n(\mathbf{v})\}_{n\in\mathbb{Z}}$ is not compact in $\mathcal{U}$. In particular, if $\mathbf{v}$ is aperiodic then $\mathcal{V}$ will have limit points that are not contained in $\mathcal{V}$.*

5.3 An echo index for recurrent neural networks

We first consider the ESP linked to an input for driven RNNs as defined in [74]. Then, we introduce a more general definition accounting for reliable RNN behaviour in a phase space subset only. This generalisation also leads to a well-defined notion of multistable behaviour for driven RNNs.

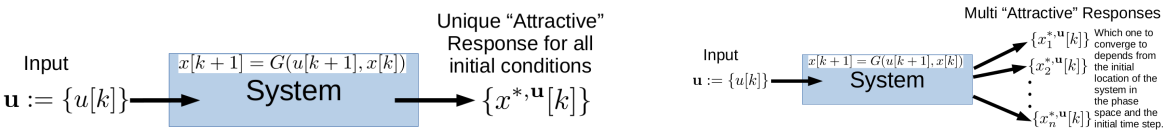


Figure 21: **Left:** scheme of an input-driven system with the echo state property. The system can be interpreted as a filter responding with a unique solution when driven by an input signal $\mathbf{u}$. **Right:** the input-driven system develops a number n of solutions in response to a driving input signal $\mathbf{u}$. Although, this eventuality might represent a symptom of malfunctioning, e.g. due to unsuccessful training, the emergence of more than one reliable options can be intentional for the purpose of solving a particular task.

5.3.1 Entire solutions and the Echo State Property

Recall from Definition 5.2 that an *entire solution* for input $\mathbf{v} = \{v[k]\}_{k\in\mathbb{Z}} \in \mathcal{U}$ is a sequence $\mathbf{x} = \{x[k]\}_{k\in\mathbb{Z}}$ that satisfies (55), for all $k \in \mathbb{Z}$.

Definition 5.8 *[74, Definition 2] The nonautonomous system* (55) *has the* Echo State Property *for input* $\mathbf{v} \in \mathcal{U}$ *if there exists a unique entire solution* $\mathbf{x}$*.*

Here below, in Proposition 5.6, we give a simple proof of the fact that, if the ESP holds for a given $\mathbf{v} \in \mathcal{U}$, then the entire solution is the pullback attractor as in Definition 5.5, using the notation of the cocycle mapping introduced in Section 5.1.2.

Proposition 5.6 *If* (55) *has the ESP for input* $\mathbf{v} \in \mathcal{U}$*, then the (unique) entire solution is the global pullback attractor of the system* (55) *with input* $\mathbf{v}$*.*

Proof. First, note that if $\{\mathbf{x}^*[k]\}_{k\in\mathbb{Z}}$ is an entire solution for input $\mathbf{v} \in \mathcal{U}$, then by the definition of Φ, it holds true that $x^*[n] = \Phi(n-m, \sigma^m(\mathbf{v}), x^*[m])\ \forall\, m < n, \forall n \in \mathbb{Z}$. Compactness of X guarantees that $x^*[m] \in X$ from which it follows that $x^*[n] \in \bigcap_{m<n} \Phi(n-m, \sigma^m(\mathbf{v}), X)\ \forall n \in \mathbb{Z}$. The ESP hypothesis means $\{\mathbf{x}^*[k]\}_{k\in\mathbb{Z}}$ is the only entire solution for input $\mathbf{v} \in \mathcal{U}$. Therefore, we need to prove that for all $n \in \mathbb{Z}$ the pullback attractor is composed of the singleton $x^*[n] = \bigcap_{m<n} \Phi(n-m, \sigma^m(\mathbf{v}), X)$. We prove this by contradiction: suppose there exists a $n_0 \in \mathbb{Z}$, where w.l.o.g. $n_0 = 0$, and a $y_0 \in X$ with $y_0 \neq x^*[0]$ such that

$$\{x^*[0], y_0\} \subset \bigcap_{m<0} \Phi(-m, \sigma^m(\mathbf{v}), X).$$

This means that $y_0 \in \Phi(-m, \sigma^m(\mathbf{v}), X)\ \forall m \leq -1$, i.e. for each $m \leq -1$ there exists at least one point, let us denote it $y[m] \in X$, such that $y_0 = \Phi(-m, \sigma^m(\mathbf{v}), y[m])$. In other words, there exists a backward trajectory $\mathbf{y}^- := \{\ldots, y[-2], y[-1], y[0] := y_0\}$ for the point y_0. Now, by iterating (55) forward in time from $y[0] := y_0$, we get the entire solution $\mathbf{y} := \{y[k]\}_{k\in\mathbb{Z}}$. This gives a contradiction. □

Jaeger [51] shows that, if the ESP holds *for all input sequences* assuming values in a given compact set, then the unique entire solution is forward attracting [51, Definition 4]. Below, in Proposition 5.8, we provide an alternative proof of this fact by applying [60, Theorem 3.44] to RNNs as special class of nonautonomous dynamical systems.

5.3.2 ESP for all input sequences implies forward convergence, linked to an input does not

For the sake of completeness, in Theorem 5.7 we show, using our framework, a result found in the literature, which links pullback attraction with forward attraction in the particular case where the global pullback attractor is an entire solution.

Theorem 5.7 *[60, Theorem 3.44] Let $U \subset \mathbb{R}^{N_i}$ be compact and $(\mathcal{U}, d_{\mathcal{U}})$ be the compact metric space of admissible input sequences, where $\mathcal{U} := U^{\mathbb{Z}}$ and $d_{\mathcal{U}}$ as defined in* (57)*. Let $\sigma : \mathcal{U} \longrightarrow \mathcal{U}$ be the shift operator. Let (σ, Φ) be the skew product flow on the complete metric space $Y = \mathbb{R}^{N_r}$, with the Euclidean distance, and Φ defined as Definition 5.1. Assume there exists a nonempty* compact *subset $B \subset Y$ such that:*

- *B is a $\mathcal{U}$-*positively invariant set*; and*
- *B is* uniformly forward absorbing for $\mathcal{U}$*.*

Suppose that for all $\mathbf{u} \in \mathcal{U}$ the (global) pullback attractor for input $\mathbf{u}$ is an entire solution. Then, for any given input sequence $\mathbf{v} \in \mathcal{U}$ such entire solution for input $\mathbf{v}$, denoted as $\mathbf{x} = \{x[k]\}_{k\in\mathbb{Z}}$, is also attracting in forward sense uniformly in time, that is

$$\lim_{k\to\infty} \sup_{n\in\mathbb{Z}} h(\Phi(k, \sigma^n(\mathbf{v}), D), x[n+k]) = 0, \qquad \forall \text{ bounded } D \subseteq Y.$$

In Proposition 5.2 we proved that $B = [-L, L]^{N_r}$ fulfils the hypothesis of Theorem 5.7 for a generic leaky RNNs with ϕ, ψ upper semi-continuous functions and ϕ non-decreasing with bounded image. Therefore, Theorem 5.7 for a generic RNN reads as: if the global pullback attractor is an entire solution for all $\mathbf{u} \in U^{\mathbb{Z}}$ then such unique entire solution of the system is also forward attracting uniformly in time. More precisely, we can state the following result.

Proposition 5.8 *Let be given an $\alpha \in (0, 1]$ and real matrices W_r, W_{in}, W_{fb} of dimensions, respectively, $N_r \times N_r$, $N_r \times N_i$, $N_r \times N_o$. Let $\phi : \mathbb{R} \to (-L, L)$, $\psi : \mathbb{R}^{N_r} \to \mathbb{R}^{N_o}$ be upper semi-continuous functions and ϕ non-decreasing. Consider the following input-driven leaky RNN with feedback of the output*

$$x[k] = G(u[k], x[k-1]), \qquad x \in \mathbb{R}^{N_r},\ u \in U \subset \mathbb{R}^{N_i} \text{ compact}, \tag{76}$$

$$G(u, x) = (1 - \alpha)x + \alpha\phi(W_r x + W_{in} u + W_{fb}\psi(x)). \tag{77}$$

If the ESP as originally introduced in [51, Definition 1] holds for the input-driven RNN (76) *w.r.t the compact input space $\mathcal{U} = U^{\mathbb{Z}}$ then the unique entire solution of such nonautonomous system is* uniformly state contracting *[51, Definition 4].*

Limitations of pullback attractors in describing reliable RNN behaviour Here, we emphasise the difference between pullback and forward mechanisms of attraction with an example of Kloeden *et al.* [59].

Let us consider an input-driven system where X is the compact $[-1, 1]$ and the equation ruling the dynamics is

$$x[k+1] = \tanh\left(u[k]\frac{x[k]}{1 + |x[k]|}\right). \tag{78}$$

Now, let us drive the system with the input sequence $u[k] = a$ when $k \geq 0$, and $u[k] = a^{-1}$ when $k < 0$, for some constant $a > 1$. The process generated by this input sequence has only one entire solution, i.e. $x[k] = 0$, for all $k \in \mathbb{Z}$. Accordingly, Proposition 5.6 implies that such an entire solution $x[k] \equiv 0$ is the global pullback attractor. Nevertheless, that unique entire solution is "attractive" in the past, yet it is "repulsive" in the future. Moreover, all initial conditions (except $x = 0$) asymptotically tend to assume two possible values forward in time; see Figure 22 for a visual representation.

Definition 5.8 implies that the system taken into account here has the ESP and therefore we would conclude that it produces a unique reliable response. However, our example shows how such

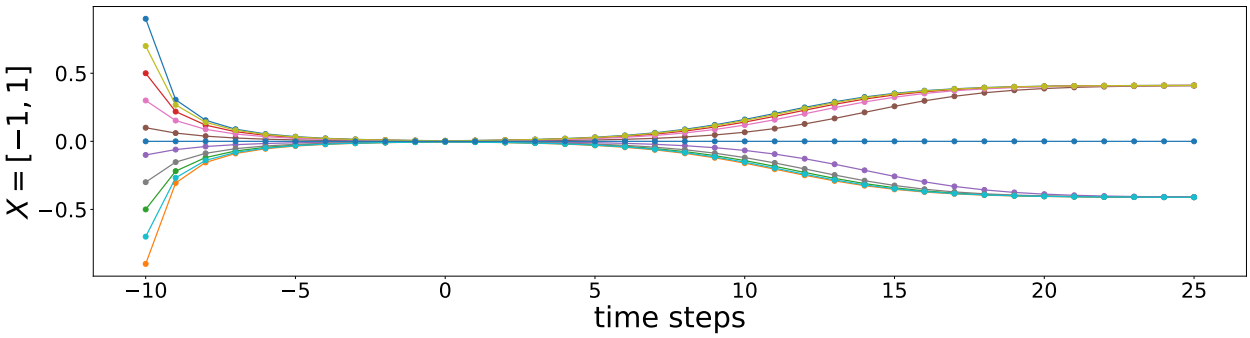


Figure 22: Equation (78), with $a = 1.5$, has been evolved from $k = -10$ to $k = 25$ starting with 11 initial conditions in $[-1, 1]$. One of these initial conditions is $x = 0$ which is exactly on the global pullback attractor of the system driven by the input sequence $u[k] = a$ for $k \geq 0$ and $u[k] = a^{-1}$ for $k < 0$, with $a = 1.5$.

a conclusion might be misleading. Furthermore, the unique entire solution which is supposed to be the "echo" of the input sequence is actually produced by the system exclusively when the system is initialised exactly at $x_0 = 0$.

It appears clear that the mere consideration of pullback attraction w.r.t. an input sequence is not enough to meet the ideas behind the ESP (i.e. the idea of reliable, unique asymptotic behaviour in response to an input sequence).

5.3.3 Uniformly attracting entire solutions and the echo index

If the ESP (Definition 5.8) does not hold, then clearly a wide variety of behaviours are possible. Here, we are interested in describing the case where there is a finite number of stable responses to an input sequence. For this we state a notion of *uniformly attracting entire solution* (UAES).[13] This is a local attractor: as noted previously [86], it is possible for a pullback attractor to have a decomposition into a number of local attractors.

Definition 5.9 *Consider a fixed input sequence* $\mathbf{u} \in \mathcal{U}$*, an entire solution* $\{x[k]\}_{k\in\mathbb{Z}}$ *and a positively invariant nonautonomous set* $\{B[k]\}_{k\in\mathbb{Z}}$ *composed of compact sets.*

[13] Compared to [60, Definition 3.48(iii)]: although the attraction is uniform, we do not require the neighbourhood to be uniform in k. On the other hand, this is a special case that only considers entire solutions that are attractors.

(i) If

$$\lim_{k\to\infty}\left(\sup_{j\in\mathbb{Z}} h(\Phi(k,\sigma^j(\mathbf{u}),B[j]),x[j+k])\right)=0 \tag{79}$$

then we say $\{B[k]\}_{k\in\mathbb{Z}}$ *is* uniformly attracted *to* $\{x[k]\}_{k\in\mathbb{Z}}$.

(ii) We say $\{x[k]\}_{k\in\mathbb{Z}}$ *is a UAES if there is a neighbourhood* $\{B[k]\}_{k\in\mathbb{Z}}$ *of* $\{x[k]\}_{k\in\mathbb{Z}}$ *that is uniformly attracted to* $\{x[k]\}_{k\in\mathbb{Z}}$.

Remark 5.6 *Note that if* $\{x[k]\}_{k\in\mathbb{Z}}$ *is a UAES, then it is both a forward attractor and a pullback attractor [60, Definition 3.11]. Moreover, note that a neighbourhood* $\{B[k]\}_{k\in\mathbb{Z}}$ *of* $\{x[k]\}_{k\in\mathbb{Z}}$ *has necessarily positive Lebesgue measure. Therefore a UAES* $\{x[k]\}_{k\in\mathbb{Z}}$ *cannot be an entire solution attracting a nonautonomous set which has some fibres with zero Lebesgue measure.*

UAESs allow us to rigorously define the number of stable RNN responses as a function of the specific input sequence driving the dynamics.

Definition 5.10 *We say the system* (55) *with input* $\mathbf{u}$ *admits a* decomposition *into* $n\geq 1$ *UAESs if there are* n *UAESs* $\{x_1[k]\}_{k\in\mathbb{Z}},\dots,\{x_n[k]\}_{k\in\mathbb{Z}}$ *such that, for all* $\eta>0$ *and* $i=1,\dots,n$, *there are neighbourhoods* $\{B_i^\eta[k]\}_{k\in\mathbb{Z}}$ *uniformly attracted by* $\{x_i[k]\}_{k\in\mathbb{Z}}$ *and*

$$\lambda(X\setminus\bigcup_{i=1}^{n}B_i^\eta[k])<\eta,\qquad \forall k\in\mathbb{Z}, \tag{80}$$

where λ *denotes the Lebesgue measure on* $X\subset\mathbb{R}^{N_r}$. *We say this is a* proper decomposition *if in addition*

$$\inf_{k\in\mathbb{Z}} d_X(x_i[k],x_j[k])>0,\qquad \forall i,j\in\{1,\dots,n\},\ i\neq j. \tag{81}$$

Note that (79) together with (81) imply that $B_i^\eta[k]\cap B_j^\eta[k]=\emptyset$ at each time step $k\in\mathbb{Z}$. Condition (80) means we can exclude a neighbourhood of the repelling dynamics that may sit on basin boundaries, while (81) implies there is a uniform threshold that can be used to separate all attractors.

Definition 5.11 *We say the system* (55) *driven by input* $\mathbf{u}\in\mathcal{U}$ *has* echo index $n\geq 1$, *and write*

$$\mathcal{I}(\mathbf{u})=n,$$

if it admits a proper decomposition into n *UAESs. In this case, we say* (55) *has the* n-Echo State Property *(*n*-ESP) for input* $\mathbf{u}$.

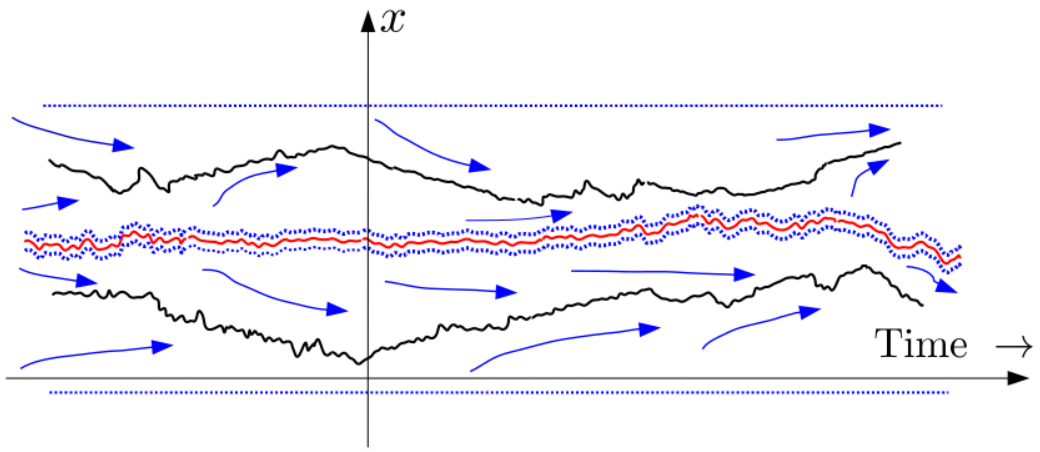


Figure 23: Blue dashed curves represent the boundaries of two nonautonomous compact sets which are uniformly attracted by two UAESs depicted as black curves. In red is depicted an entire solution that delimits the boundary between the basins of attraction of the two UAESs. Uniform convergence is guaranteed as long as, at each time step, we exclude a small neighbourhood of the separatrix solution shown in red; the smaller the excluded neighbourhood of the separatrix solution, the smaller the residual of the measure of (80).

Note that if $\mathbf{u}$ has echo index n, then so does $\sigma^m(\mathbf{u})$ for any $m \in \mathbb{Z}$.

Definition 5.11 describes the case where a finite number $\mathcal{I}(\mathbf{u})$ of "attracting" solutions (here modeled as UAESs) emerge as a consequence of driving the system with input sequence $\mathbf{u}$. Apart from a zero measure set of initial conditions, the RNN dynamics converge to one of these UAESs: which of these is actually followed will depend on the particular initial condition, see Figure 23.

Our definition of echo index assumes that all local nonautonomous attractors are UAESs. This is a restriction even when the input is null. In fact, there can be invariant curves or chaotic sets that attract some portion of the phase space. For all those cases where a proper decomposition in n UAESs does not exist, we say the driven system (55) has *indefinite* echo index. We refer the reader to [16] for phase space decompositions in more general nonautonomous attractors, and to [26] for decompositions framed within the theory of random dynamical systems.

Condition (81) guarantees that the UAESs do not merge into each other, implying that the number n of stable responses to an input in the infinite past coincides with the number of stable responses in the infinite future. Hence, the echo index characterises those input-driven systems

whose degree of multistability is invariant over time.

Remark 5.7 *The set of input-driven systems possessing the ESP according to Definition 5.8 and those having echo index 1 do not coincide. The example in Section 5.3.2 represents an input-driven system possessing the ESP according to Definition 5.8 whilst, however, a UAES does not exist. Therefore the echo index is not well-defined for such cases. Vice-versa, there might be cases where input-driven systems have one UAES attracting a nonautonomous set with full Lebesgue measure (case with echo index 1), but such a UAES might coexist with other entire solutions attracting some zero measure set of phase space. In such a scenario, there will be more than one entire solution and thus Definition 5.8 is not satisfied. Nevertheless, such an input-driven system will produce a unique stable response for almost all initial conditions.*

It is possible to extend the definition of echo index to sets of input sequences as follows.

Definition 5.12 *Consider a system* (55) *with a set of possible input sequences* $\mathcal{V} \subset \mathcal{U}$ *and define*

$$\mathcal{V}_{[n]} = \{\mathbf{v} \in \mathcal{V} \ : \ \mathcal{I}(\mathbf{v}) = n\}. \tag{82}$$

We say that such a system has echo index set $\mathcal{N}(\mathcal{V})$ *for a subset* $\mathcal{V} \subset \mathcal{U}$ *if*

$$\mathcal{N}(\mathcal{V}) = \{n \in \mathbb{N} \ : \ \mathcal{V}_{[n]} \neq \emptyset\}. \tag{83}$$

We can split a set of inputs $\mathcal{V}$ into a disjoint union according to the echo index:

$$\mathcal{V} = \left(\bigcup_{n \in \mathcal{N}(\mathcal{V})} \mathcal{V}_{[n]} \right) \cup \mathcal{V}_{[\text{ind}]}, \tag{84}$$

where $\mathcal{V}_{[\text{ind}]}$ are those input sequences that give an indefinite echo index.

5.4 Theoretical results

A kind of input-driven fixed point theorem is stated and proved in Theorem 5.9. Thanks to this fundamental result we are able to deduce other results. In particular, in Theorem 5.11 we describe one way in which UAESs can emerge in systems driven by low amplitude inputs. Specularly, in Proposition 5.14, we prove that forcing RNNs with large-amplitude inputs induces echo index 1 (and the ESP). Moreover, in Theorem 5.15, we investigate on the stability of the echo index depending on the particular metric we choose to equip the space of input sequences with.

5.4.1 A theorem of existence and uniqueness for uniformly attracting entire solutions

First, let us delimit the region of uniform contraction of the map $G : U \times X \longrightarrow X$.

Definition 5.13 *Given a positive real number μ such that $0 < \mu < 1$, we define the set of* linear μ-contraction uniform in U *of the map G as*

$$C(\mu, U) := \{x \in X \ : \ \sup_{u \in U} \|D_x G(u, x)\| \leq \mu\}, \tag{85}$$

where $\|\cdot\|$ denotes the matrix norm induced by the Euclidean norm on $\mathbb{R}^{N_r}$.

Remark 5.8 *The set in* (85) *is the phase space region where the nonautonomous dynamics contract at each time step with a rate of at most μ, i.e. where each autonomous map $G(u, \cdot) : X \to X$, with $u \in U$, contracts with a rate of at most μ.*

For the sake of clarity, we formally extend the definition of a positively invariant set $B \subset X$ of Definition 5.4 to all input sequences in a family $\mathcal{V} \subseteq \mathcal{U}$.

Definition 5.14 *A nonempty compact subset $B \subseteq X$ is called* positively invariant *for $\mathcal{V} \subseteq \mathcal{U}$ (or* $\mathcal{V}$-positively invariant*) if*

$$\Phi(k, \mathbf{u}, B) \subseteq B, \qquad \forall \mathbf{u} \in \mathcal{V}, \ k \in \mathbb{Z}_0^+$$

Remark 5.9 *If the family of input sequences under consideration is $\mathcal{V} := \{\sigma^n(\mathbf{v})\}_{n \in \mathbb{Z}}$, for some $\mathbf{v} \in \mathcal{U}$, then we simply say that B is a* $\mathbf{v}$-positively invariant *set. This coincides with the case in Definition 5.4 of a constant nonautonomous set $B_n \equiv B$, which is positively invariant for the input $\mathbf{v}$.*

The following theorem gives sufficient conditions to prove the existence and uniqueness of a UAES in a compact and convex phase space subset. The proof of the theorem exploits the contraction of the family of autonomous maps $\{G(u, \cdot)\}_{u \in U}$ defining the nonautonomous system. This result can be seen as a deterministic input-driven fixed point theorem, see [103] for similar results in the context of autonomous dynamical systems or [50] in the context of random dynamical systems. In Figure 24 a visual depiction of Theorem 5.9.

Theorem 5.9 *Let μ be a positive real number such that $0 < \mu < 1$. Suppose Q_μ is a $\mathcal{U}$-positively invariant nonempty compact set such that it is contained inside $C(\mu, U)$ of Definition 5.13, and*

suppose further that Q_μ is convex. Then, for all $\mathbf{v} \in \mathcal{U}$ the system (55) driven by the input sequence $\mathbf{v}$ admits a unique entire solution in Q_μ. In particular, if such entire solution is contained inside the interior of Q_μ, then this entire solution is a UAES.

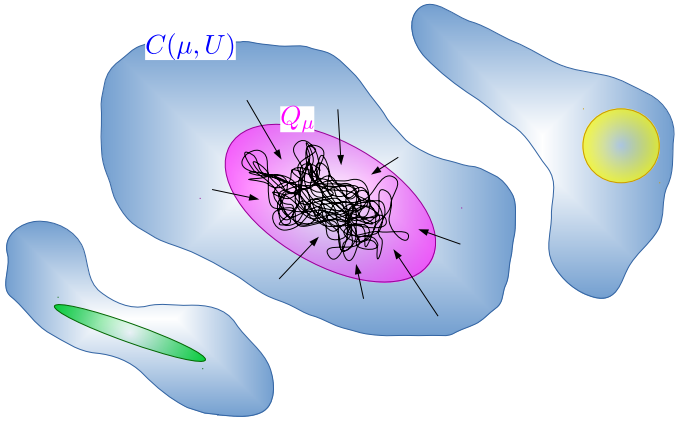


Figure 24: Sketch of the ingredients of Theorem 5.9. In blue is represented the set defined in Definition 5.13 as $C(\mu, U)$. Note that such region of phase space might be disconnected. In magenta a convex and $\mathcal{U}$-positively invariant set Q_μ is is depicted. In black the unique UAES emerging inside Q_μ is illustrated. Note that UAESs can appear extremely tangled due to the action of the input, see Figure 28 in Section 5.5.3 for an actual UAES developed in an ESN solving a real task. In yellow and green other convex sets inside $C(\mu, U)$ where Theorem 5.9 can be applied, and hence other attractive reliable responses emerge.

Proof. The assumption of convexity of Q_μ implies that for all $x_0, y_0 \in Q_\mu$ the line segment $\ell_{[x_0,y_0]} := \{z = (1-s)x_0 + sy_0 \mid s \in [0,1]\}$ lies inside Q_μ and the following inequality holds

$$\begin{aligned} d_X(G(u,x_0),G(u,y_0)) =& \|G(u,x_0) - G(u,y_0)\| = \left\| \int_{s=0}^{1} D_x G(u,(1-s)x_0 + sy_0) \cdot y \; ds \right\| \\ &\leq \sup_{z \in \ell_{[x_0,y_0]}} \|D_x G(u,z)\| \; \|x_0 - y_0\| = \sup_{z \in \ell_{[x_0,y_0]}} \|D_x G(u,z)\| \; d_X(x_0,y_0) \leq \mu \; d_X(x_0,y_0), \end{aligned} \tag{86}$$

where the last inequality holds whenever $u \in U$, by the hypothesis that $Q_\mu \subseteq C(\mu, U)$.

Now, let us take an input sequence $\mathbf{v} \in U^{\mathbb{Z}}$ and an arbitrarily chosen initial time step $k_0 \in \mathbb{Z}$. Denote $x_k := \Phi(k, \sigma^{k_0}(\mathbf{v}), x_0)$ and $y_k := \Phi(k, \sigma^{k_0}(\mathbf{v}), y_0)$, for some $k > 0$. Now, $\mathbf{v} = \{v[k]\}_{k \in \mathbb{Z}}$

assumes values in U, hence in particular $v[k_0] \in U$ and (86) reads $d_X(x_1, y_1) \leq \mu\ d_X(x_0, y_0)$. By hypothesis Q_μ is positively invariant for all input sequences $\mathbf{u} \in U^{\mathbb{Z}}$, thus $x_1, y_1 \in Q_\mu$ if $x_0, y_0 \in Q_\mu$. Therefore, we can repeat the same argument on the pair (x_1, y_1) with $v[k_0 + 1]$ in place of (x_0, y_0) with $v[k_0]$, obtaining that $d_X(x_2, y_2) \leq \mu\ d_X(x_1, y_1) \leq \mu^2\ d_X(x_0, y_0)$, and so on for all pairs (x_k, y_k) forming the forward trajectories. In other words, by induction we proved that

$$d_X(\Phi(k, \sigma^{k_0}(\mathbf{u}), x_0), \Phi(k, \sigma^{k_0}(\mathbf{u}), y_0)) \leq \mu^k d_X(x_0, y_0) \quad \forall \mathbf{u} \in \mathcal{U}, \forall k_0 \in \mathbb{Z}, \forall x_0, y_0 \in Q_\mu, \forall k > 0. \tag{87}$$

Moreover, for any given input sequence $\mathbf{v} \in \mathcal{U}$, thanks to the fact that Q_μ is $\mathbf{v}$-positively invariant, the nonautonomous set defined as

$$A_n := \bigcap_{m \leq n} \Phi(n - m, \sigma^m(\mathbf{v}), Q_\mu), \qquad n \in \mathbb{Z} \tag{88}$$

is an invariant nonautonomous set for input $\mathbf{v}$ as in Definition 5.3 which is confined inside Q_μ; see [60, Lemma 2.20] for a proof of this fact.

We will prove that the invariant nonautonomous set of (88) is indeed a UAES. Let us first prove that it is an entire solution. By contradiction, let us assume there are $x_s, y_s \in A_s$ distinct points, i.e. $x_s \neq y_s$, for some $s \in \mathbb{Z}$. By the invariance of $\{A_n\}_{n \in \mathbb{Z}}$, it follows that $\Phi(k, \sigma^{-k+s}(\mathbf{v}), A_{-k+s}) = A_s$. Therefore, there exist $x', y' \in A_{-k+s}$ such that $x_s = \Phi(k, \sigma^{-k+s}(\mathbf{v}), x')$ and $y_s = \Phi(k, \sigma^{-k+s}(\mathbf{v}), y')$. Hence, the following inequalities hold:

$$0 < d_X(x_s, y_s) = d_X(\Phi(k, \sigma^{-k+s}(\mathbf{v}), x'), \Phi(k, \sigma^{-k+s}(\mathbf{v}), y')) \overset{(87)}{\leq} \mu^k d_X(x', y') \leq \mu^k \operatorname{diam}(Q_\mu).$$

Therefore, by the compactness of Q_μ, there cannot exist two distinct points in A_s, i.e. the invariant nonautonomous set (88) is an entire solution $\mathbf{x} = \{x[n]\}_{n \in \mathbb{Z}}$ for input $\mathbf{v}$. This fact implies that inside Q_μ there can be only one entire solution. Indeed, if there exists another entire solution $\mathbf{y} = \{y[n]\}_{n \in \mathbb{Z}}$ for input $\mathbf{v}$, then it means that $\Phi(k, \sigma^{-k+s}(\mathbf{v}), y[-k + s]) = y[s]$ holds for all $s \in \mathbb{Z}$ and for all $k > 0$. Therefore, if such entire solution is contained in Q_μ, i.e. $y[n] \in Q_\mu$ for all $n \in \mathbb{Z}$, then $y[s] \in \bigcap_{k \geq 0} \Phi(k, \sigma^{-k+s}(\mathbf{v}), Q_\mu) = \bigcap_{m \leq s} \Phi(s - m, \sigma^m(\mathbf{v}), Q_\mu) = A_s$, which coincides with $x[s]$.

Now, we prove that such unique entire solution is uniformly attracting according to (79) of Definition 5.9. Let us take a sequence of integers $\{j_k\}_{k \in \mathbb{Z}}$. Note that

$$h(\Phi(k, \sigma^{j_k}(\mathbf{u}), Q_\mu), x[j_k + k]) = \max_{x \in Q_\mu} d_X(\Phi(k, \sigma^{j_k}(\mathbf{u}), x), x[j_k + k]) \tag{89}$$

$$= \max_{x \in Q_\mu} d_X(\Phi(k, \sigma^{j_k}(\mathbf{v}), x), \Phi(k, \sigma^{j_k}(\mathbf{v}), x[j_k])). \tag{90}$$

Now, relation (87) implies that $\max_{x\in Q_\mu} d_X(\Phi(k,\sigma^{j_k}(\mathbf{v}),x),\Phi(k,\sigma^{j_k}(\mathbf{v}),x[j_k])) \leq \mu^k \operatorname{diam}(Q_\mu)$ and hence that $\lim_{k\to\infty} h(\Phi(k,\sigma^{j_k}(\mathbf{u}),Q_\mu),x[j_k+k]) = 0$, regardless of the sequence of integers $\{j_k\}_{k\in\mathbb{Z}}$. Therefore, $\lim_{k\to\infty}\sup_{n\in\mathbb{Z}} h(\Phi(k,\sigma^n(\mathbf{u}),Q_\mu),x[n+k]) = 0$. In particular, all forward trajectories starting at the same time step from initial conditions inside Q_μ synchronise to a (unique) common solution which is indeed $\mathbf{x}$. Moreover, as long as $x[n] \in \operatorname{int}(Q_\mu), \forall n \in \mathbb{Z}$, then we can find a neighbourhood of $\mathbf{x} = \{x[n]\}_{n\in\mathbb{Z}}$ such that Definition 5.9 is satisfied. □

5.4.2 n-ESP for systems perturbed by low-amplitude inputs

In this section, we analyse the input-driven dynamics of an autonomous RNN perturbed by low-amplitude input sequences. A special case of (55) is the constant-input case:

$$x[k+1] = G(u_0, x[k]), \tag{91}$$

i.e. with $\mathbf{u} \equiv u_0$ constant, which gives an autonomous nonlinear system. We will look at how the autonomous system (91), possessing a uniformly stable point x^*, reacts under small external perturbations represented by a forcing input sequence. Therefore, input values are taken in a neighbourhood of u_0, i.e. $\mathcal{U} := B_r(u_0)^{\mathbb{Z}}$ with $r > 0$ but small. The idea is that stable fixed points will become UAESs in the nonautonomous setting, giving rise to the n-ESP.

In the following, we will assume that the autonomous map (91) possesses a uniformly attracting stable fixed point according to the following definition.

Definition 5.15 *Let us be given an autonomous map $F : X \longrightarrow X$ and a fixed point $x^* \in X$ for F, i.e. $F(x^*) = x^*$. We call x^* a Uniformly Attracting Stable Point (UASP) if there is an $0 < M < 1$ and $\delta > 0$, such that*

$$d_X(F(z),x^*) < M d_X(z,x^*) \qquad \forall z \in B_\delta(x^*). \tag{92}$$

The property (92) imposes some algebraic condition on the linearised map at the fixed point x^*.

Lemma 5.10 *Let us be given an autonomous map F with one UASP x^* characterised by a contraction rate $0 < M < 1$. Let F be differentiable at x^*. Denote by $A := D_xF(x^*)$ the Jacobian matrix evaluated onto the fixed point x^*, and its maximum singular value as $\sigma(A)$. Then $\sigma(A) \leq M$.*

Proof. Let us proceed by contradiction assuming that $\sigma(A) > M$. By definition,

$$\sigma(A) = \max_{y \in \mathbb{R}^{N_r} \setminus \{0\}} \frac{\|Ay\|}{\|y\|}.$$

Therefore, for a small enough $\varepsilon > 0$, it must exist a unit vector $\overrightarrow{v}$ such that

$$\|A(c\overrightarrow{v})\| \geq (M + \varepsilon)\|c\overrightarrow{v}\| \qquad \forall c \in \mathbb{R} \setminus \{0\}. \tag{93}$$

Now let us move on the line $z = x^* + c\overrightarrow{v}$ and consider the linearisation of the map F around x^*, which reads

$$F(z) = x^* + A(z - x^*) + R(z - x^*), \tag{94}$$

where the rest of the expansion $R(z - x^*)$ is such that

$$\lim_{z \to x^*} \frac{\|R(z - x^*)\|}{\|z - x^*\|} = 0, \tag{95}$$

whenever the map F is regular enough in x^*. Now, since x^* is a UASP, there exists a $\delta > 0$ such that

$$\|F(z) - x^*\| < M\|z - x^*\|, \qquad \forall z \in B_\delta(x^*). \tag{96}$$

By means of the expansion (94) and applying the reverse triangle inequality we get

$$\Big|\|A(z - x^*)\| - \|R(x - x^*)\|\Big| \leq \|A(x - x^*) + R(x - x^*)\| = \|F(x) - x^*\| < M\|x - x^*\|. \tag{97}$$

Now, if z is close enough to x^* along the direction pointed by $\overrightarrow{v}$, then it holds that $\|A(z - x^*)\| \geq \|R(z - x^*)\|$. Indeed, (95) implies that for any $\epsilon > 0$ there exists a $c_\epsilon > 0$ such that $\|R(c\overrightarrow{v})\| \leq \epsilon\|c\overrightarrow{v}\|$ for all $|c| \leq c_\epsilon$. Moreover, exploiting the fact that z is chosen such that $z - x^* = c\overrightarrow{v}$ satisfies (93), then it holds that $(M + \varepsilon)\|c\overrightarrow{v}\| = (M + \varepsilon)\|z - x^*\| \leq \|A(z - x^*)\|$. Hence, for the choice $\epsilon = (M + \varepsilon)$ there exists a $c_{M+\varepsilon} > 0$ such that $\|R(z - x^*)\| \leq \|A(z - x^*)\|$ for all $z - x^* = c\overrightarrow{v}$ having $|c| \leq c_{M+\varepsilon}$. Therefore, we can rewrite (97) as follows

$$\|R(z - x^*)\| > \|A(z - x^*)\| - M\|z - x^*\|. \tag{98}$$

Finally, applying again (93) in (98), we obtain

$$\|R(z - x^*)\| > (M + \varepsilon - M)\|z - x^*\|, \tag{99}$$

i.e. for all $z - x^* = c\overrightarrow{v}$ having $|c| \leq c_{M+\varepsilon}$ holds $\|R(z - x^*)\| > \varepsilon\|z - x^*\|$, which is in contradiction with relation (95). □

Theorem 5.11 *Suppose there is a UASP x^* of (91), characterised by a rate of contraction M in a ball of radius δ. Then, $\forall\, 0 < \varepsilon < 1 - M,\ \exists\ 0 < \delta_\varepsilon < \delta$, $r_\varepsilon > 0$ such that for all input sequences $\mathbf{u} \in \overline{B_{r_\varepsilon}(u_0)}^{\mathbb{Z}}$ there exists a unique UAES which uniformly attracts $B_{\delta_\varepsilon}(x^*)$ with a rate of $M + \varepsilon$.*

Proof. We claim that $\forall\, 0 < \varepsilon < 1 - M,\ \exists\, 0 < \delta_\varepsilon < \delta\ :\ \forall\, 0 < \gamma < \delta_\varepsilon(1 - M),\ \exists\, \zeta_{\gamma,\varepsilon} > 0$ such that for all input sequences $\mathbf{u} \in \overline{B_{\zeta_{\gamma,\varepsilon}}(u_0)}^{\mathbb{Z}}$ there exists a unique UAES confined in $\overline{B_{\frac{\gamma}{1-M}}(x^*)}$ which uniformly attracts the neighbourhood $B_{\delta_\varepsilon}(x^*)$ with a rate of $M + \varepsilon$. This will prove the thesis choosing r_ε in the statement of Theorem 5.11 as $\zeta_{\gamma,\varepsilon}$.
By definition, $\|A\| = \sigma(A)$, where $\|\cdot\|$ denotes the matrix norm induced by the Euclidean norm on $\mathbb{R}^{N_r}$. Lemma 5.10 and the fact that x^* is a UASP for $G(u_0, \cdot) : X \to X$ imply that $\|D_x G(u_0, x^*)\| \leq M$. Assumption 5.1(i) implies continuity of $\|D_x G\|$ at (u_0, x^*), hence for all $\varepsilon > 0$ we can find a $\rho_\varepsilon > 0$ and a $\delta_\varepsilon > 0$ such that $\max_{u' \in \overline{B_{\rho_\varepsilon}(u_0)}} \|D_x G(u', x)\| \leq M + \varepsilon, \quad \forall x \in \overline{B_{\delta_\varepsilon}(x^*)}$. In terms of Definition 5.13 we can state that for all $0 < \varepsilon < 1 - M$ and denoting $\mu = M + \varepsilon < 1$, there exist $\rho_\varepsilon, \delta_\varepsilon > 0$ such that $\overline{B_{\delta_\varepsilon}(x^*)} \subseteq C(\mu, \overline{B_{\rho_\varepsilon}(u_0)})$. Now, continuity of $G(\cdot, x) : U \to X$ for each $x \in \overline{B_{\delta_\varepsilon}(x^*)}$ implies that for all $\gamma > 0, \exists \zeta_\gamma > 0$ such that

$$G(\overline{B_{\zeta_\gamma}(u_0)}, \overline{B_{\delta_\varepsilon}(x^*)}) \subseteq B_\gamma(G(u_0, \overline{B_{\delta_\varepsilon}(x^*)}). \tag{100}$$

The hypothesis that x^* is a UASP for the autonomous map $F(\cdot) = G(u_0, \cdot)$ reads $G(u_0, \overline{B_{\delta_\varepsilon}(x^*)}) \subseteq B_{M\delta_\varepsilon}(x^*)$, as long as $\delta_\varepsilon < \delta$. Therefore, we obtain from (100) that

$$G(\overline{B_{\zeta_\gamma}(u_0)}, \overline{B_{\delta_\varepsilon}(x^*)}) \subseteq \overline{B_{M\delta_\varepsilon + \gamma}(x^*)}. \tag{101}$$

Therefore, as long as we choose γ such that $M\delta_\varepsilon + \gamma < \delta_\varepsilon$, i.e. $\gamma < \delta_\varepsilon(1 - M)$, we get from (101) that $G(\overline{B_{\zeta_\gamma}(u_0)}, \overline{B_{\delta_\varepsilon}(x^*)}) \subset \overline{B_{\delta_\varepsilon}(x^*)}$, which proves that $\overline{B_{\delta_\varepsilon}(x^*)}$ is a $\mathcal{U}$-positively invariant set, considering $\mathcal{U} = \overline{B_{\zeta_\gamma}(u_0)}^{\mathbb{Z}}$. This allows us to make use of Theorem 5.9 with $Q_\mu = \overline{B_{\delta_\varepsilon}(x^*)}$ and $\mathcal{U} = \overline{B_{\zeta_{\gamma,\varepsilon}}(u_0)}^{\mathbb{Z}}$, with $\zeta_{\gamma,\varepsilon} = \min\{\rho_\varepsilon, \zeta_\gamma\}$. Hence, we proved that $\forall\, 0 < \varepsilon < 1 - M,\ \exists\, 0 < \delta_\varepsilon < \delta\ :$ $\forall\, 0 < \gamma < \delta_\varepsilon(1-M),\ \exists\, \zeta_{\gamma,\varepsilon} > 0$ such that for all input sequences $\mathbf{u} \in \overline{B_{\zeta_{\gamma,\varepsilon}}(u_0)}^{\mathbb{Z}}$ there exists a unique entire solution $\mathbf{x}$ confined in $\overline{B_{\delta_\varepsilon}(x^*)}$. Moreover, note that once chosen a γ such that $M\delta_\varepsilon + \gamma < \delta_\varepsilon$ we can iterate the above argument in (101) leading to $G(\overline{B_{\zeta_\gamma}(u_0)}, \overline{B_{M\delta_\varepsilon+\gamma}(x^*)}) \subseteq \overline{B_{M^2\delta_\varepsilon+\gamma(1+M)}(x^*)}$ and so on. This leads to a sequence $\delta_n = M\delta_{n-1} + \gamma$, for all $n \geq 1$, starting with $\delta_0 = \delta_\varepsilon$, which is strictly decreasing. Unfolding such a recursive relation leads to $\delta_n = M^n\delta_\varepsilon + \gamma \sum_{j=0}^{n-1} M^j$, whose limit is $\dfrac{\gamma}{1 - M}$. As a consequence $\mathbf{x}$ is confined in $\overline{B_{\frac{\gamma}{1-M}}(x^*)}$ which is strictly contained in the

interior of $\overline{B_{\delta_\varepsilon}(x^*)}$; thus $\mathbf{x}$ is an UAES uniformly attracting the neighbourhood $B_{\delta_\varepsilon}(x^*)$ with a rate of $M + \varepsilon$. □

The assumption on x^* to be a UASP makes proof of existence of a UAES easier. Nevertheless, the same results can presumably be proved just assuming x^* to be a stable hyperbolic fixed point, using exponential dichotomies; see for example [91].

As a corollary of Theorem 5.11 we get the following result.

Corollary 5.12 *If for a given constant input value u_0 the autonomous map* (91) *presents a number $n \geq 1$ of UASPs, then the nonautonomous system $x[k] = G(u[k], x[k-1])$ driven by any deterministic input sequence $u[k]$ assuming values in a neighbourhood of u_0 will present (at least) n UAESs, as in Definition 5.9. Therefore, input sequences $\mathbf{u} \in B_r(u_0)^{\mathbb{Z}}$ with small enough $r > 0$ can give rise to input-driven systems with the n-ESP.*

5.4.3 ESP for RNNs driven by large-amplitude inputs

We rigorously prove, in Proposition 5.14, that driving RNNs with large-amplitude inputs generally leads to echo index 1. Our result is similar to the one obtained in [74, Theorem 2], with the important differences that we (i) account for the uniform (i.e. also forward in time) attractiveness of solutions, (ii) consider also the feedback of the output, and (iii) highlight how W_{in} induces a geometric structure in the space of input values relevant for the analysis of reliable responses of an input-driven RNN.

First, we prove a technical Lemma that requires some additional definitions. We will denote with $(W)_{(j)}$ the jth row of a matrix W.

Definition 5.16 *We define*

$$H_j := \{u \in \mathbb{R}^{N_i} \ : \ (W_{in})_{(j)} \cdot u = 0\} \tag{102}$$

the hyperplane of input values which vanish the jth row of the matrix W_{in}. Moreover, given real values $\epsilon > 0$ (meant to be small) and $R > 0$ (meant to be large), we define

$$P_j(\epsilon, R) := \left\{ u \in \mathbb{R}^{N_i} \setminus B_R(0) \ : \ \frac{\left|(W_{in})_{(j)} \cdot u\right|}{\|(W_{in})_{(j)}\| \|u\|} \geq \epsilon \right\}. \tag{103}$$

Remark 5.10 *Note that, denoted with θ the angle between the vectors pointing to $(W_{in})_{(j)}$ and u, then* $\cos(\theta) = \dfrac{(W_{in})_{(j)} \cdot u}{\|(W_{in})_{(j)}\|\|u\|}$. *Therefore, the set $P_j(\epsilon, R)$ is basically the set $\mathbb{R}^{N_i} \setminus B_R(0)$ where we cut out all the lines which form an angle $\theta \in (\arccos(\epsilon), \arccos(-\epsilon))$, i.e. those close to the hyperplane H_j.*

Lemma 5.13 *Let W_r, W_{fb}, W_{in} be real matrices of dimensions, respectively, $N_r \times N_r$, $N_r \times N_o$, $N_r \times N_i$, and $\psi \in C^0(\mathbb{R}^{N_r}, \mathbb{R}^{N_o})$. Consider the functions $\xi_j : \mathbb{R}^{N_i} \times [-L, L]^{N_r} \longrightarrow \mathbb{R}$, for $j = 1, \ldots, N_r$, defined as $\xi_j(u, x) := (W_{in})_{(j)} \cdot u + f_j(x)$, where $f_j(x) := (W_r)_{(j)} \cdot x + (W_{fb})_{(j)} \cdot \psi(x)$. If $(W_{in})_{(j)}$ is not the null vector then in the subset of input values $P_j(\epsilon, R)$ we can make the function $|\xi_j(u, x)|$ large as much as we want. Precisely, for all $\epsilon > 0$ and for all $\overline{\xi} > 0$ there exists an $R_{\overline{\xi},\epsilon} > 0$ (which depends on $\|(W_{in})_{(j)}\|^{-1}$ and $\max_{x \in [-L,L]^{N_r}} |f_j(x)|$) such that for all $R \geq R_{\overline{\xi},\epsilon}$ we have that $\inf \{ \ |\xi_j(u, x)| \ : \ u \in P_j(\epsilon, R)\} \geq \overline{\xi}$ holds for all $x \in [-L, L]^{N_r}$.*

Proof. If $(W_{in})_{(j)}$ is the null vector then $\xi_j(u, x) = f_j(x)$, i.e. it does not depend on the input, thus let us assume $(W_{in})_{(j)}$ is not the null vector. Then, a hyperplane $H_j = \{u \in \mathbb{R}^{N_i} \ : \ (W_{in})_{(j)} \cdot u = 0\}$ is defined in the space of input values $\mathbb{R}^{N_i}$ such that H_j is the orthogonal space of the vector pointing to $(W_{in})_{(j)}$.

In general,

$$(W_{in})_{(j)} \cdot u = \|(W_{in})_{(j)}\| R \cos(\theta)$$

where R is the norm of u and θ is the angle between the vectors pointing to $(W_{in})_{(j)}$ and u. Let us fix arbitrarily a $R > 0$ and consider u to vary on the surface of the ball of radius R centred on the origin of $\mathbb{R}^{N_i}$, i.e. $u \in \partial B_R(0)$, thus we have that $\|(W_{in})_{(j)}\| R$ is constant and $\cos(\theta)$ can assume any value in $[-1, 1]$. In particular, $(W_{in})_{(j)} \cdot u$ can assume any value in the interval $\big[-\|(W_{in})_{(j)}\| R, \ \|(W_{in})_{(j)}\| R\big]$.

Now, let us take an arbitrary $\epsilon > 0$ and consider the subset $P_j(\epsilon, R)$ for some $R > 0$. Note that for any given $u \in P_j(\epsilon, R)$, denoted with $\varepsilon := |\cos(\theta)|$, where θ is the angle between $(W_{in})_{(j)}$ and u, and with ρ the norm of $\|u\|$, then $\varepsilon \geq \epsilon$ and $\rho \geq R$. Therefore, we have that

$$|(W_{in})_{(j)} \cdot u| \geq \|(W_{in})_{(j)}\| R \epsilon \tag{104}$$

holds for all $u \in P_j(\epsilon, R)$. Therefore, the reverse triangle inequality leads to

$$|\xi_j(u, x)| = |f_j(x) + (W_{in})_{(j)} \cdot u| \geq \Big| \big|f_j(x)\big| - \big|(W_{in})_{(j)} \cdot u\big| \Big| \geq \big|(W_{in})_{(j)} \cdot u\big| \ - \big|f_j(x)\big|, \tag{105}$$

where the last inequality holds for all $x \in [-L, L]^{N_r}$ and for all $u \in P_j(\epsilon, R)$ as long as $R \geq \dfrac{\sigma}{\epsilon \|(W_{in})_{(j)}\|}$, where we denoted $\sigma = \max_{x \in [-L,L]^{N_r}} |f_j(x)|$.

Furthermore, for all $\overline{\xi} > 0$ if $R \geq R_{\overline{\xi},\epsilon} := \dfrac{\overline{\xi} + \sigma}{\epsilon \|(W_{in})_{(j)}\|}$ then thanks to (104) and (105) we get that the following

$$|\xi_j(u,x)| \geq \left|(W_{in})_{(j)} \cdot u\right| - \left|f_j(x)\right| \geq \|(W_{in})_{(j)}\| R_{\overline{\xi},\epsilon} \epsilon - \sigma \geq \overline{\xi}$$

holds for all $x \in [-L, L]^{N_r}$ and for all $u \in P_j(\epsilon, R)$. In conclusion, we have that for all $\epsilon > 0$ and for all $\overline{\xi} > 0$ there exists an $R_{\overline{\xi},\epsilon} > 0$ such that for all $R \geq R_{\overline{\xi},\epsilon}$ we have that

$$\inf_{u \in P_j(\epsilon,R)} |\xi_j(u,x)| \geq \overline{\xi}$$

holds for all $x \in [-L, L]^{N_r}$. □

We now state the main result of this section.

Proposition 5.14 *Consider the RNN* (8)-(9) *with a bounded* $\phi \in C^1(\mathbb{R}, (-L, L))$ *that is monotonically increasing and* ϕ' *has a unique maximum point at* $\xi = 0$ *and* $\psi \in C^1(\mathbb{R}^{N_r}, \mathbb{R}^{N_o})$. *Then, the following two statements are true:*

(i) If the following condition holds, for some $0 < \mu < 1$,

$$\phi'(0) \|W_r + W_{fb} D_x \psi(x)\| \leq \mu, \qquad \forall x \in [-L, L]^{N_r}, \tag{106}$$

then for all sets of input values $V \subseteq \mathbb{R}^{N_i}$ *it holds that* $X \subseteq C(\mu, V)$ *of* (85). *In particular, Theorem 5.9 implies that, for all compact sets* $U \subset \mathbb{R}^{N_i}$ *and for all input sequences* $\mathbf{u} \in U^{\mathbb{Z}}$, *the RNN* (8)-(9) *driven by* $\mathbf{u}$ *admits exactly one UAES which attracts the whole phase space* $X = [-L, L]^{N_r}$, *i.e. it has echo index 1.*

(ii) If each row $(W_{in})_{(j)}$ *is non zero, then* $\forall \varepsilon > 0$ *and* $0 < \mu < 1$ *there exists a radius* $R_{\varepsilon,\mu} > 0$ *such that* $X \subseteq C(\mu, V)$ *as long as the dynamics are driven by inputs assuming values in a set* V *such that* $V \subseteq \bigcap_{j=1}^{N_r} P_j(\epsilon, R_{\epsilon,\mu})$ *of* (103). *In particular, for any compact* $U \subset \bigcap_{j=1}^{N_r} P_j(\epsilon, R_{\epsilon,\mu})$, *Theorem 5.9 implies that for any* $\mathbf{u} \in U^{\mathbb{Z}}$ *there exists a unique UAES which attracts the whole phase space* X, *i.e. it has echo index 1.*

Proof. First, let us consider the case of $\alpha = 1$. Given a subset of input values $V \subseteq \mathbb{R}^{N_i}$, we define

$$l_V(x) := \sup_{u \in V} \|S(u,x)\|,$$

where $S(u,x)$ is given in (21). Then, for all $x \in [-L,L]^{N_r}$, the norm of D_xG (20), can be upper-bounded with:

$$\sup_{u\in V}\|D_xG(u,x)\| \leq l_V(x)\,\|M(x)\|.$$

Now, since $S(u,x)$ is a diagonal matrix, we have

$$l_V(x) = \sup_{u\in V}\|S(u,x)\| = \sup_{u\in V}\max_{j=1,\dots,N_r}|\phi'(\xi_j(u,x))| \tag{107}$$

$$= \sup_{u\in V}\phi'\left(\min_{j=1,\dots,N_r}|\xi_j(u,x)|\right) \tag{108}$$

$$= \phi'\left(\inf_{u\in V}\left\{\min_{j=1,\dots,N_r}\left|\xi_j(u,x)\right|\right\}\right), \tag{109}$$

where the last two equalities hold because ϕ' is a continuous function with a unique maximum point in $\xi = 0$.

(i) Now, since $\phi'(z)$ assumes its maximum value at $z = 0$ then $l_V(x) \leq \phi'(0)$. Therefore, if there exists a $0 < \mu < 1$ such that (106) holds then

$$\sup_{u\in V}\|D_xG(u,x)\| \leq l_V(x)\,\|M(x)\| \leq \phi'(0)\|M(x)\| \overset{(106)}{\leq} \mu.$$

In other words, if (106) holds then $X \subseteq C(\mu, V)$ for any $V \subseteq \mathbb{R}^{N_i}$. Now, since $X = [-L,L]^{N_r}$ is a convex and compact set which is $U^{\mathbb{Z}}$-positively invariant for any compact $U \subset \mathbb{R}^{N_i}$ (see Proposition 5.2), we conclude the claim, applying Theorem 5.9 with $\mathcal{U} := U^{\mathbb{Z}}$ on the whole space $X = [-L,L]^{N_r}$.

(ii) Let us fix an arbitrary $\epsilon > 0$ and a $\mu \in (0,1)$. We will prove that there exists a $R_{\epsilon,\mu} > 0$ such that $\sup_{u\in V}\|D_xG(u,x)\| \leq \mu$ w.r.t. input values in a subset $V \subseteq \bigcap_{j=1}^{N_r} P_j(\epsilon, R_{\epsilon,\mu})$. First of all, thanks to Lemma 5.13 we know that for all $\epsilon > 0$ and $\overline{\xi} > 0$ there exists an $R_{\overline{\xi},\epsilon} > 0$ such that

$$\inf_{u\in P_j(\epsilon,R_{\overline{\xi},\epsilon})}\left\{\min_{j=1,\dots,N_r}\left|\xi_j(u,x)\right|\right\} \geq \overline{\xi}, \qquad \forall x \in [-L,L]^{N_r},$$

as long as each row $(W_{in})_{(j)}$ is not the zero vector. Moreover, $\phi \in C^1(\mathbb{R},(-L,L))$ is monotonic and its image is $(-L,L)$, hence $\lim_{|\xi|\to\infty}\phi'(\xi) = 0$. Hence, $l_V(x)$ can be made arbitrarily low, regardless of x, as long as $V \subseteq P_j(\epsilon,R)$ with R large enough. More precisely, denoting $\tilde{\sigma} := \max_{x\in[-L,L]^{N_r}}\|M(x)\|$, for all $\epsilon > 0$ and $0 < \mu < 1$ there exists an $R_{\epsilon,\mu} > 0$, such that $l_V(x) \leq \dfrac{\mu}{\tilde{\sigma}}$ (an consequently such that $\sup_{u\in V}\|D_xG(u,x)\| \leq \mu$) holds for all $x \in [-L,L]^{N_r}$,

w.r.t. input values in a subset $V \subseteq \bigcap_{j=1}^{N_r} P_j(\epsilon, R_{\epsilon,\mu})$. Finally, Theorem 5.9 implies that for any compact $U \subset \bigcap_{j=1}^{N_r} P_j(\epsilon, R_{\epsilon,\mu})$ the RNN driven by any $\mathbf{u} \in U^{\mathbb{Z}}$ admits a UAES which attracts the whole phase space, i.e. it has echo index 1.

On the other hand, when $\alpha \in (0,1)$, we have

$$\|D_x G(u,x)\| \leq (1-\alpha)^{N_r} + \alpha^{N_r}\|S(u,x)M(x)\| \leq 1 - \alpha\big(1 - \|S(u,x)M(x)\|\big),$$

where the last inequality holds since the function $f(x) = (1-\alpha)^x + \alpha^x s$ is strictly monotonically decreasing for all $\alpha \in (0,1)$ and for all $s \geq 0$. Now, from these last inequalities we note that if $x \in [-L,L]^{N_r}$ is such that $\sup_{u\in V}\|S(u,x)M(x)\| \leq \mu \in (0,1)$ then it holds that $\sup_{u\in V}\|D_x G(u,x)\| \leq 1 - \alpha(1-\mu)$, which leads back to the already proved case with $\alpha = 1$. Therefore, we can apply Theorem 5.9 on the whole space $[-L,L]^{N_r}$ which is contained inside $C\big(1-\alpha(1-\mu), V\big)$. □

Remark 5.11 *Remarkably, the result of Proposition 5.14(ii) does not rely on any particular assumption on the matrices W_r, W_{fb}, and the readout ψ, as long as the input values are large enough in amplitude and are sufficiently far from $H = \bigcup_{j=1}^{N_r} H_j$, in a sense made precise in Remark 5.10. The need to exclude the sets H_j is due to the fact that, for all input sequences $\mathbf{u} \in H_j^{\mathbb{Z}}$ in some phase space directions, the RNN is basically a mere function of x. Hence, in general, in these directions the dynamics might be repulsive unless we impose some conditions on the matrices W_r, W_{fb} and the function ψ, as the condition in* (106)*. Nevertheless, note that the set $H = \bigcup_{j=1}^{N_r} H_j$ is the union of hyperplanes in $\mathbb{R}^{N_i}$, which has a zero Lebesgue measure in the space of input values. Roughly speaking, this means that if the input sequence $\mathbf{u}$ is a realisation of a random process which generates values inside U according to a uniform distribution, then for all $R > 0$ such that $U \subset \mathbb{R}^{N_i} \setminus B_R(0)$ the probability to observe values of the input outside of the compact space $U \cap \bigcap_{j=1}^{N_r} P_j(\epsilon, R)$ will be proportional to ϵ.*

Note that, in the presence of feedback of the output with linear readout, i.e. $\psi(x) = W_o x$, whenever the activation function is such that $\phi'(0) = 1$, as for example $\phi = \tanh$, then from (106) *we obtain the condition on the maximum singular value $\|M\| < 1$, where M is the effective recurrent matrix* (19)*. In particular, when there is no feedback, i.e. $W_{fb} = 0$, we recover the well-known sufficient condition $\|W_r\| < 1$.*

5.4.4 Stability of echo index to inputs

We now show how the metric imposed on the space of input sequences affects the stability of the echo index to input perturbations. Notably, we show that the widely-used (e.g. [41]) product topology (57) implies that echo indices greater than 1 cannot remain the same (i.e. are not stable) even when considering arbitrarily small perturbations of input sequences.

Theorem 5.15 *Let us consider an RNN of the form* (8)-(9) *driven by inputs in the space* $\mathcal{U}$ *of bounded input sequences* $\{u[k]\}_{k\in\mathbb{Z}}$*, i.e.* $\sup_k\{d_{\mathbb{R}^{N_i}}(0,u[k])\}<+\infty$*. If* $(W_{in})_{(j)}$ *is non zero for all* $j=1,\ldots,N_r$*, then the subset* $\mathcal{U}_{[1]}\subset\mathcal{U}$ (82) *of bounded input sequences for which an RNN driven by* $\{v[k]\}_{k\in\mathbb{Z}}\in\mathcal{U}_{[1]}$ *has echo index 1 is dense in* $\mathcal{U}$ *with the product topology* (57)*.*

Proof. Assume an RNN of the form (8)-(9) satisfying hypothesis of Proposition 5.14(ii) is given, i.e. such that $(W_{in})_{(j)}$ is non zero for all $j=1,\ldots,N_r$. Let's define $\mathcal{U}:=\left\{\{u[k]\}_{k\in\mathbb{Z}}\ :\ \sup_k\{d_{\mathbb{R}^{N_i}}(0,u[k])\}<+\infty\right\}$ as the universe of possible input sequences for driving the RNN. Let us equip the space $\mathcal{U}$ with the metric given by (57) and let's denote with $\mathcal{U}_{[1]}$, as in (82), the subset of input sequences that gives echo index 1.

We claim that for all $\mathbf{u}\in\mathcal{U}$ and $\varepsilon>0$ there exists a $\mathbf{v}\in\mathcal{U}_{[1]}$ such that $d_{\text{prod}}(\mathbf{u},\mathbf{v})<\varepsilon$.

Let us take an arbitrary input sequence $\mathbf{u}=\{u[k]\}_{k\in\mathbb{Z}}\in\mathcal{U}$ and assume $\mathbf{u}$ has not echo index 1. Proposition 5.14(ii) says that $\forall\varepsilon>0$ and $0<\mu<1$ there exists a radius $R_{\varepsilon,\mu}>0$ such that $X\subseteq C(\mu,U)$ as long as the dynamics are driven by inputs assuming values in a compact U such that $U\subset\bigcap_{j=1}^{N_r}P_j(\epsilon,R_{\epsilon,\mu})$ of (103). Now, let's define an input sequence $\mathbf{v}=\{v[k]\}_{k\in\mathbb{Z}}$ such that $v[k]=u[k]$ for all $|k|\leq M$, for some $M>0$, and $v[k]\in U$ for all $|k|>M$, for some compact $U\subset\bigcap_{j=1}^{N_r}P_j(\epsilon,R_{\epsilon,\mu})$ such that $U\subset B_{R_{\epsilon,\mu}+1}(0)$. Now, driving the RNN with $\mathbf{v}$ Proposition 5.14(ii) implies that it is well defined a left-infinite internal state sequence $\mathbf{p}^{-M}:=\{\ldots,p[-M-2],p[-M-1]\}$ which consists of the left branch (i.e. relatively to the infinite past) of the global pullback attractor for the input sequence $\mathbf{v}$. Therefore, at time-step $k=-M-1$ we end up sitting exactly at the location $p[-M-1]\in X$. Now, from the initial condition $x_0=p[-M-1]$ we move driven by the sequence $\mathbf{v}$ in the window of time $[-M,M]$ tracing an orbit $x[k]=\Phi(M+1+k,\sigma^{-M}(\mathbf{v}),x_0)$, for $k=-M,-M+1,\ldots,M$, in phase space and ending somewhere at position $x[M]\in X$. Now, despite the potentially expanding dynamics in the window of time $[-M,M]$, Proposition 5.14(ii) ensures that, when driven by $\mathbf{v}$ for $k>M$ the entire phase space contracts forward in time with a rate μ. Hence, all forward trajectories starting from any initial condition at time step $k=M$

synchronise with the one starting from $x[M]$ in the future. In particular, there exists a unique entire solution $\mathbf{z}$ for the system driven by $\mathbf{v}$ and it is defined as

$$z[k] = \begin{cases} p[k] & k < -M \\ \Phi(M+1+k, \sigma^{-M}(\mathbf{v}), p[-M-1]) & k \geq -M. \end{cases} \tag{110}$$

Moreover, $\mathbf{z}$ is a UAES which uniformly attracts the whole phase space. Therefore, the RNN driven by $\mathbf{v}$ has echo index 1. Finally, note that

$$d_{\text{prod}}(\mathbf{u}, \mathbf{v}) = \sum_{|k|>M} \frac{d_U(u[k], v[k])}{2^{|k|}} \leq \text{diam}(U) \sum_{|k|>M} \frac{1}{2^{|k|}} \leq \text{diam}(U) \frac{1}{2^{M+1}} \leq (R_{\epsilon,\mu} + 1) \frac{1}{2^M} \tag{111}$$

can be made arbitrarily small by increasing M. This proves that, for the topology induced by the metric $d_{\text{prod}}(\cdot, \cdot)$, the set $\mathcal{U}_{[1]}$ is dense in $\mathcal{U}$. □

Remark 5.12 *As a consequence of Theorem 5.15, if an RNN has echo index $\mathcal{I}(\mathbf{u}) \geq 2$ for input sequence $\mathbf{u}$, then, considering the product topology induced by the metric in* (57)*, the echo index is not stable. In fact, the product topology allows arbitrarily large variations of input values in the far past and future of an input sequence $\mathbf{u}$. This means that there exists an arbitrarily small perturbation of $\mathbf{u}$ leading to an input sequence $\tilde{\mathbf{u}}$ such that $\mathcal{I}(\tilde{\mathbf{u}}) = 1$. Interestingly, the proof of Theorem 5.15 can be adapted to prove that the set $\mathcal{U}_{[ind]}$ is dense in $\mathcal{U}$. This observation suggests that the product topology is not suitable for the analysis of the stability of reliable responses of an RNN to variations of input sequences, e.g. due to noise, while the uniform metric* (58) *seems to be a more appropriate choice. In fact, if a system* (55) *driven by an input sequence $\mathbf{u}$ has a well-defined echo index, then its echo index might be stable when considering perturbations of $\mathbf{u}$ which are uniformly bounded in time, as suggested by Theorem 5.11.*

5.5 Examples of input-driven RNNs with multistable dynamics

In this section, we consider the RNN in (8) and a linear model for the RNN output (9). The results here are illustrative of some of the behaviours we expect in more general cases. In Section 5.5.1, we show a two-dimensional example of RNN with echo index 2, highlighting the need for the concepts introduced in Definition 5.10. Then, in Section 5.5.2, we show how the echo index may change also depending on the characteristics of the input sequence driving the dynamics. Section 5.5.3 shows the application of our modeling framework to the high-dimensional task in [46, Figure 5].

5.5.1 An example of switching system with echo index 2

We now report some numerical experiments on a simple RNN that can be thought of as switching dynamics between two autonomous maps. We will consider the space of inputs defined as $\mathcal{U} := \{u_1, u_2\}^{\mathbb{Z}}$, i.e. as made by sequences assuming just two possible values: u_1 and u_2. Let us consider the map $G_\alpha(u, x) := (1-\alpha)x + \alpha \tanh(W_r x + W_{in} u)$ where $W_r = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{3}{2} \end{bmatrix}$, $W_{in} = I_2$ is the identity matrix and x a real vector of dimension two. What follows can be observed for any value of $\alpha \in (0, 1]$. We select $\alpha = \frac{1}{4}$ because a small value slows down the state-update and hence highlights transient dynamics[14].

The space of possible input sequences is $\mathcal{U} = \{u_1, u_2\}^{\mathbb{Z}}$, where $u_1 := \begin{pmatrix} \frac{1}{4} \\ \frac{3}{20} \end{pmatrix}$ and $u_2 := -u_1$. Therefore, the nonautonomous dynamics driven by input $\mathbf{u} \in \mathcal{U}$ consists of a switching pattern between the two component maps defined as $f_1(x) := G(u_1, x)$ and $f_2(x) := G(u_2, x)$.

The autonomous system $x[k+1] = f_1(x[k])$ has two asymptotically stable fixed points with a saddle between them along the vertical line of $x_1 \approx 0.45$, see Figure 25. Analogously, the autonomous system $x[k+1] = f_2(x[k])$ has two asymptotically stable points with a saddle between them along the vertical line $x_1 \approx -0.45$.

Exploiting Theorem 5.9 we are able to prove the existence of two UAESs for every deterministic input sequence $\mathbf{u} \in \{u_1, u_2\}^{\mathbb{Z}}$. The Jacobian matrix of f_1 reads

$$D_x f_1(x_1, x_2) = \begin{bmatrix} 1 - \alpha/2\big[1 + \tanh^2(x_1/2 + 1/4)\big] & 0 \\ 0 & 1 + \alpha/2\big[1 - 3\tanh^2(3x_2/2 + 3/20)\big] \end{bmatrix}.$$

Diagonal elements are thus eigenvalues and also singular values, hence $\sigma(D_x f_1(x_1, x_2)) = 1 + \alpha/2\big[1 - 3\tanh^2(3x_2/2 + 3/20)\big]$. Thus $\|D_x f_1(x_1, x_2)\| = 1 + \alpha/2\big[1 - 3\tanh^2(3x_2/2 + 3/20)\big]$. The region of phase space where contraction occurs, i.e. $\|D_x f_1(x_1, x_2)\| < 1$, consists of 2 connected components divided by the strip $-0.54 < x_2 < 0.34$, approximately. Similarly, for f_2 we have that

[14] As explained in equations (21)-(23) of [20], α scales the velocity field. In our example, the equation ruling the dynamics is $x[k+1] = (1-\alpha)x[k] + \alpha \tanh(W_r x[k] + W_{in} u[k+1])$ which can be equivalently written as

$$x[k+1] = x[k] + \alpha\Big(\tanh(W_r x[k] + W_{in} u[k+1]) - x[k]\Big).$$

In the latter expression is evident that the vector to be added to the current state $x[k]$ in order to get the next state $x[k+1]$ is scaled by the parameter α. In this sense, a smaller α will slow down the dynamics highlighting the transients.

$\|D_x f_2(x_1, x_2)\| < 1$ holds true everywhere except for $-0.34 < x_2 < 0.54$, approximately. Now, it is easy to see that the region $R_+ = \{(x_1, x_2) \in [-1, 1]^2 : x_2 > 0.54\}$ is positively invariant for both maps f_1 and f_2. Analogously, the region $R_- = \{(x_1, x_2) \in [-1, 1]^2 : x_2 < -0.54\}$ is positively invariant for both maps f_1 and f_2. Hence, both regions R_+ and R_- result to be convex $\mathcal{U}$-positively invariant compact sets of the nonautonomous system where contraction occurs. Therefore, Theorem 5.9 ensures that inside the set R_+, there exists a UAES and, similarly, there exists another UAES inside R_-.

For the simulation in Figure 25, we generated an input sequence with same probability to occur either u_1 or u_2. In the top panel of Figure 25, we show the UAESs of such input-driven system. In the bottom panel, time series of the observable $\dfrac{x_1 + x_2}{2}$ of many initial conditions are shown. As can be seen, a UAES is substantially different from a fixed point due to its input-driven nature.

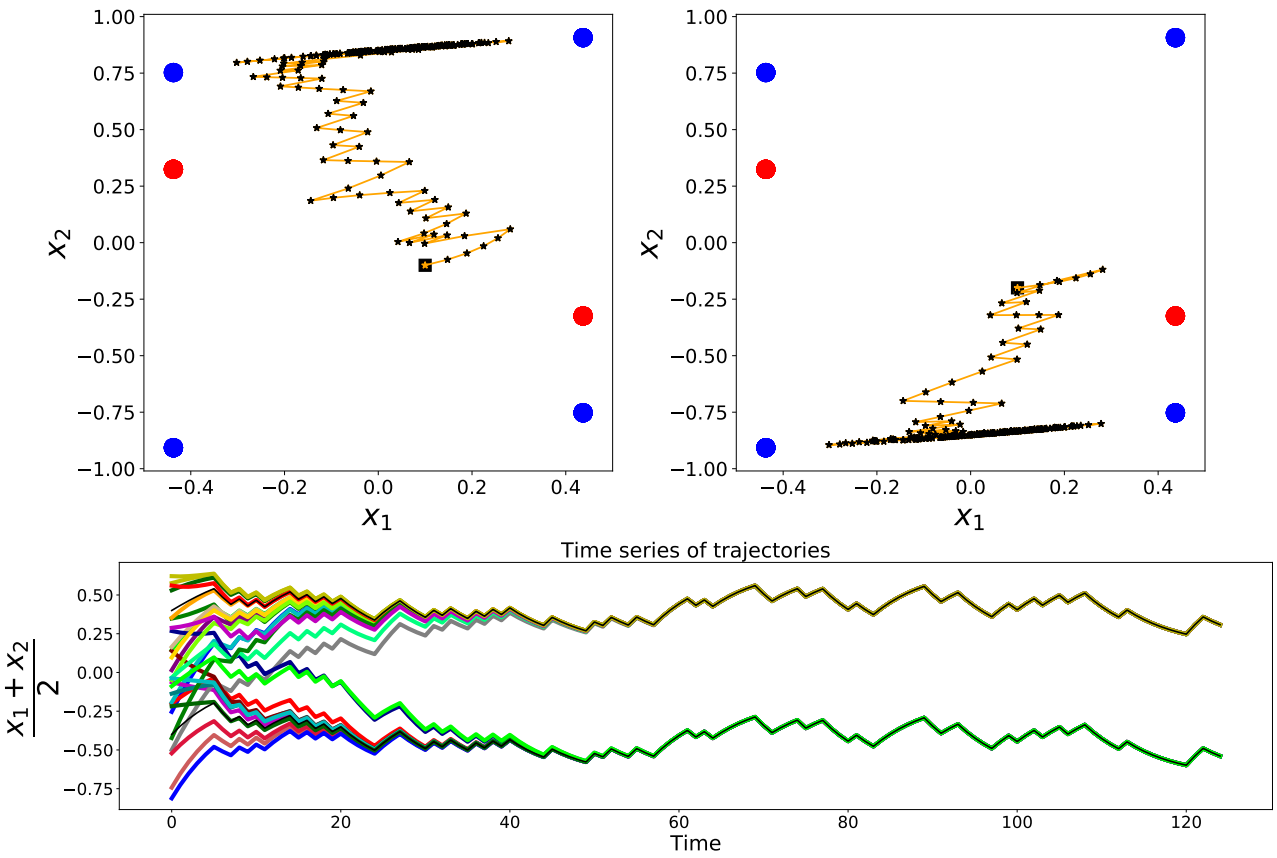


Figure 25: All pictures refer to the same input sequence $\mathbf{v} \in \{u_1, u_2\}^{\mathbb{Z}}$ which has been randomly generated with equal probability to switch between u_1 and u_2. **Top:** Fixed points of both (autonomous) component maps $f_1(x), f_2(x)$ are showed as blue (stable nodes) and red (saddle) dots, respectively. Notably, fixed points vertically aligned on the positive side of the x_1 variable characterise the f_1 map, while the ones on the negative side characterise f_2. On the left panel, the initial condition of coordinates $(0.1, -0.1)$ evolves towards the upper UAES solution; on the right panel, the initial condition of coordinates $(0.1, -0.2)$ evolves towards the bottom UAES solution. Under the same driving input sequence, almost every initial condition converges to one of those two nonautonomous attractors. **Bottom:** Time series of the observable $\frac{x_1+x_2}{2}$ of 30 trajectories starting from randomly chosen initial conditions are shown. This plot suggests a decomposition for the phase space in 2 UAESs, which leads the system to synchronise around two stable responses when driven by the given input sequence.

The separatrix entire solutions The basin boundary of the two UAESs is, at each fixed time step, a horizontal line which lies in-between the stable manifolds of the two saddles of the component maps. In the top left of Figure 26, this line is shown for time step $k = 0$ for the particular input sequence used in simulation of Figure 25. In the top right plot of Figure 26, we evolved many different initial conditions in a neighbourhood of such a horizontal line. We randomly chose one of these trajectories as reference trajectory and computed the distances, at each time step, between each trajectory with respect to a randomly chosen one. It is numerically observed that the closer we start to such horizontal line, the more time is needed to converge towards one of the UAESs. Indeed, the uniform convergence (i.e. in the Hausdorff semi-metric sense) [31, 60] towards a UAES holds only excluding a neighbourhood of the boundary of its basin of attraction. Moreover, from the top right plot of Figure 26 we observe that each initial condition seems to be attracted for a few time steps to a special trajectory (all the distances go to zero), but are eventually pushed away from it in the long run towards one of the two UAESs. This suggests the existence of a third attractive entire solution whose basin of attraction is exactly the (moving in time) separatrix line. As a matter of fact, each point of such separatrix line represents a fibre of an entire solution. Interestingly, all those entire solutions seem to converge towards a unique solution which appear as in bottom pictures of Figure 26. Note that, in particular, such nonautonomous system admits an infinite number of entire solutions but only two of them essentially characterise its dynamics[15]. Although this third attractive entire solution plays a key role in the nonautonomous dynamics of the system, it should not be considered equivalent to the other two UAESs. In fact, the probability to pick up an initial condition in phase space converging to this particular entire solution is null. This is the reason why we require for a UAES to have a neighbourhood of attraction, as formally stated in Definition 5.9, which in particular implies that its basin of attraction has positive Lesbegue measure. In the bottom panels of Figure 26, we tried to detect the third attractive entire solution letting evolve initial conditions extremely close to the separatrix line. Such a special entire solution appears unstable and it traces an orbit that wanders erratically in the region between the stable manifolds of the two saddles of f_1 and f_2.

[15]Note that all the infinitely many separatrix entire solutions are part of the global pullback attractor of Definition 5.5. Moreover, all the entire solutions forward converging to the two UAESs and originating in the infinite past from the separatrix solutions are also them part of the global pullback attractor of Definition 5.5.

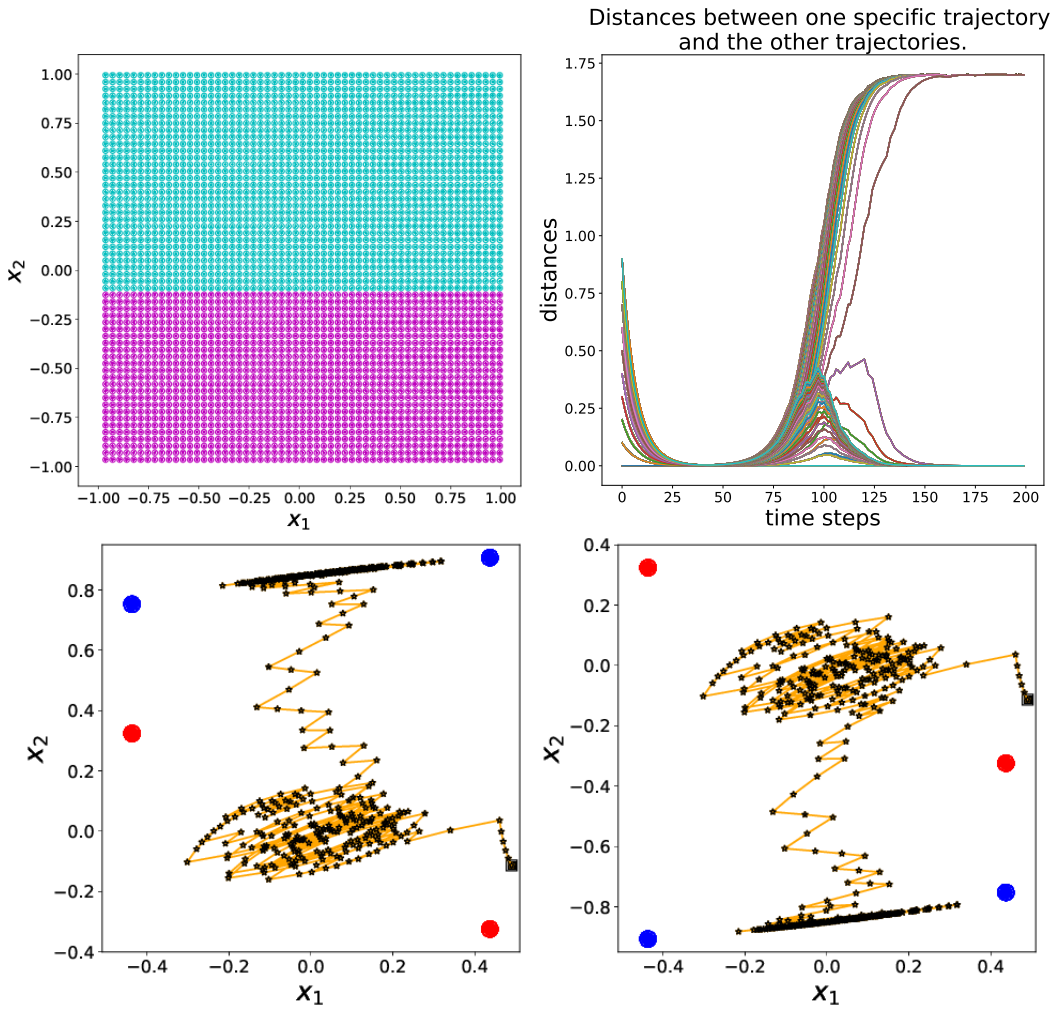


Figure 26: All pictures refer to the same randomly generated input sequence of Figure 25. **Top left:** computation of the basins of attraction of the two UAESs at time step $k = 0$. **Top right:** 950 initial conditions have been chosen around the position of the separatrix at time step $k = 0$; the x_2 coordinate starting from -0.113248 and cumulatively increasing of 10^{-6} until the value -0.113198, the x_1 coordinate starting from -1 and cumulatively increasing of 10^{-1} until the value 1. The trajectory starting from the initial condition $(0, -0.113218)$ has been chosen as reference. At each time step, it has been computed and plotted the distance between the reference trajectory and all other trajectories. **Bottom left:** initial condition is $(0.49, -0.11322366651)$. The trajectory rambles in phase space following the separatrix attractive entire solution, then after about 230 time steps it starts to converge towards the upper UAES. **Bottom right:** here the initial condition is $(0.49, -0.11322366652)$, hence it differs of 10^{-11} from the one in the left picture. The trajectory practically coincides with the one in the left picture but eventually converges towards the bottom UAES.

5.5.2 An example with input-dependent echo index

In this section, we consider a one-dimensional RNN and show how the echo index may vary according to the specific input sequence driving the dynamics.

Let us consider the input-driven system $x[k+1] = G(u[k+1], x[k])$, with

$$G(u, x) = \tanh(1.01x + wu), \qquad x \in [-1, 1], \quad u \in [-1, 1], \tag{112}$$

where w is a positive real value playing the input gain role. Note that in this example the value 1.01 represents the maximum singular value (and spectral radius as well) of a one-dimensional recurrent layer. Therefore, the autonomous system defined by the map $F(x) := G(0, x)$ is expanding around the (unstable) fixed point $x = 0$ and there exist two (uniformly attracting) stable points $x_1^* \approx -0.17, x_2^* \approx 0.17$.

Theorem 5.11 and Proposition 5.14 suggest that the echo index is also determined by the amplitude of the inputs driving the dynamics. Here, we show how modulating the input amplitude via w yields different values of echo index. Note that, as w scales the inputs provided to the system (112), in practice we force the autonomous map F with inputs assuming values in $[-w, w]$. It is possible to analytically compute the value of w such that a fold bifurcation occurs in the system (112), which is approximately $w_c \approx 0.0007$; see Eq. 36 of [20] for details. This implies that perturbing the autonomous dynamics by means of any input sequence with amplitude less than w_c will induce an input-driven system with echo index 2.

To show evidence of this, we generated an input sequence according to a uniform distribution in $[-1, 1]$, then we scaled it by means of three values of $w = 0.0006, 0.01, 0.05$. Figure 27 shows the evolution of ten initial conditions of the system for the three values of w. In the first case of $w = 0.0006$ (top Figure 27), the input-driven system exhibited echo index 2, in accordance with the fact that the critical value w_c is greater than $w = 0.0006$. The second case of $w = 0.01$ (centre Figure 27) produced an interesting dynamics with echo index 1: the unique UAES is characterised by a switching behaviour between the "ghosts" of the two randomly perturbed autonomous stable solutions of F. Finally, in the third case of $w = 0.05$ (bottom Figure 27) the nonautonomous dynamics converge toward a unique UAES (i.e. echo index 1), which wanders randomly around the origin.

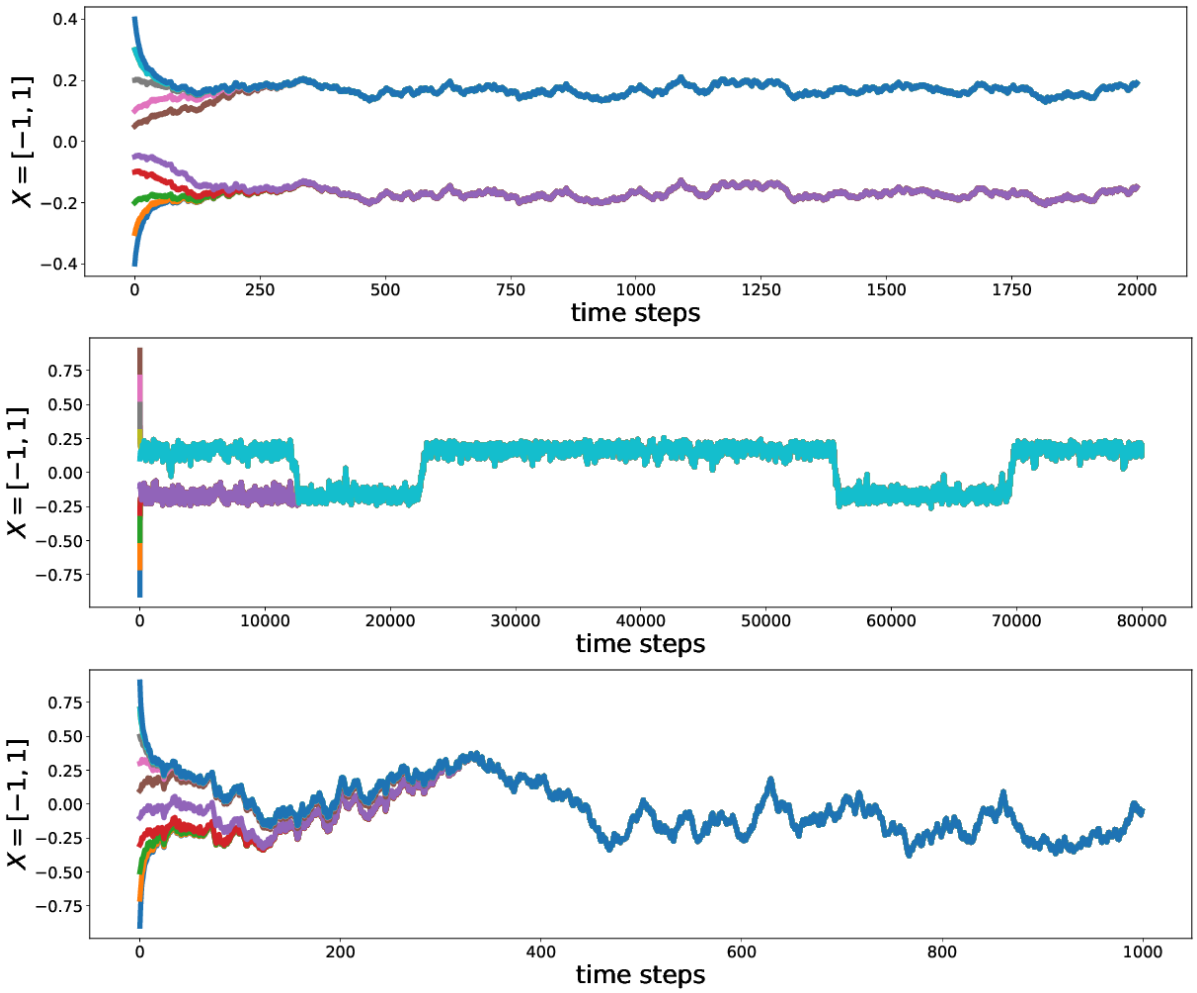


Figure 27: Nonautonomous dynamics of the system (112) driven by an input sequence generated according to a uniform distribution in $[-1, 1]$ and then scaled by means of a positive real parameter w, for three values of the parameter $w = 0.0006, 0.01, 0.05$. Ten initial conditions have been run in all three cases, represented with different colours in the plots. **Top:** case of $w = 0.0006$. The input driven system exhibits echo index 2, i.e. there are two UAESs. **Centre:** case of $w = 0.01$. The input driven system has echo index 1. There exists a unique UAES whose behaviour is affected by the vicinity of the fold bifurcation of the underlying autonomous map $F(x) = G(0, x)$. The resulting dynamics manifest a switching motion. **Bottom:** case of $w = 0.05$. The input-driven system presents echo index 1 and the corresponding uniformly attracting entire solution does not exhibit any switching behaviour.

5.5.3 RNNs dynamics in a context-dependent task

Here, we train an RNN to solve the task described in [46, Figure 5], which consists of performing some context-dependent computation. Our results suggest that the RNN dynamics are characterised by a decomposition in two UAESs, i.e. the trained RNN has echo index 2.

The task is the following. The RNN dynamics is driven by a bi-dimensional input sequence, $\mathbf{u} = \{(u_1[k], u_2[k])\}$, and produces bi-dimensional output $\mathbf{z} = \{(z_1[k], z_2[k])\}$. The behaviour of the input-driven RNN dynamics is controlled by two impulsive control inputs, $u_3[k], u_4[k]$, which do not contribute in driving the dynamics but give the RNN a context of the "on" and "off" type. These instantaneous pulses have unitary amplitude and occur with probability 0.01, i.e. $u_3[k], u_4[k]$ are null most of the time and occur, on average, every 100 time steps. The first readout is used to produce an output $z_1[k]$ which maintains a working memory of the context: assuming the value $+1$ for the "on" state and -1 for the "off" state. For this reason, the first output $z_1[k]$ is fed back into the network via the output feedback connections, W_{fb}. The two input sequences driving the dynamics, $u_1[k], u_2[k]$, are two independently generated time-varying signals obtained through the discrete convolution[16] of a uniformly distributed signal assuming values in $[0, 1)$ with the smooth exponential filter $g(s) = \exp(-s/50)$. Bias values of, respectively, 0.3 and 0.15 were added to these convolutions; input sequences $u_1[k], u_2[k]$ are normalised so their maximum value is one. The second readout has to learn the routing of $u_1[k]$ to output $z_2[k]$ if the network is in state "on", and $u_2[k]$ to output $z_2[k]$ if the network is in state "off".

In Figure 28 we report the results of a trained RNN with $N_r = 500$ neurons; in Appendix C, we provide details regarding how we trained a RNN of the form (8)-(9) to solve this task. In the top two plots of Figure 28, the outputs $z_1[k], z_2[k]$ produced by the RNN versus the target signals are shown, demonstrating that the network learned to solve the task quite well. The three plots in the middle of Figure 28 show the evolution of the test trajectory projected in the two-dimensional subspace of the RNN phase space obtained by computing the first two principal components of the state trajectory. In the two plots at the bottom of Figure 28, we switch off the control inputs and run 100 initial conditions uniformly distributed in phase space. These last two plots show the presence of two UAESs describing the behaviour of the RNN trained to solve such a context-dependent task.

[16] Given two sequences $\{a[k]\}_{k\in\mathbb{Z}}$ and $\{b[k]\}_{k\in\mathbb{Z}}$, the discrete convolution is defined as $(a*b)[n] = \sum_{m=-\infty}^{\infty} a[m]b[n-m]$.

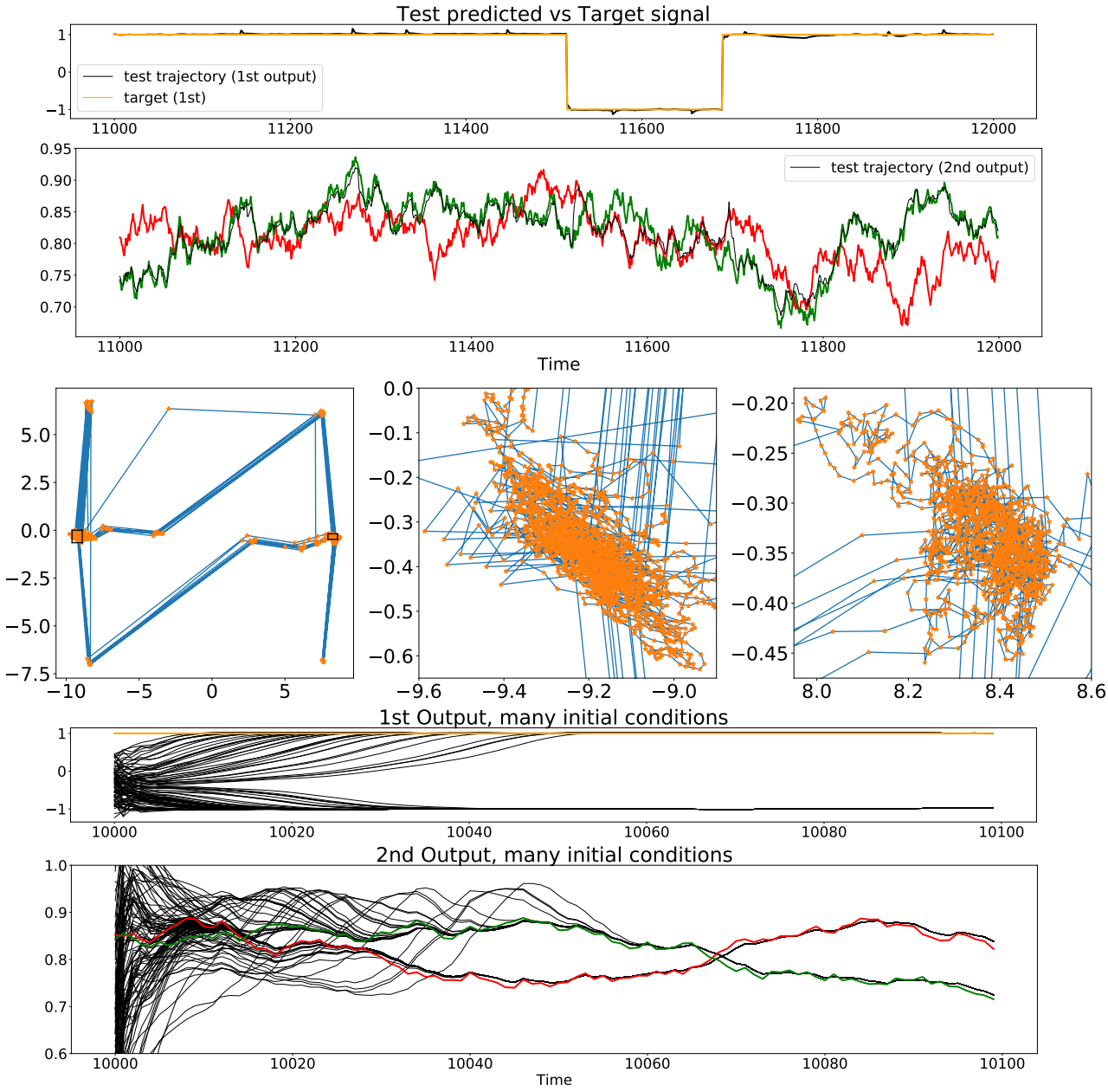


Figure 28: All plots refer to the test session, after training is completed. The internal context-state to be learned is shown in orange and assumes values $+1$ for the "on" state or -1 for the "off" state, while the inputs $u_1[k], u_2[k]$, driving the dynamics are shown in green and red, respectively. A detailed caption for all the pictures of this panel is provided in the next page.

Figure 28: **Top:** the top two plots show the outputs $z_1[k], z_2[k]$ (in black) versus the target for the first output (in orange) and for the second output (in green when the state is "on", and in red when the state is "off"). **Middle:** the left middle plot depicts the evolution of the test trajectory projected in the two-dimensional space spanned by the two principal components of the network states. The cumulative variance of the first two principal components is 0.98. The test trajectory spends most of the time around two locations delimited in the figure by black frames. The other two plots in the middle show a zoomed view of the test trajectory inside of these black frames. The attractor on the left corresponds to the "on" state and it is related to the green signal, while the attractor on the right corresponds to the "off" state and it is related to the red signal. While these two nonautonomous attractors look very complex and unstable, they are in fact very stable to variations of initial conditions, as demonstrated in the bottom plots. **Bottom:** Evolution of the trained RNN dynamics with 100 initial conditions randomly distributed in phase space. All initial conditions converge either to the left or to the right nonautonomous attractor. This experiment strongly suggests that the RNN training produced a phase space decomposition in two UAESs, which in turn are used to solve the context-dependent task.

6 Conclusion

In chapters 3-4, we presented a novel methodology for modeling and interpreting the behaviour of RNNs driven by inputs. In order to obtain a mechanistic model describing how RNNs solve tasks, we exploited the theoretical framework offered by excitable network attractors [5], which are defined as networks of stable fixed points connected by excitable connections. We introduced a procedure to extract excitable network attractors directly from a trajectory generated by a trained RNN. Such a procedure is composed of two main steps: first, fixed points are computed by solving a non-linear optimisation problem [107] and successively excitable connections, with related thresholds, are determined by simulating the dynamics of the autonomous system.

We validated our theoretical developments by considering ESNs trained on the flip-flop task, a simple yet relevant benchmark that consists of learning a prescribed number of stable states and related switching patterns guided by control inputs. We cannot see any particular theoretical limitations in the application of our framework to more complex RNN architectures, including those belonging to the family of Long Short-Term Memory networks, as well as for more complicated dynamical tasks, although a detailed computation of bifurcation behaviour may become unfeasible. Simulation results provide several interesting insights on how RNNs solve tasks and highlight the usefulness of excitable network attractors in describing how RNN undertake computations. We trained echo state networks by means of ridge regression. Our results (not shown here) suggest that the regularisation parameter has a direct impact on the number of attracting regions in phase space generated through training: using too low values produces under-regularised models with a large number of attractors. An interesting future perspective consists of studying the impact of different training mechanisms (e.g., via FORCE learning or similar online approaches) on the resulting network attractor.

We believe that the proposed modelling framework based on excitable network attractors will be suitable to describe the RNN behaviour for many if not all tasks requiring the learning of a number of attractors (which need not be stable fixed points) and related switching patterns. To this end, in future we will also examine classification tasks. As already mentioned, network attractors can in principle be constructed between any type of invariant sets, including limit cycles and chaotic attractors [32, 49]. Our focus on fixed points was mainly dictated by the fact that computing fixed points from a trajectory is significantly easier than computing limit cycles, for instance.

Clearly, fixed points are not powerful enough to accurately model all possible behaviours in phase space. Therefore, an interesting future perspective consists in extending our modeling framework to handle networks composed of heterogeneous attractors, such as a network in phase space connecting fixed points and limit cycles. In turn, this will allow modeling more complex RNN behaviour. Notably, excitable network attractors may provide an useful tool to investigate multifunctionality in reservoir computers [32]. Multifunctionality refers to the ability of a system to behave differently according to the context given by an external input. Thus, for the same trained reservoir computer, it corresponds to the ability to solve more than one task when driven with different input signals.

Another pioneering future direction would be to exploit the extracted excitable network attractor to reverse engineer a set of few ordinary differential equations [4] describing the RNN high dimensional dynamics for the task under consideration.

Furthermore we can imagine to interpret the behaviour of a RNN solving a task through a sequence of graphs. Such a sequence of graphs is meant to be the time-varying, effective excitable network attractor ruling the underlying dynamics. Although in this thesis we mainly considered the dynamics of a trained RNNs, it might be interesting to investigate on the learning dynamics occurring during the training session by means of such a sequence of graph. Qualitative changes in the time-varying graph imply key moments in the learning process for the RNN.

In chapter 5, motivated by the need to go beyond the framework of autonomous dynamical systems theory, we investigated RNN dynamics from a properly time-varying input perspective. Particular attention has been given to the echo state property, since it is a guiding principle that, however, is still not completely understood in the reservoir computing community. Within the framework of nonautonomous dynamical systems theory, we introduced a suitable definition of attractor for input-driven RNNs that we call a uniformly attracting entire solution. This allowed us to formally model and characterise responses of RNNs to input sequences. To the same extent pulse-driven RNNs achieve computations exploiting many autonomous attractors (e.g. in the flip-flop task of chapter 4) we showed that RNNs driven by generic time-varying input sequences could exploit many nonautonomous attractors carrying out reliable computations. In this sense, breaking the echo state property might be functional in some tasks where more than one stable responses are needed. The presence of more than one stable response indicates the possibility to observe and exploit multiple, yet consistent behaviours of a RNN driven by an input sequence (for example,

the context-dependent task shown in Section 5.5.3 requires learning two stable responses). On the other hand, echo index greater than one might also indicate incorrect training and hence signal a possible malfunctioning on a task requiring a unique behaviour in phase space.

Among the original contributions of this thesis there are a number of theoretical results such as theorems 5.9, 5.11, proposition 5.14 and theorem 5.15, concerning the existence, uniqueness and stability of solutions for a generic input-driven RNN. Moreover, we highlight how the echo state property, which guarantees the existence of a unique (stable) response to an input sequence, may be generalised so that only local behaviour in phase space is taken into account. Accordingly, we show how a RNN might reliably produce several stable responses to an input sequence: the echo index introduced here counts such stable responses.

Despite these promising results, many questions are left unsolved. An important feature that we did not consider is the interplay between the time scale of the input driving the dynamics and the internal input-free time scales of the RNN. It is reasonable that few time steps spent outside the regions of uniform contraction will not compromise the stability of the input-driven dynamics. Therefore, a small expansion in phase space is not necessarily detrimental to the RNN dynamics, and so, for example, the hypotheses of theorem 5.9 may be sufficient but are not necessary for the stated conclusion. On the contrary, it rather might add the complexity that is necessary to deal with more challenging tasks. This relates to another intriguing principle regarding complex systems, the so-called *edge of chaos*. It has been argued that complex systems evolve themselves towards the boundary region between ordered dynamics (contraction) and chaotic dynamics (expansion). Such a region has been proved to be the locus of parameters where RNNs reach the best performance [11, 25, 29, 65]. We believe our developments highlight the need to rectify what is commonly misunderstood as unreliable behaviour of driven RNNs with the occurrence of multistable, nonautonomous dynamics. This paradigm shift will ultimately lead to a more suitable definition of "chaotic behaviour" for driven RNNs and related onset mechanisms.

Another interesting open question related to chapter 5 is given by the understanding of the mechanisms regulating a change of echo index. From a purely mathematical perspective, the echo index we formulated lends itself as an interesting number characterising the input-driven dynamics, and as such an intriguing research line might consist of investigating the mechanisms leading to changes of this number due to both variations of the model parameters and variations of the input

sequences driving the dynamics. Particularly original would be to consider changes of the echo index w.r.t. input sequence variations, since usually qualitative changes are detected only as a function of the parameters of the model, but as we highlighted the input sequence plays an active role in the determination of the input-driven RNN dynamics.

In conclusion, we believe that the notions and results introduced in this thesis will prove fundamental for the more general and ambitious goal of providing mechanistic models describing the behaviour of RNNs in a variety of machine learning tasks, such as time series classification (for example inferring which labels will be given to unseen samples to classify prior to computation) and forecasting (for example providing tools to estimate the temporal window of reliability of the forecast), and in general to help individuate which models have learned potentially wrong behaviours.

References

[1] Abbott, L. F. (1999). Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin*, 50(5-6):303–304.

[2] Aljadeff, J., Renfrew, D., Vegué, M., and Sharpee, T. O. (2016). Low-dimensional dynamics of structured random networks. *Physical Review E*, 93(2):022302.

[3] Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Pérez, J. M., and Perona, I. (2013). An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243–256.

[4] Ashwin, P. and Postlethwaite, C. (2013). On designing heteroclinic networks from graphs. *Physica D: Nonlinear Phenomena*, 265:26–39.

[5] Ashwin, P. and Postlethwaite, C. (2016). Designing heteroclinic and excitable networks in phase space using two populations of coupled cells. *Journal of Nonlinear Science*, 26(2):345–364.

[6] Ashwin, P. and Postlethwaite, C. (2018). Sensitive finite-state computations using a distributed network with a noisy network attractor. *IEEE transactions on neural networks and learning systems*, 29(12):5847–5858.

[7] Ashwin, P. and Postlethwaite, C. M. (2020). Excitable networks for finite state computation with continuous time recurrent neural networks. *arXiv preprint arXiv:2012.04129*.

[8] Barak, O. (2017). Recurrent neural networks as versatile tools of neuroscience research. *Current Opinion in Neurobiology*, 46:1–6.

[9] Beer, R. D. (1995). On the dynamics of small continuous-time recurrent neural networks. *Adaptive Behavior*, 3(4):469–509.

[10] Beer, R. D. (2006). Parameter space structure of continuous-time recurrent neural networks. *Neural Computation*, 18(12):3009–3051.

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

[12] Bianchi, F. M., Livi, L., and Alippi, C. (2018). Investigating echo state networks dynamics by means of recurrence analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 29(2):427–439.

[13] Bianchi, F. M., Maiorino, E., Kampffmeyer, M. C., Rizzi, A., and Jenssen, R. (2017). *Recurrent Neural Networks for Short-Term Load Forecasting: An Overview and Comparative Analysis*. Springer.

[14] Bianchi, F. M., Scardapane, S., Uncini, A., Rizzi, A., and Sadeghian, A. (2015). Prediction of telephone calls load using echo state network with exogenous variables. *Neural Networks*, 71:204–213.

[15] Buonomano, D. V. and Maass, W. (2009). State-dependent computations: spatiotemporal processing in cortical networks. *Nature Reviews Neuroscience*, 10(2):113–125.

[16] Caraballo, T., Jara, J. C., Langa, J. A., and Liu, Z. (2013). Morse decomposition of attractors for non-autonomous dynamical systems. *Advanced Nonlinear Studies*, 13(2):309–329.

[17] Carroll, T. L. (2018). Using reservoir computers to distinguish chaotic signals. *Physical Review E*, 98:052209.

[18] Castelvecchi, D. (2016). Can we open the black box of AI? *Nature News*, 538(7623):20.

[19] Cencini, M., Cecconi, F., and Vulpiani, A. (2010). *Chaos: From Simple Models to Complex Systems*. World Scientific, Singapore.

[20] Ceni, A., Ashwin, P., and Livi, L. (2020a). Interpreting recurrent neural networks behaviour via excitable network attractors. *Cognitive Computation*, 12(2):330–356.

[21] Ceni, A., Ashwin, P., Livi, L., and Postlethwaite, C. (2020b). The echo index and multistability in input-driven recurrent neural networks. *Physica D: Nonlinear Phenomena*, 412:132609.

[22] Ceni, A., Olmi, S., Torcini, A., and Angulo-Garcia, D. (2020c). Cross frequency coupling in next generation inhibitory neural mass models. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(5):053121.

[23] Cheng, C.-Y., Lin, K.-H., and Shih, C.-W. (2006). Multistability in recurrent neural networks. *SIAM Journal on Applied Mathematics*, 66(4):1301–1320.

[24] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[25] Cocchi, L., Gollo, L. L., Zalesky, A., and Breakspear, M. (2017). Criticality in the brain: A synthesis of neurobiology, models and cognition. *Progress in neurobiology*, 158:132–152.

[26] Crauel, H., Duc, L. H., and Siegmund, S. (2004). Towards a Morse theory for random dynamical systems. *Stochastics and Dynamics*, 4(3):277–296.

[27] Csáji, B. C. et al. (2001). Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24(48):7.

[28] De Pasquale, B., Cueva, C. J., Rajan, K., Escola, G. S., and Abbott, L. F. (2018). full-FORCE: A target-based method for training recurrent networks. *PLoS ONE*, 13(2):1–18.

[29] Derrida, B. and Pomeau, Y. (1986). Random networks of automata: a simple annealed approximation. *EPL (Europhysics Letters)*, 1(2):45.

[30] Durstewitz, D., Huys, Q. J., and Koppe, G. (2020). Psychiatric illnesses as disorders of network dynamics. *Biological Psychiatry: Cognitive Neuroscience and Neuroimaging*.

[31] Falconer, K. (2004). *Fractal Geometry: Mathematical Foundations and Applications*. John Wiley & Sons, New York, NY.

[32] Flynn, A., Tsachouridis, V. A., and Amann, A. (2021). Multifunctionality in a reservoir computer. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(1):013125.

[33] Funahashi, K. and Nakamura, Y. (1993). Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801–806.

[34] Gallicchio, C. and Micheli, A. (2017). Echo state property of deep reservoir computing networks. *Cognitive Computation*, 9(3):337–350.

[35] Gerstner, W. and Kistler, W. M. (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press.

[36] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

[37] Golub, M. D. and Sussillo, D. (2018). Fixedpointfinder: A tensorflow toolbox for identifying and characterizing fixed points in recurrent neural networks. *Journal of Open Source Software*, 3(31):1003.

[38] Goodman, B. and Flaxman, S. (2016). European union regulations on algorithmic decision-making and a "right to explanation". *arXiv preprint arXiv:1606.08813*.

[39] Graves, A., Mohamed, A.-R., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, Vancouver, BC, Canada. IEEE.

[40] Grigoryeva, L. and Ortega, J.-P. (2018). Echo state networks are universal. *Neural Networks*, 108:495–508.

[41] Grigoryeva, L. and Ortega, J.-P. (2019). Differentiable reservoir computing. *Journal of Machine Learning Research*, 20(179):1–62.

[42] Hammer, B. (2000). On the approximation capability of recurrent neural networks. *Neurocomputing*, 31(1-4):107–123.

[43] Haschke, R. and Steil, J. J. (2005). Input space bifurcation manifolds of recurrent neural networks. *Neurocomputing*, 64:25–38.

[44] Haykin, S. (2008). *Neural Networks and Learning Machines*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

[45] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[46] Hoerzer, G. M., Legenstein, R., and Maass, W. (2012). Emergence of complex computational structures from chaotic neural networks through reward-modulated Hebbian learning. *Cerebral Cortex*, 24(3):677–690.

[47] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.

[48] Ibáñez-Soria, D., Garcia-Ojalvo, J., Soria-Frisch, A., and Ruffini, G. (2018). Detection of generalized synchronization using echo state networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(3):033118.

[49] Inoue, K., Nakajima, K., and Kuniyoshi, Y. (2020). Designing spontaneous behavioral switching via chaotic itinerancy. *Science advances*, 6(46):eabb3989.

[50] Itoh, S. (1979). Random fixed point theorems with an application to random differential equations in Banach spaces. *Journal of Mathematical Analysis and Applications*, 67(2):261–273.

[51] Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *German National Research Center for Information Technology GMD Technical Report*, 148(34):13.

[52] Jaeger, H. and Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80.

[53] Jaeger, H., Lukoševičius, M., Popovici, D., and Siewert, U. (2007). Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352.

[54] Kanai, S., Fujiwara, Y., and Iwamura, S. (2017). Preventing gradient explosions in gated recurrent units. In *Advances in Neural Information Processing Systems*, pages 435–444.

[55] Karpatne, A., Atluri, G., Faghmous, J. H., Steinbach, M., Banerjee, A., Ganguly, A., Shekhar, S., Samatova, N., and Kumar, V. (2017). Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2318–2331.

[56] Katz, G. E. and Reggia, J. A. (2017). Using directional fibers to locate fixed points of recurrent neural networks. *IEEE transactions on neural networks and learning systems*, 29(8):3636–3646.

[57] Keuninckx, L., Danckaert, J., and Van der Sande, G. (2017). Real-time audio processing with a cascade of discrete-time delay line-based reservoir computers. *Cognitive Computation*, 9(3):315–326.

[58] Kidger, P. and Lyons, T. (2019). Universal approximation with deep narrow networks. *arXiv preprint arXiv:1905.08539*.

[59] Kloeden, P. E., Pötzsche, C., and Rasmussen, M. (2012). Limitations of pullback attractors for processes. *Journal of Difference Equations and Applications*, 18(4):693–701.

[60] Kloeden, P. E. and Rasmussen, M. (2011). *Nonautonomous Dynamical Systems*. Number 176. American Mathematical Soc.

[61] Kloeden, P. E. and Yang, M. (2016). Forward attraction in nonautonomous difference equations. *Journal of Difference Equations and Applications*, 22(8):1027–1039.

[62] Koryakin, D., Lohmann, J., and Butz, M. V. (2012). Balanced echo state networks. *Neural Networks*, 36:35–45.

[63] Kuznetsov, Y. A. (2013). *Elements of Applied Bifurcation Theory*, volume 112. Springer Science & Business Media, Berlin, Germany.

[64] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[65] Legenstein, R. and Maass, W. (2007). Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, 20(3):323–334.

[66] Li, J., Waegeman, T., Schrauwen, B., and Jaeger, H. (2014). Frequency modulation of large oscillatory neural networks. *Biological Cybernetics*, 108(2):145–157.

[67] Livi, L., Bianchi, F. M., and Alippi, C. (2018). Determination of the edge of criticality in echo state networks through Fisher information maximization. *IEEE Transactions on Neural Networks and Learning Systems*, 29(3):706–717.

[68] Løkse, S., Bianchi, F. M., and Jenssen, R. (2017). Training echo state networks with regularization through dimensionality reduction. *Cognitive Computation*, 9(3):364–378.

[69] Lu, Z., Hunt, B. R., and Ott, E. (2018). Attractor reconstruction by machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(6):061104.

[70] Lukoševičius, M. (2012). *A Practical Guide to Applying Echo State Networks*, pages 659–686. Springer Berlin Heidelberg, Berlin, Heidelberg.

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

[72] Maass, W., Joshi, P., and Sontag, E. D. (2007). Computational aspects of feedback in neural circuits. *PLoS Computational Biology*, 3(1):e165.

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

[74] Manjunath, G. and Jaeger, H. (2013). Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks. *Neural Computation*, 25(3):671–696.

[75] Manjunath, G., Tino, P., and Jaeger, H. (2012). Theory of input driven dynamical systems. *dice. ucl. ac. be, number April*, pages 25–27.

[76] Massar, M. and Massar, S. (2013). Mean-field theory of echo state networks. *Physical Review E*, 87(4):042809.

[77] Mastrogiuseppe, F. and Ostojic, S. (2018). Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*.

[78] Mastrogiuseppe, F. and Ostojic, S. (2019). A geometrical analysis of global stability in trained feedback networks. *Neural Computation*, 31(6):1139–1182.

[79] Mayer, N. M. and Yu, Y.-H. (2017). Orthogonal echo state networks and stochastic evaluations of likelihoods. *Cognitive Computation*, 9(3):379–390.

[80] Miller, K. D. and Fumarola, F. (2012). Mathematical equivalence of two common forms of firing rate models of neural networks. *Neural Computation*, 24(1):25–31.

[81] Miller, P. (2016). Itinerancy between attractor states in neural systems. *Current Opinion in Neurobiology*, 40:14–22.

[82] Milnor, J. (1985). On the concept of attractor. *Communications in Mathematical Physics*, 99(2):177–195.

[83] Montavon, G., Samek, W., and Müller, K.-R. (2017). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15.

[84] Neves, F. S., Voit, M., and Timme, M. (2017). Noise-constrained switching times for heteroclinic computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(3):033107.

[85] Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, Berlin, Germany.

[86] Ochs, G. (1999). *Weak Random Attractors*. Issue 449 of Report, Institut für Dynamische Systeme. Univ. of Bremen.

[87] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1310–1318, Atlanta, Georgia, USA.

[88] Pham, V., Bluche, T., Kermorvant, C., and Louradour, J. (2014). Dropout improves recurrent neural networks for handwriting recognition. In *14th International Conference on Frontiers in Handwriting Recognition*, pages 285–290, Crete Island, Greece.

[89] Pisarchik, A. N. and Feudel, U. (2014). Control of multistability. *Physics Reports*, 540(4):167–218.

[90] Pollack, J. B. (1991). The induction of dynamical recognizers. In *Connectionist approaches to language learning*, pages 123–148. Springer.

[91] Pötzsche, C. (2011). Nonautonomous continuation of bounded solutions. *Commun. Pure Appl. Anal*, 10(3):937–961.

[92] Rabinovich, M., Huerta, R., and Laurent, G. (2008). Transient dynamics for neural processing. *Science*, 321(5885):48–50.

[93] Rabinovich, M., Volkovskii, A., Lecanda, P., Huerta, R., Abarbanel, H., and Laurent, G. (2001). Dynamical encoding by networks of competing neuron groups: winnerless competition. *Physical Review Letters*, 87(6):068102.

[94] Rajan, K., Abbott, L. F., and Sompolinsky, H. (2010). Stimulus-dependent suppression of chaos in recurrent neural networks. *Physical Review E*, 82(1):011903.

[95] Reinhart, R. F. and Steil, J. J. (2012). Regularization and stability in reservoir networks with output feedback. *Neurocomputing*, 90:96–105.

[96] Rivkind, A. and Barak, O. (2017a). Local dynamics in trained recurrent neural networks. *Physical review letters*, 118(25):258101.

[97] Rivkind, A. and Barak, O. (2017b). Local dynamics in trained recurrent neural networks. *Physical Review Letters*, 118:258101.

[98] Rodan, A. and Tiňo, P. (2012). Simple deterministically constructed cycle reservoirs with regular jumps. *Neural Computation*, 24(7):1822–1852.

[99] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[100] Scardapane, S. and Uncini, A. (2017). Semi-supervised echo state networks for audio classification. *Cognitive Computation*, 9(1):125–135.

[101] Seoane, L. F. (2019). Evolutionary aspects of reservoir computing. *Philosophical Transactions of the Royal Society B*, 374(1774):20180377.

[102] Siegelmann, H. T. and Sontag, E. D. (1991). Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77–80.

[103] Smart, D. R. (1980). *Fixed Point Theorems*, volume 66. CUP Archive.

[104] Sompolinsky, H., Crisanti, A., and Sommers, H.-J. (1988). Chaos in random neural networks. *Physical Review Letters*, 61(3):259.

[105] Strogatz, S. H. (2014). *Nonlinear Dynamics and Chaos*. Hachette, UK.

[106] Sussillo, D. and Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557.

[107] Sussillo, D. and Barak, O. (2013). Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649.

[108] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

[109] Tabor, W. (2000). Fractal encoding of context-free grammars in connectionist networks. *Expert Systems*, 17(1):41–56.

[110] Tallec, C. and Ollivier, Y. (2018). Can recurrent neural networks warp time? In *International Conference on Learning Representations*.

[111] Tiňo, P., Horne, B. G., and Giles, C. L. (2001). Attractive periodic sets in discrete-time recurrent networks (with emphasis on fixed-point stability and bifurcations in two-neuron networks). *Neural Computation*, 13(6):1379–1414.

[112] Tiňo, P., Horne, B. G., Giles, C. L., and Collingwood, P. C. (1998). Finite state machines and recurrent neural networks—automata and dynamical systems approaches. In *Neural networks and pattern recognition*, pages 171–219. Elsevier.

[113] Tripp, B. P. (2017). Similarities and differences between stimulus tuning in the inferotemporal visual cortex and convolutional networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3551–3560. IEEE.

[114] Tsuda, I. (2015). Chaotic itinerancy and its roles in cognitive neurodynamics. *Current Opinion in Neurobiology*, 31:67–71.

[115] Vincent-Lamarre, P., Lajoie, G., and Thivierge, J.-P. (2016). Driving reservoir models with oscillations: a solution to the extreme structural sensitivity of chaotic networks. *Journal of Computational Neuroscience*, pages 1–18.

[116] Wang, X. and Blum, E. K. (1992). Discrete-time versus continuous-time models of neural networks. *Journal of Computer and System sciences*, 45(1):1–19.

[117] Wang, X. and Blum, E. K. (1995). Dynamics and bifurcation of neural networks. *The Handbook of Brain Theory and Neural Networks*, pages 339–343.

[118] Weinberger, O. and Ashwin, P. (2018). From coupled networks of systems to networks of states in phase space. *Discrete & Continuous Dynamical Systems - B*, 23:2043.

[119] Wisdom, S., Powers, T., Hershey, J., Le Roux, J., and Atlas, L. (2016). Full-capacity unitary recurrent neural networks. In *Advances in neural information processing systems*, pages 4880–4888.

[120] Yildiz, I. B., Jaeger, H., and Kiebel, S. J. (2012). Re-visiting the echo state property. *Neural Networks*, 35:1–9.

[121] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.

A Aggregation of fixed points via clustering

The optimization procedure described in Section 4.3.1 provides a number of solutions equal to the number of initial conditions taken into account (assuming all initial conditions converge); however, such solutions might be similar up to a prescribed numerical precision. In order to reduce the set of all solutions to a few effective fixed points, we run the k-means clustering algorithm and retain only the elements belonging the final clusters minimizing the distance w.r.t. the cluster centroids. Instead of aggregating all solution at once, we first group such solutions according to their linear stability properties as indicated by the spectrum of the Jacobian matrix of the autonomous map (22). In particular, we form the group of linear stable fixed points, the group with one unstable direction and so on. Finally, for each group, we run k-means with parameter k identified according

with the minimum of the Davies-Bouldin index [3].

B Hausdorff distance

Let (X, d_X) be a metric space: for convenience we recall some basic properties of the Hausdorff distance between two subsets of X. For any subset $Y \subset X$, we denote $B_\varepsilon(Y)$ the ε-neighbourhood of the set Y, i.e. $B_\varepsilon(Y) := \bigcup_{y\in Y} B_\varepsilon(y)$ where $B_\varepsilon(y) := \{x \in X \mid d_X(x, y) < \varepsilon\}$ is the open ball of radius ε centred on $y \in Y$.

Definition B.1 *For any pair of nonempty subsets $Y, Z \subset X$, let us define the following function*

$$h(Y, Z) := \sup_{y\in Y} \underbrace{\inf_{z\in Z} d_X(y, z)}_{=:d_X(y,Z)\ \textit{point-set distance}}. \tag{113}$$

We call $h : (\wp(X) \setminus \{\emptyset\})^2 \to [0, +\infty]$ the Hausdorff semi-distance *[31] of the metric space (X, d_X), where $\wp(X)$ is the power set of X.*

Function h is not symmetric, i.e. in general $h(Y, Z) \neq h(Z, Y)$. Unfortunately, the fact that $h(Y, Z) = 0$ does not imply that $Y = Z$. Nevertheless, it holds that $h(Y, Z) = \inf\{\varepsilon \geq 0 \mid Y \subseteq B_\varepsilon(Z)\}$ hence the following is true for all $\varepsilon \geq 0$,

$$h(Y, Z) = \varepsilon \quad \Longrightarrow \quad Y \subseteq \overline{B_\varepsilon(Z)}, \tag{114}$$

where $\overline{B}$ denotes the closure of a set B. Moreover, if the subset Y is bounded then $h(Y, Z) < +\infty$. Therefore, the function

$$H(A, B) := \max\{h(Y, Z)\,,\, h(Z, Y)\} \tag{115}$$

is a metric on the space of all nonempty *compact* subsets of X and we call it the *Hausdorff distance* of the metric space (X, d_X).

C Training details for the context-dependent task in Section 5.5.3

We trained an RNN of the form (8)-(9) with $N_r = 500$ neurons by means of a supervised learning algorithm (ridge regression) which exploits the output feedback as a mechanism for optimising the recurrent layer; see [20] for more details. The state-update (8) was configured with $\alpha = 1$. The

readout function in (9) was parametrised through a matrix W_o so that $\psi(x) = W_o x$. In the training phase, Gaussian noise was added inside the activation function ϕ, see [20, Equation 5], with zero mean and standard deviation set to 0.05. The entries of matrices W_{in}, W_{fb} and W_r were i.i.d. drawn from a uniform distribution in $[-1, 1]$; the sparseness of W_r was set to 95%. Moreover, the matrix W_r was rescaled so that its spectral radius equals to 0.9. Finally, the readout matrix W_o have been determined via ridge regression, with regularisation parameter set to 0.7. We consider input sequences with 15000 time-steps. The first 10000 time-steps have been used for training and the remaining 5000 for testing. Once the training session was completed, we closed the feedback loop by injecting the output $z_1[k]$ into the network state-update.