



# Emulating complex networks with a single delay differential equation

Florian Stelzer<sup>1,2,a</sup> and Serhiy Yanchuk<sup>1</sup>

<sup>1</sup> Institute of Mathematics, Technische Universität Berlin, Berlin, Germany

<sup>2</sup> Department of Mathematics, Humboldt-Universität zu Berlin, Berlin, Germany

Received 27 November 2020 / Accepted 23 April 2021 / Published online 6 June 2021

© The Author(s) 2021

## 1 Introduction

Systems with time-delays, or delay differential equations (DDE), play an important role in modeling various natural phenomena and technological processes [1–8]. In optoelectronics, delays emerge due to finite optical or electric signal propagation time between the elements [9–20]. Similarly, in neuroscience, propagation delays of the action potentials play a crucial role in information processing in the brain [21–28].

Machine Learning is another rapidly developing application area of delay systems [29–47]. It is shown recently that DDEs can successfully realize a reservoir computing setup, theoretically [41–44, 46, 48–50], and implemented in optoelectronic hardware [30, 32, 39]. In time-delay reservoir computing, a single DDE with either one or a few variables is used for building a ring network of coupled maps with fixed internal weights and fixed input weights. In a certain sense, the network structure emerges by properly unfolding the temporal behavior of the DDE. In this paper, we explain how such an unfolding appears, not only for the ring network as in reservoir computing but also for arbitrary networks of coupled maps. In [51], a training method is proposed to modify the input weights while the internal weights are still fixed.

Among the most related previous publications, Hart and collaborators unfold networks with arbitrary topology from delay systems [42, 48]. Our work extends their results in several directions, including varying coupling weights and applying it to a broader class of delay systems. The networks constructed by our method allow for a modulation of weights. Hence, they can be employed in Machine-Learning applications with weight training. In our recent paper [50], we show that a single DDE can emulate a deep neural network and perform various computational tasks successfully. More specifically, the work [50] derives a multilayer neural network from a delay system with modulated feed-

back terms. This neural network is trained by gradient descent using back-propagation and applied to machine-learning tasks.

As follows from the above-mentioned machine-learning applications, delay models can be effectively used for unfolding complex network structures in time. Our goal here is a general description of such networks. While focusing on the network construction, we do not discuss details of specific machine-learning applications such as, e.g., weights training by gradient descent, or specific tasks.

The structure of the paper is as follows. In Sect. 2, we derive a feed-forward network from a DDE with modulated feedback terms. Section 3 describes a recurrent neural network. In Sect. 4, we review a special but practically important case of delay systems with a linear instantaneous part and nonlinear delayed feedback containing an affine combination of the delayed variables; originally, these results have been derived in [50].

## 2 From delay systems to multilayer feed-forward networks

### 2.1 Delay systems with modulated feedback terms

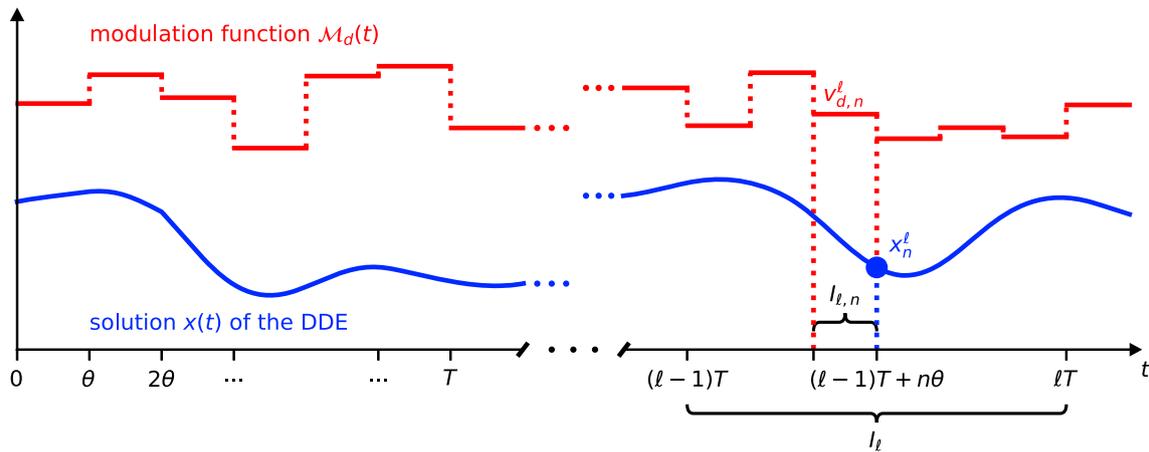
Multiple delays are required for the construction of a network with arbitrary topology by a delay system [42, 48, 50]. In such a network, the connection weights are emulated by a modulation of the delayed feedback signals [50]. Therefore, we consider a DDE of the following form:

$$\dot{x}(t) = f(x(t), z(t), \mathcal{M}_1(t)x(t - \tau_1), \dots, \mathcal{M}_D(t)x(t - \tau_D)), \quad (1)$$

with  $D$  delays  $\tau_1, \dots, \tau_D$ , a nonlinear function  $f$ , a time-dependent driving signal  $z(t)$ , and modulation functions  $\mathcal{M}_1(t), \dots, \mathcal{M}_D(t)$ .

System (1) is a non-autonomous DDE, and the properties of the functions  $\mathcal{M}_d(t)$  and  $z(t)$  play an important role for unfolding a network from (1). To define these

<sup>a</sup> e-mail: [stelzer@math.tu-berlin.de](mailto:stelzer@math.tu-berlin.de) (corresponding author)



**Fig. 1** The clock cycle intervals  $I_\ell$  and sub-intervals  $I_{\ell,n}$ . The node  $x_n^\ell$  (blue dot) is defined by the value of the solution  $x(t)$  of system (1) (blue line) at the time point  $t = (\ell - 1)T + n\theta$ . The modulation function  $\mathcal{M}_d(t)$  is a step function with constant values  $v_{d,n}^\ell$  on the intervals  $I_{\ell,n}$

properties, a time quantity  $T > 0$  is introduced, called the *clock cycle*. Further, we choose a number  $N$  of grid points per  $T$ -interval and define  $\theta := T/N$ . We define the clock cycle intervals

$$I_\ell := ((\ell - 1)T, \ell T], \quad \ell = 1, \dots, L,$$

which we split into smaller sub-intervals

$$I_{\ell,n} := ((\ell - 1)T + (n - 1)\theta, (\ell - 1)T + n\theta], \quad n = 1, \dots, N,$$

see Fig. 1. We assume the following properties for the delays and modulation functions:

Property (I): The delays satisfy  $\tau_d = n_d\theta$ ,  $d = 1, \dots, D$  with natural numbers  $0 < n_1 < \dots < n_D < 2N$ . Consequently, it holds  $0 < \tau_1 < \dots < \tau_D < 2T$ .

Property (II): The functions  $\mathcal{M}_d(t)$  are step functions, which are constant on the intervals  $I_{\ell,n}$ . We denote these constants as  $v_{d,n}^\ell$ , i.e.,

$$\mathcal{M}_d(t) = v_{d,n}^\ell \quad \text{for } t \in I_{\ell,n}.$$

In the following sections, we show that one can consider the intervals  $I_\ell$  as layers with  $N$  nodes of a network arising from the delay system (1) if the modulation functions  $\mathcal{M}_d(t)$  fulfill certain additional requirements. The  $n$ th node of the  $\ell$ -th layer is defined as

$$x_n^\ell := x((\ell - 1)T + n\theta), \quad n = 1, \dots, N, \quad \ell = 1, \dots, L, \tag{2}$$

which corresponds to the solution of the DDE (1) at time point  $(\ell - 1)T + n\theta$ . The solution at later time points  $x_{n'}^{\ell'}$ , with either  $\ell' > \ell$  or  $n' > n$  for  $\ell' = \ell$  depends, in general, on  $x_n^\ell$ , thus, providing the interdependence between the nodes. Such dependence can be found explicitly in some situations. The simplest way is

to use a discretization for small  $\theta$ , and we consider such a case in the following Sect. 2.2. Another case, when  $\theta$  is large, can be found in [50].

Let us remark about the initial state for DDE (1). According to the general theory [2], to solve an initial value problem, an initial history function  $x_0(s)$  must be provided on the interval  $s \in [-\tau_D, 0]$ , where  $\tau_D$  is the maximal delay. In terms of the nodes, one needs to specify  $x_n^\ell$  for  $n_D$  “history” nodes. However, the modulation functions  $\mathcal{M}_d(t)$  can weaken this requirement. For example, if  $\mathcal{M}_d(t) = 0$  for  $t \leq \tau_d$ , then it is sufficient to know the initial state  $x(0) = x_0^1 = x_0$  at a single point, and we do not require a history function at all. In fact, the latter special case has been employed in [50] for various machine-learning tasks.

### 2.2 Disclosing network connections via discretization of the DDE

Here, we consider how a network of coupled maps can be derived from DDE (1). Since the network nodes are already introduced in Sect. 2.1 as  $x_n^\ell$  by Eq. (2), it remains to describe the connections between the nodes. Such links are functional connections between the nodes  $x_n^\ell$ . Hence, our task is to find functional relations (maps) between the nodes.

For simplicity, we restrict ourselves to the Euler discretization scheme since the obtained network topology is independent of the chosen discretization. Similar network constructions by discretization from ordinary differential equations have been employed in [52–54].

We apply a combination of the forward and backward Euler method: the instantaneous system states of (1) are approximated by the left endpoints of the small-step intervals of length  $\theta$  (forward scheme). The driving signal  $z(t)$  and the delayed system states are approximated by the right endpoints of the step intervals (backward scheme). Such an approach leads to sim-

pler expressions. We obtain

$$x_n^\ell = x_{n-1}^\ell + \theta f(x_{n-1}^\ell, z(t_n^\ell), \mathcal{M}_1(t_n^\ell)x(t_n^\ell - \tau_1), \dots, \mathcal{M}_D(t_n^\ell)x(t_n^\ell - \tau_D)) \tag{3}$$

for  $n = 2, \dots, N$ , where  $t_n^\ell := (\ell - 1)T + n\theta$ , and

$$x_1^\ell = x_N^{\ell-1} + \theta f(x_N^{\ell-1}, z(t_1^\ell), \mathcal{M}_1(t_1^\ell)x(t_1^\ell - \tau_1), \dots, \mathcal{M}_D(t_1^\ell)x(t_1^\ell - \tau_D)) \tag{4}$$

for the first node in the  $I_\ell$ -interval.

According to Property (I), the delays satisfy  $0 < \tau_d < 2T$ . Therefore, the delay-induced feedback connections with target in the interval  $I_\ell$  can originate from one of the following intervals:  $I_\ell, I_{\ell-1}$ , or  $I_{\ell-2}$ . In other words: the time points  $t_n^\ell - \tau_d$  can belong to one of these intervals  $I_\ell, I_{\ell-1}, I_{\ell-2}$ . Formally, it can be written as

$$t_n^\ell - \tau_d = t_n^\ell - n_d\theta = \begin{cases} t_{n-n_d}^\ell \in I_\ell, & \text{if } n_d < n, \\ t_{N+n-n_d}^{\ell-1} \in I_{\ell-1}, & \text{if } n \leq n_d < N + n_d, \\ t_{2N+n-n_d}^{\ell-2} \in I_{\ell-2}, & \text{if } N + n \leq n_d. \end{cases} \tag{5}$$

We limit the class of networks to multilayer systems with connections between the neighboring layers. Such networks, see Fig. 4b, are frequently employed in machine-learning tasks, e.g., as deep neural networks [50, 55–58]. Using (5), we can formulate a condition for the modulation functions  $\mathcal{M}_d(t)$  to ensure that the delay terms  $x(t - \tau_d)$  induce only connections between subsequent layers. For this, we set the modulation functions’ values to zero if the originating time point  $t_n^\ell - \tau_d$  of the corresponding delay connection does not belong to the interval  $I_{\ell-1}$ . This leads to the following assumption on the modulation functions:

Property (III): The modulation functions  $\mathcal{M}_d(t)$  vanish at the following intervals:

$$\mathcal{M}_d(t) = v_{d,n}^\ell = 0 \quad \text{for } t \in I_{\ell,n} \quad \text{if } (n_d < n) \quad \text{or } (N + n \leq n_d). \tag{6}$$

In the following, we assume that condition (III) is satisfied.

Expressions (3)–(4) contain the interdependencies between  $x_n^\ell$ , i.e., the connections between the nodes of the network. We explain these dependencies and present them in a more explicit form in the following. Our goal is to obtain the multilayer network shown in Fig. 4b.

### 2.3 Effect of time-delays on the network topology

Taking into account property (III), the node  $x_n^\ell$  of layer  $I_\ell$  receives a connection from a node  $x_{n-n'_d}^{\ell-1}$  of layer  $I_{\ell-1}$ , where  $n'_d := n_d - N$ . Two neighboring layers are illustrated in Fig. 2, where the nodes in each layer

are ordered vertically from top to bottom. Depending on the size of the delay, we can distinguish three cases.

- (a) For  $\tau_d < T$ , there are  $n_d$  “upward” connections as shown in panel Fig. 2a.
- (b) For  $\tau_d = T$ , there are  $n_d = N$  “horizontal” delay-induced connections, i.e. connections from nodes of layer  $\ell - 1$  to nodes of layer  $\ell$  with the same index, see Fig. 2b.
- (c) For larger delays  $\tau_d > T$ , there are  $2N - n_d$  “downward” delay-induced connections, as shown in Fig. 2c.

In all cases, the connections induced by one delay  $\tau_d$  are parallel. Since the delay system possesses multiple delays  $0 < \tau_1 < \dots < \tau_D < 2T$ , the parallel connection patterns overlap, as illustrated in Fig. 4b, leading to a more complex topology. In particular, a fully connected pattern appears for  $D = 2N - 1$  and  $\tau_d = \theta d$ , i.e.  $\tau_1 = \theta, \tau_2 = 2\theta, \dots, \tau_D = D\theta = (2N - 1)\theta$ .

### 2.4 Modulation of connection weights

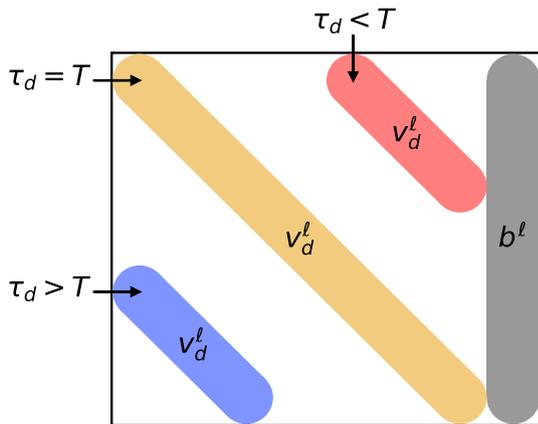
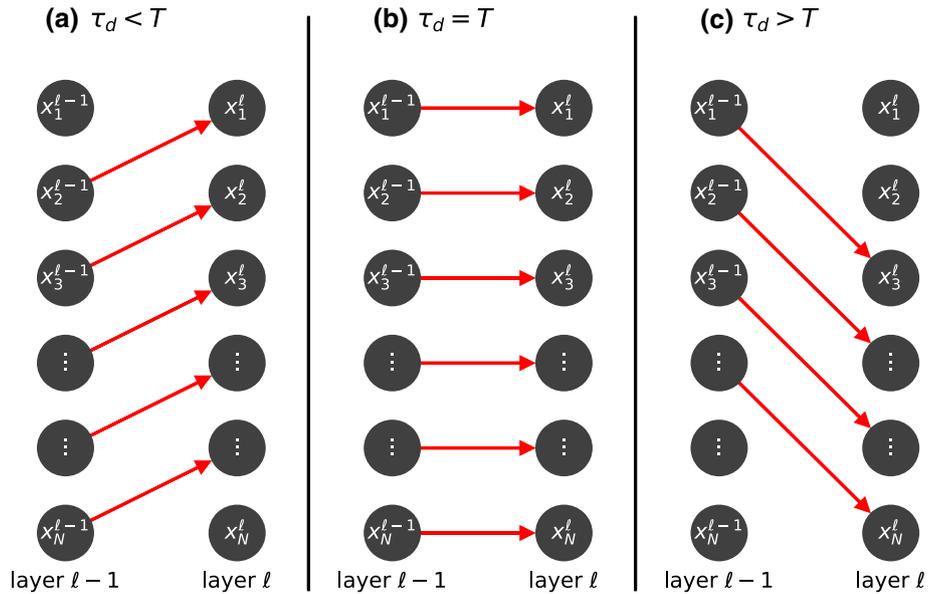
With the modulation functions satisfying property (III), the Euler scheme (3)–(4) simplifies to the following map:

$$x_1^\ell = x_N^{\ell-1} + \theta f(x_N^{\ell-1}, z(t_1^\ell), v_{1,1}^\ell x_{1-n'_1}^{\ell-1}, \dots, v_{D,1}^\ell x_{1-n'_D}^{\ell-1}), \tag{7}$$

$$x_n^\ell = x_{n-1}^\ell + \theta f(x_{n-1}^\ell, z(t_n^\ell), v_{1,n}^\ell x_{n-n'_1}^{\ell-1}, \dots, v_{D,n}^\ell x_{n-n'_D}^{\ell-1}), \quad n = 2, \dots, N, \tag{8}$$

where Eq. (6) implies  $v_{d,n}^\ell = 0$  if  $n - n'_d < 1$  or  $n - n'_d > N$ . In other words, the dependencies at the right-hand side of (7)–(8) contain only the nodes from the  $\ell - 1$ -th layer. Moreover, the numbers  $v_{d,n}^\ell$  determine the strengths of the connections from  $x_{n-n'_d}^{\ell-1}$  to  $x_n^\ell$  and can be considered as network weights. By reindexing, we can define weights  $w_{n,j}^\ell$  connecting node  $j$  of layer  $\ell - 1$  to node  $n$  of layer  $\ell$ . These weights are given by the equation

**Fig. 2** Network connections induced by one time-delay  $\tau_d$ . **a** Connections induced by  $\tau_d < T$ . **b**  $\tau_d = T$ . **c**  $\tau_d > T$ . Multiple delays  $\tau_1, \dots, \tau_D$  result in a superposition of parallel patterns as shown in Fig. 4b



**Fig. 3** Coupling matrix  $W^\ell$  between the hidden layers  $\ell - 1$  and  $\ell$ , see Eq. (9)–(10). The nonzero weights are arranged along the diagonals, and equal  $v_{d,n}^\ell$ . The position of the diagonals is determined by the corresponding delay  $\tau_d$ . If  $\tau_d = T = N\theta$ , then the main diagonal contains the entries  $v_{d,1}^\ell, \dots, v_{d,N}^\ell$  (shown in yellow). If  $\tau_d = n_d\theta < T$ , then the corresponding diagonal lies above the main diagonal and contains the values  $v_{d,1}^\ell, \dots, v_{d,n_d}^\ell$  (red). If  $\tau_d = n_d\theta > T$ , then the corresponding diagonal lies below the main diagonal and contains the values  $v_{d,n_d-N+1}^\ell, \dots, v_{d,N}^\ell$  (blue). The last column of the matrix contains the bias weights (gray)

$$w_{nj}^\ell := \sum_{d=1}^D \delta_{n-n'_d, j} v_{d,n}^\ell = \begin{cases} 0 & \text{if } \forall d: j \neq n - n'_d, \\ v_{d,n}^\ell & \text{if } \exists d: j = n - n'_d, \end{cases} \quad (9)$$

and define the entries of the weight matrix  $W^\ell = (w_{nj}^\ell) \in \mathbb{R}^{N \times (N+1)}$ , except for the last column, which is defined below and contains bias weights. The symbol  $\delta_{nj}$  is the Kronecker delta, i.e.  $\delta_{nj} = 1$  if  $n = j$ , and  $\delta_{nj} = 0$  if  $n \neq j$ .

The time-dependent driving function  $z(t)$  can be utilized to realize a bias weight  $b_n^\ell$  for each node  $x_n^\ell$ . For

details, we refer to Sect. 2.5. We define the last column of the weight matrix  $W^\ell$  by

$$w_{n,N+1}^\ell := b_n^\ell. \quad (10)$$

The weight matrix is illustrated in Fig. 3. This matrix  $W^\ell$  is in general sparse, where the degree of sparsity depends on the number  $D$  of delays. If  $D = 2N - 1$  and  $\tau_d = d\theta$ ,  $d = 1, \dots, D$ , we obtain a dense connection matrix. Moreover, the positions of the nonzero entries and zero entries are the same for all matrices  $W^2, \dots, W^L$ , but the values of the nonzero entries are in general different.

### 2.5 Interpretation as multilayer neural network

The map (7)–(8) can be interpreted as the hidden layer part of a multilayer neural network provided we define suitable input and output layers.

The input layer determines how a given input vector  $u \in \mathbb{R}^{M+1}$  is transformed to the state of the first hidden layer  $x(t)$ ,  $t \in I_1$ . The input  $u \in \mathbb{R}^{M+1}$  contains  $M$  input values  $u_1, \dots, u_M$  and an additional entry  $u_{M+1} = 1$ . To ensure that  $x(t)$ ,  $t \in I_1$  depends on  $u$  and the initial state  $x(0) = x_0$  exclusively, and does not depend on a history function  $x(s)$ ,  $s < 0$ , we set all modulation functions to zero on the first hidden layer interval. This leads to the following

Property (IV): The modulation functions satisfy

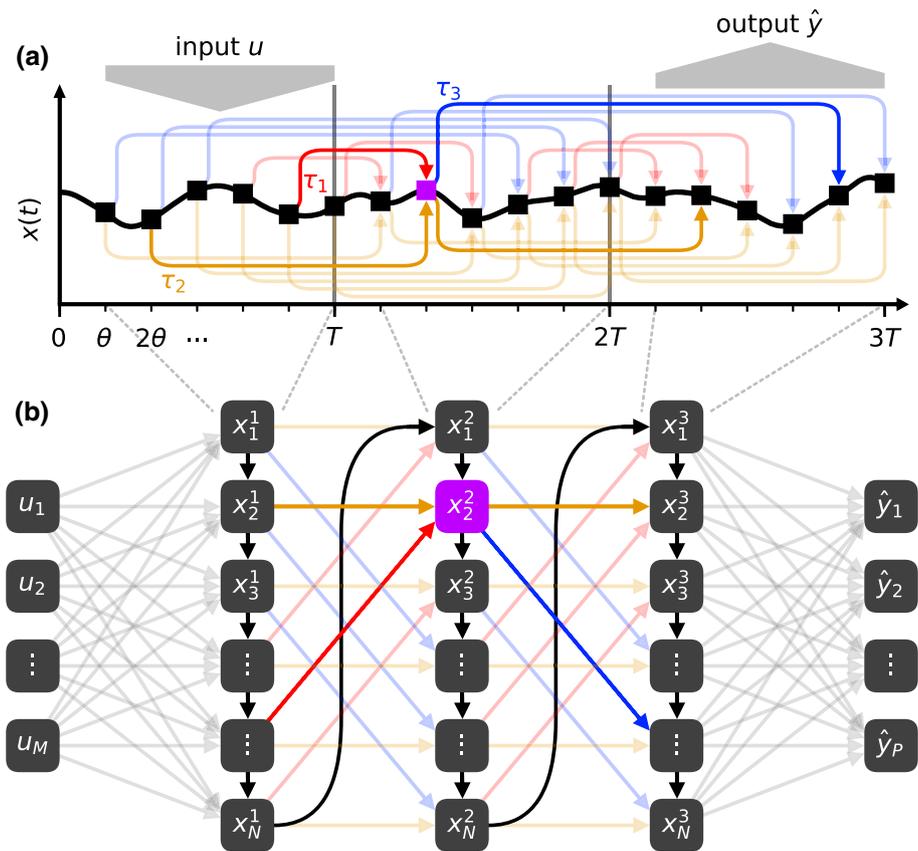
$$\mathcal{M}_d(t) = 0, \quad t \in I_1, \quad d = 1, \dots, D. \quad (11)$$

The dependence on the input vector  $u \in \mathbb{R}^{M+1}$  can be realized by the driving signal  $z(t)$ .

Property (V): The driving signal  $z(t)$  on the interval  $I_1$  is the step function given by

$$z(t) = J(t) \quad \text{for } t \in I_1, \quad (12)$$

**Fig. 4** Implementing a multilayer neural network by delay system (1). **a** The system state is considered at discrete time points  $x_n^\ell := x((\ell - 1)T + n\theta)$ . The intervals  $I_\ell$  correspond to layers. Due to delayed feedback, non-local connections emerge (color lines). **b** shows the resulting neural network



$$J(t) = J_n = [f^{\text{in}}(W^{\text{in}}u)]_n \quad \text{for } t \in I_{1,n}, \quad (13)$$

where  $f^{\text{in}}(W^{\text{in}}u) \in \mathbb{R}^N$  is the preprocessed input,  $W^{\text{in}} \in \mathbb{R}^{N \times (M+1)}$  is an input weight matrix, and  $f^{\text{in}}$  is an element-wise input preprocessing function. For example,  $f^{\text{in}}(a) = \tanh(a)$  was used in [50].

As a result, the following holds for the first hidden layer

$$\dot{x}(t) = f(x(t), J(t), 0, \dots, 0), \quad t \in I_1, \quad (14)$$

which is just a system of ordinary differential equations, which requires an initial condition at a single point  $x(0) = x_0$  for solving it in positive time. This yields the coupled map representation

$$x_1^\ell = x_0 + \theta f(x_0, J_1, 0, \dots, 0), \quad (15)$$

$$x_n^1 = x_{n-1}^1 + \theta f(x_{n-1}^1, J_n, 0, \dots, 0), \quad n = 2, \dots, N. \quad (16)$$

For the hidden layers  $I_2, I_3, \dots$ , the driving function  $z(t)$  can be used to introduce a bias as follows.

Property (VI): The driving signal  $z(t)$  on the intervals  $I_\ell$ ,  $\ell \geq 2$ , is the step function given by

$$z(t) = b(t) \quad \text{for } t > T, \quad (17)$$

$$b(t) = b_n^\ell \quad \text{for } t \in I_{\ell,n}, \quad \ell \geq 2. \quad (18)$$

Assuming the properties (I)–(VI), Eqs. (7)–(8) imply

$$x_1^\ell = x_N^{\ell-1} + \theta f(x_N^{\ell-1}, b_1^\ell, v_{1,1}^\ell x_{1-n'_1}^{\ell-1}, \dots, v_{D,1}^\ell x_{1-n'_D}^{\ell-1}), \quad (19)$$

$$x_n^\ell = x_{n-1}^\ell + \theta f(x_{n-1}^\ell, b_n^\ell, v_{1,n}^\ell x_{n-n'_1}^{\ell-1}, \dots, v_{D,n}^\ell x_{n-n'_D}^{\ell-1}), \quad n = 2, \dots, N. \quad (20)$$

Let us finally define the output layer, which transforms the node states  $x_1^L, \dots, x_N^L$  of the last hidden layer to an output vector  $\hat{y} \in \mathbb{R}^P$ . For this, we define a vector  $x^L := (x_1^L, \dots, x_N^L, 1)^T \in \mathbb{R}^{N+1}$ , an output weight matrix  $W^{\text{out}} \in \mathbb{R}^{P \times (N+1)}$ , and an output activation function  $f^{\text{out}}: \mathbb{R}^P \rightarrow \mathbb{R}^P$ . The output vector is then defined as

$$\hat{y} = f^{\text{out}}(W^{\text{out}}x^L). \quad (21)$$

Figure 4 illustrates the whole construction process of the coupled maps network; it is given by Eqs. (15)–(21). We summarize the main result of section 2.

Under assumptions (I)–(VI) and for small  $\theta$ , DDE (1) describes the multilayer network of coupled maps shown in Fig. 4, with the specific dependencies given by Eqs. (15), (16), (19), (20), and (21).

### 3 Constructing a recurrent neural network from a delay system

System (1) can also be considered as recurrent neural network. To show this, we consider the system on the time interval  $[0, KT]$ , for some  $K \in \mathbb{N}$ , which is divided into intervals  $I_k := ((k-1)T, kT]$ ,  $k = 1, \dots, K$ . We use  $k$  instead of  $\ell$  as index for the intervals to make clear that the intervals do not represent layers. The state  $x(t)$  on an interval  $I_k$  is interpreted as the state of the recurrent network at time  $k$ . More specifically,

$$x_n^k := x((k-1)T + n\theta), \quad n = 1, \dots, N, \quad k = 1, \dots, K \tag{22}$$

is the state of node  $n$  at the discrete time  $k$ . The driving function  $z(t)$  can be utilized as an input signal for each  $k$ -time-step.

Property (VII):  $z(t)$  is the  $\theta$ -step function with

$$z(t) = z_n^k \quad \text{for } t \in I_{k,n}, \tag{23}$$

$$(z_1^k, \dots, z_N^k)^T = f^{\text{in}}(W^{\text{in}}u(k)), \tag{24}$$

where  $u(k)$ ,  $k = 1, \dots, K$  are  $(M + 1)$ -dimensional input vectors,  $W^{\text{in}} \in \mathbb{R}^{N \times (M+1)}$  is an input weight matrix,  $f^{\text{in}}$  is an element-wise input preprocessing function. Each input vector  $u(k)$  contains  $M$  input values  $u_1(k), \dots, u_M(k)$  and a fixed entry  $u_{M+1}(k) := 1$  which is needed to include bias weights in the last column of  $W^{\text{in}}$ .

The main difference of the Property (VII) from (VI) is that it allows for the information input through  $z(t)$  in all intervals  $I_k$ . Another important difference is related to the modulation functions, which must be  $T$ -periodic to implement a recurrent network. This leads to the following assumption.

Property (VIII): The modulation functions  $\mathcal{M}_d(t)$  are  $T$ -periodic  $\theta$ -step functions with

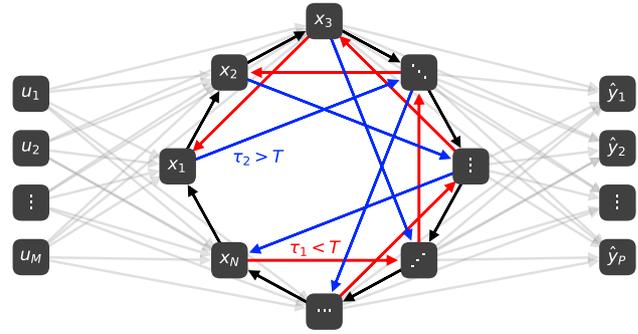
$$\mathcal{M}_d(t) = v_{d,n} \quad \text{for } t \in I_{k,n}. \tag{25}$$

Note that the value  $v_{d,n}$  is independent on  $k$  due to periodicity of  $\mathcal{M}_d(t)$ . When assuming the Properties (I), (III), (IV), (VII), and (VIII), the map equations (7)–(8) become

$$x_1^k = x_N^{k-1} + \theta f(x_N^{k-1}, z_1^k, v_{1,1}x_{1+n'_1}^{k-1}, \dots, v_{D,1}x_{1+n'_D}^{k-1}), \tag{26}$$

$$x_n^k = x_{n-1}^k + \theta f(x_{n-1}^k, z_n^k, v_{1,n}x_{n+n'_1}^{k-1}, \dots, v_{D,n}x_{n+n'_D}^{k-1}), \quad n = 2, \dots, N, \tag{27}$$

and can be interpreted as a recurrent neural network with the input matrix  $W^{\text{in}}$  and the internal weight



**Fig. 5** Recurrent network obtained from DDE (1) with two delays. The delays  $\tau_1 < T$  and  $\tau_2 > T$  induce connections with opposite direction (color arrows). Moreover, the nodes of the recurrent layer are linearly locally coupled (black arrows). All nodes of the recurrent layers are connected to the input and output layer

matrix  $W = (w_{nj}) \in \mathbb{R}^{N \times N}$  defined by

$$w_{nj} := \sum_{d=1}^D \delta_{n-n'_d, j} v_{d,n} = \begin{cases} 0 & \text{if } \forall d: j \neq n - n'_d, \\ v_{d,n} & \text{if } \exists d: j = n - n'_d. \end{cases} \tag{28}$$

When we choose the number of delays to be  $D = 2N - 1$ , we can realize any given connection matrix  $W \in \mathbb{R}^{N \times N}$ . For that we need to choose the delays  $\tau_d = d\theta$ ,  $d = 1, \dots, 2N - 1$ . Consequently there are  $D = 2N - 1$  modulation functions  $\mathcal{M}_d(t)$  which are step functions with values  $v_{d,n}$ . In this case, Eq. (28) provides for all entries  $w_{nj}$  of  $W$  exactly one corresponding  $v_{d,n}$ . Therefore, the arbitrary matrix  $W$  can be realized by choosing appropriate step heights for the modulation functions. In the setting of Sect. 3, the resulting network is an arbitrary recurrent network.

Summarizing, the main message of Sect. 3 is as follows.

Under assumptions (I), (III), (IV), (VII), and (VIII), and for small  $\theta$ , DDE (1) describes the recurrent network shown in Fig. 5, with the specific dependencies given by Eqs. (26)–(27) and an internal weight matrix  $W$  given by (28).

### 4 Networks from delay systems with linear instantaneous part and nonlinear delayed feedback

Particularly suitable for the construction of neural networks are delay systems with a stable linear instantaneous part and a feedback given by a nonlinear function of an affine combination of the delay terms and a driv-

ing signal. Such DDEs are described by the equation

$$\dot{x}(t) = -\alpha x(t) + f(a(t)), \tag{29}$$

where  $\alpha > 0$  is a constant time scale,  $f$  is a nonlinear function, and

$$a(t) = z(t) + \sum_{d=1}^D \mathcal{M}_d(t)x(t - \tau_d). \tag{30}$$

Ref. [8] studied this type of equation for the case  $D = 1$ , i.e. for one delay.

An example of (29) is the Ikeda system [59] where  $D = 1$ , i.e.  $a(t)$  consists of only one scaled feedback term  $x(t - \tau)$ , signal  $z(t)$ , and the nonlinear function  $f(a) = \sin(a)$ . This type of dynamics can be applied to reservoir computing using optoelectronic hardware [33]. Another delay dynamical system of type (29), which can be used for reservoir computing, is the Mackey–Glass system [30], where  $D = 1$  and the nonlinearity is given by  $f(a) = \eta a / (1 + |a|^p)$  with constants  $\eta, p > 0$ . In the work [50], system (29) is used to implement a deep neural network.

Even though the results of the previous sections are applicable to (29)–(30), the special form of these equations allows for an alternative, more precise approximation of the network dynamics.

### 4.1 Interpretation as multilayer neural network

It is shown in [50] that one can derive a particularly simple map representation for system (29) with activation signal (30). We do not repeat here the derivation, and only present the resulting expressions. By applying a semi-analytic Euler discretization and the variation of constants formula, the following equations connecting the nodes in the network are obtained:

$$x_1^1 = e^{-\alpha\theta} x_0 + \alpha^{-1}(1 - e^{-\alpha\theta})f(a_1^1), \tag{31}$$

$$x_n^1 = e^{-\alpha\theta} x_{n-1}^1 + \alpha^{-1}(1 - e^{-\alpha\theta})f(a_n^1), \quad n = 2, \dots, N, \tag{32}$$

for the first hidden layer. The hidden layers  $\ell = 2, \dots, L$  are given by

$$x_1^\ell = e^{-\alpha\theta} x_N^{\ell-1} + \alpha^{-1}(1 - e^{-\alpha\theta})f(a_1^\ell), \tag{33}$$

$$x_n^\ell = e^{-\alpha\theta} x_{n-1}^\ell + \alpha^{-1}(1 - e^{-\alpha\theta})f(a_n^\ell), \quad n = 2, \dots, N. \tag{34}$$

The output layer is defined by

$$\hat{y}_p := f_p^{\text{out}}(a^{\text{out}}), \quad p = 1, \dots, P, \tag{35}$$

where  $f^{\text{out}}$  is an output activation function. Moreover,

$$a_n^{\text{in}} := \sum_{m=1}^{M+1} w_{nm}^{\text{in}} u_m, \quad n = 1, \dots, N, \tag{36}$$

$$a_n^1 := g(a_n^{\text{in}}), \quad n = 1, \dots, N, \tag{37}$$

$$a_n^\ell := \sum_{j=1}^{N+1} w_{nj}^\ell x_j^{\ell-1}, \quad n = 1, \dots, N, \quad \ell = 2, \dots, L, \tag{38}$$

$$a_p^{\text{out}} := \sum_{n=1}^{N+1} w_{pn}^{\text{out}} x_n^L, \quad p = 1, \dots, P, \tag{39}$$

where  $u_{M+1} := 1$  and  $x_{N+1}^\ell := 1$ , for  $\ell = 1, \dots, L$ .

One can also formulate the relation between the hidden layers in a matrix form. For this, we define

$$A := \begin{pmatrix} 0 & \dots & \dots & 0 \\ e^{-\alpha\theta} & \ddots & & \vdots \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & e^{-\alpha\theta} & 0 \end{pmatrix}. \tag{40}$$

Then, for  $\ell = 2, \dots, L$ , Eqs. (33)–(34) become

$$x^\ell = Ax^\ell + \begin{pmatrix} e^{-\alpha\theta} x_N^{\ell-1} \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \alpha^{-1}(1 - e^{-\alpha\theta})f(W^\ell x^{\ell-1}). \tag{41}$$

where  $f$  is applied component-wise. By subtracting  $Ax^\ell$  from both sides of Eq. (41) and multiplication by the matrix

$$E := (\text{Id} - A)^{-1} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ e^{-\alpha\theta} & 1 & \ddots & & \vdots \\ e^{-2\alpha\theta} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ e^{-(N-1)\alpha\theta} & \dots & e^{-2\alpha\theta} & e^{-\alpha\theta} & 1 \end{pmatrix}, \tag{42}$$

we obtain a matrix equation describing the  $\ell$ th hidden layer

$$x^\ell = \begin{pmatrix} e^{-\alpha\theta} x_N^{\ell-1} \\ e^{-2\alpha\theta} x_N^{\ell-1} \\ \vdots \\ e^{-N\alpha\theta} x_N^{\ell-1} \end{pmatrix} + \alpha^{-1}(1 - e^{-\alpha\theta})E f(W^\ell x^{\ell-1}). \tag{43}$$

The neural network (31)–(39) obtained from delay system (29)–(30) can be trained by gradient descent [50]. The training parameters are the entries of the matrices  $W^{\text{in}}$  and  $W^{\text{out}}$ , the step heights of the modulation functions  $\mathcal{M}_d(t)$ , and the bias signal  $b(t)$ .

## 4.2 Network for large node distance $\theta$

In contrast to the general system (1), the semilinear system (29) with activation signal (30) does not only emulate a network of nodes for small distance  $\theta$ . It is also possible to choose large  $\theta$ . In this case, we can approximate the nodes given by Eq. (2) by the map limit

$$x^\ell = \alpha^{-1} f(a^\ell), \quad (44)$$

$$\text{where } a^\ell = W^\ell x^{\ell-1} \text{ for } \ell > 1 \text{ and } a^1 = g(W^{\text{in}} u), \quad (45)$$

up to exponentially small terms.

The reason for this limit behavior lies in the nature of the local couplings. Considering Eq. (29), one can interpret the parameter  $\alpha$  as a time scale of the system, which determines how fast information about the system state at a certain time point decays while the system is evolving. This phenomenon is related to the so-called instantaneous Lyapunov exponent [60–62], which equals  $-\alpha$  in this case. As a result, the local coupling between neighboring nodes emerges when only a small amount of time  $\theta$  passes between the nodes. Hence, increasing  $\theta$  one can reduce the local coupling strength until it vanishes up to a negligibly small value. For a rigorous derivation of Eq. (44), we refer to [50].

The apparent advantage of the map limit case is that the obtained network matches a classical multi-layer perceptron. Hence, known methods such as gradient descent training via the classical back-propagation algorithm [63] can be applied to the delay-induced network [50].

The downside of choosing large values for the node separation  $\theta$  is that the overall processing time of the system scales linearly with  $\theta$ . We need a period of time  $T = N\theta$  to process one hidden layer. Hence, processing a whole network with  $L$  hidden layers requires the time period  $LT = LN\theta$ . For this reason, the work [50] provides a modified back-propagation algorithm for small node separations to enable gradient descent training of networks with significant local coupling.

## 5 Conclusions

We have shown how networks of coupled maps with arbitrary topology and arbitrary size can be emulated by a single (possibly even scalar) DDE with multiple delays. Importantly, the coupling weights can be adjusted by changing the modulations of the feedback signals. The network topology is determined by the choice of time-delays. As shown previously [30, 33, 34, 39, 50], special cases of such networks are successfully applied for reservoir computing or deep learning.

As an interesting conclusion, it follows that the temporal dynamics of DDEs can unfold arbitrary spatial complexity, which, in our case, is reflected by the topology of the unfolded network. In this respect, we shall

mention previously reported spatio-temporal properties of DDEs [7, 64–72]. These results show how in some limits, mainly for large delays, the DDEs can be approximated by partial differential equations.

Further, we remark that similar procedures have been used for deriving networks from systems of ordinary differential equations [52–54]. However, in their approach, one should use an  $N$ -dimensional system of equations for implementing layers with  $N$  nodes. This is in contrast to the DDE case, where the construction is possible with just a single-variable equation.

As a possible extension, a realization of adaptive networks using a single node with delayed feedback would be an interesting open problem. In fact, the application to deep neural networks in [50] realizes an adaptive mechanism for the adjustment of the coupling weights. However, this adaptive mechanism is specially tailored for DNN problems. Another possibility would be to emulate networks with dynamical adaptivity of connections [73]. The presented scheme can also be extended by employing delay differential-algebraic equations [74, 75].

**Acknowledgements** This work was funded by the “Deutsche Forschungsgemeinschaft” (DFG) in the framework of the project 411803875 (S.Y.) and IRTG 1740 (F.S.).

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. G. Stepan, *Retarded dynamical systems: stability and characteristic functions* (Longman Scientific & Technical, Harlow, 1989)
2. J.K. Hale, S.M.V. Lunel, *Introduction to Functional Differential Equations* (Springer, New York, 1993)
3. O. Diekmann, S. M. Verduyn Lunel, S. A. van Gils, H.-O. Walther, *Delay Equations*, Vol. 110. In *Applied Mathematical Sciences* (Springer, New York, 1995)
4. T. Erneux, *Applied Delay Differential Equations*, Vol. 3. In *Surveys and Tutorials in the Applied Mathematical Sciences* (Springer, New York, 2009)

5. H. L. Smith, *An Introduction to Delay Differential Equations with Applications to the Life Sciences*, Vol. 57. In *Texts in Applied Mathematics* (Springer, New York, 2010)
6. T. Erneux, J. Javaloyes, M. Wolfrum, S. Yanchuk, *Chaos Interdiscip. J. Nonlinear Sci.* **27**, 114201 (2017)
7. S. Yanchuk, G. Giacomelli, *J. Phys. A: Math. Theor.* **50**, 103001 (2017)
8. T. Krisztin, *Period. Math. Hung.* **56**, 83–95 (2008)
9. A.G. Vladimirov, D. Turaev, *Phys. Rev. A* **72**, 033808 (2005)
10. H. Erzgräber, B. Krauskopf, D. Lenstra, SIAM, *J. Appl. Dyn. Syst.* **5**, 30–65 (2006)
11. O. D’Huys, R. Vicente, T. Erneux, J. Danckaert, I. Fischer, *Chaos* **18**, 37116 (2008)
12. R. Vicente, I. Fischer, C.R. Mirasso, *Phys. Rev. E (Stat. Nonlinear Soft Matter Phys.)* **78**, 66202 (2008)
13. B. Fiedler, S. Yanchuk, V. Flunkert, P. Hövel, H.-J.J. Wünsche, E. Schöll, *Phys. Rev. E* **77**, 066207 (2008)
14. M. Wolfrum, S. Yanchuk, P. Hövel, E. Schöll, *Eur. Phys. J. Spec. Top.* **191**, 91–103 (2011)
15. S. Yanchuk, M. Wolfrum, SIAM *J. Appl. Dyn. Syst.* **9**, 519–535 (2010)
16. M.C. Soriano, J. García-Ojalvo, C.R. Mirasso, I. Fischer, *Rev. Mod. Phys.* **85**, 421–470 (2013)
17. N. Oliver, T. Jüngling, I. Fischer, *Phys. Rev. Lett.* **114**, 123902 (2015)
18. M. Marconi, J. Javaloyes, S. Barland, S. Balle, M. Giudici, *Nat. Photon.* **9**, 450–455 (2015)
19. D. Puzryev, A.G. Vladimirov, S.V. Gurevich, S. Yanchuk, *Phys. Rev. A* **93**, 041801 (2016)
20. S. Yanchuk, S. Ruschel, J. Sieber, M. Wolfrum, *Phys. Rev. Lett.* **123**, 053901 (2019)
21. J. Foss, J. Milton, *J. Neurophysiol.* **84**, 975–985 (2000)
22. J. Wu, *Introduction to Neural Dynamics and Signal Transmission Delay* (Walter de Gruyter, Berlin, 2001)
23. E.M. Izhikevich, *Neural Comput.* **18**, 245–282 (2006)
24. G. Stepan, *Philos. Trans. R. Soc. Lond. A Math. Phys. Eng. Sci.* **367**, 1059–1062 (2009)
25. G. Deco, V. Jirsa, A.R. McIntosh, O. Sporns, R. Kotter, *Proc. Natl. Acad. Sci.* **106**, 10302–10307 (2009)
26. P. Perlikowski, S. Yanchuk, O.V. Popovych, P.A. Tass, *Phys. Rev. E* **82**, 036208 (2010)
27. O.V. Popovych, S. Yanchuk, P.A. Tass, *Phys. Rev. Lett.* **107**, 228102 (2011)
28. M. Kantner, S. Yanchuk, *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **371**, 20120470 (2013)
29. H. Paugam-Moisy, R. Martinez, S. Bengio, *Neurocomputing* **71**, 1143–1158 (2008)
30. L. Appeltant, M.C. Soriano, G. van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C.R. Mirasso, I. Fischer, *Nat. Commun.* **2**, 468 (2011)
31. R. Martinenghi, S. Rybalko, M. Jacquot, Y.K. Chembo, L. Larger, *Phys. Rev. Lett.* **108**, 244101 (2012)
32. L. Appeltant, PhD thesis, Vrije Universiteit Brussel, Universitat de les Illes Balears (2012)
33. L. Larger, M.C. Soriano, D. Brunner, L. Appeltant, J.M. Gutierrez, L. Pesquera, C.R. Mirasso, I. Fischer, *Opt. Express* **20**, 3241–3249 (2012)
34. D. Brunner, M. Soriano, C. Mirasso, I. Fischer, *Nat. Commun.* **4**, 1364 (2013)
35. J. Schumacher, H. Toutounji, G. Pipa, *In Artificial Neural Networks and Machine Learning: Proceedings of the 23rd International Conference on Artificial Neural Networks, 26–33* (Springer, Berlin, 2013)
36. H. Toutounji, J. Schumacher, G. Pipa, *IEICE Proc. Ser.* **1**, 519–522 (2014)
37. L. Grigoryeva, J. Henriques, L. Larger, J.-P.P. Ortega, *Sci. Rep.* **5**, 1–11 (2015)
38. B. Penkovsky, PhD thesis, Université Paris-Sud 11 (2017)
39. L. Larger, A. Baylón-Fuentes, R. Martinenghi, V.S. Udaltsov, Y.K. Chembo, M. Jacquot, *Phys. Rev. X* **7**, 1–14 (2017)
40. K. Harkhoe, G. Van der Sande, *Photonics* **6**, 124 (2019)
41. F. Stelzer, A. Röhm, K. Lüdge, S. Yanchuk, *Neural Netw.* **124**, 158–169 (2020)
42. J.D. Hart, L. Larger, T.E. Murphy, R. Roy, *Philos. Trans. R. Soc. A. Math. Phys. Eng. Sci.* **377**, 20180123 (2019)
43. F. Köster, S. Yanchuk, K. Lüdge, Preprint at [arXiv:2009.07928](https://arxiv.org/abs/2009.07928) (2020)
44. F. Köster, D. Ehlert, K. Lüdge, *Cognit. Comput.* **1–8** (2020)
45. A. Argyris, J. Cantero, M. Galletero, E. Pereda, C.R. Mirasso, I. Fischer, M.C. Soriano, *IEEE J. Sel. Top. Quantum Electron.* **26**, 1–9 (2020)
46. M. Goldmann, F. Köster, K. Lüdge, S. Yanchuk, *Chaos Interdiscip. J. Nonlinear Sci.* **30**, 93124 (2020)
47. C. Sugano, K. Kanno, A. Uchida, *IEEE J. Sel. Top. Quantum Electron.* **26**, 1–9 (2020)
48. J.D. Hart, D.C. Schmadel, T.E. Murphy, R. Roy, *Chaos Interdiscip. J. Nonlinear Sci.* **27**, 121103 (2017)
49. L. Keuninckx, J. Danckaert, G. Van der Sande, *Cognit. Comput.* **9**, 315–326 (2017)
50. F. Stelzer, A. Röhm, R. Vicente, I. Fischer, S. Yanchuk, Preprint at [arXiv:2011.10115](https://arxiv.org/abs/2011.10115) (2020)
51. M. Hermans, J. Dambre, P. Bienstman, *IEEE Trans. Neural Netw. Learning Syst.* **26**, 1545–1550 (2015)
52. E. Haber, L. Ruthotto, *Inverse Probl.* **34**, 014004 (2017)
53. Y. Lu, A. Zhong, Q. Li, B. Dong, in *Beyond Finite Layer Neural Networks: Bridging Deep Architectures and Numerical Differential Equations*, Vol. 80. *Proceedings of Machine Learning Research*, 3276–3285 (PMLR, 2018)
54. R.T.Q. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, *In Proceedings of the 32nd International Conference on Neural Information Processing Systems, 6572–6583* (Curran Associates Inc., Red Hook, 2018)
55. C.M. Bishop, *Pattern Recognition and Machine Learning* (Springer, New York, 2006)
56. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning* (MIT Press, Cambridge, 2016)
57. Y. Lecun, Y. Bengio, G. Hinton, *Nature* **521**, 436–444 (2015)
58. J. Schmidhuber, *Neural Netw.* **61**, 85–117 (2015)
59. K. Ikeda, *Opt. Commun.* **30**, 257–261 (1979)
60. S. Heiligenthal, T. Dahms, S. Yanchuk, T. Jüngling, V. Flunkert, I. Kanter, E. Schöll, W. Kinzel, *Phys. Rev. Lett.* **107**, 234102 (2011)
61. S. Heiligenthal, T. Jüngling, O. D’Huys, D.A. Arroyo-Almanza, M.C. Soriano, I. Fischer, I. Kanter, W. Kinzel, *Phys. Rev. E* **88**, 12902 (2013)
62. W. Kinzel, *Philos. Trans. R. Soc. Math. Phys. Eng. Sci.* **371**, 20120461 (2013)

63. D. Rumelhart, G.E. Hinton, R.J. Williams, *Nature* **323**, 533–536 (1986)
64. F.T. Arecchi, G. Giacomelli, A. Lapucci, R. Meucci, *Phys. Rev. A* **45**, R4225–R4228 (1992)
65. G. Giacomelli, R. Meucci, A. Politi, F.T. Arecchi, *Phys. Rev. Lett.* **73**, 1099–1102 (1994)
66. G. Giacomelli, A. Politi, *Phys. Rev. Lett.* **76**, 2686–2689 (1996)
67. M. Bestehorn, E.V. Grigorieva, H. Haken, S.A. Kaschenko, *Physica D* **145**, 110–129 (2000)
68. G. Giacomelli, F. Marino, M.A. Zaks, S. Yanchuk, *EPL (Europhysics Letters)* **99**, 58005 (2012)
69. S. Yanchuk, G. Giacomelli, *Phys. Rev. Lett.* **112**, 1–5 (2014)
70. S. Yanchuk, G. Giacomelli, *Phys. Rev. E* **92**, 042903 (2015)
71. S. Yanchuk, L. Lücken, M. Wolfrum, A. Mielke, *Discrete Continuous Dyn. Syst. A* **35**, 537–553 (2015)
72. I. Kashchenko, S. Kaschenko, *Commun. Nonlinear Sci. Numer. Simul.* **38**, 243–256 (2016)
73. R. Berner, E. Schöll, S. Yanchuk, *SIAM J. Appl. Dyn. Syst.* **18**, 2227–2266 (2019)
74. P. Ha, V. Mehrmann, *BIT Numer. Math.* **56**, 633–657 (2016)
75. B. Unger, *Electron. J. Linear Algebra* **34**, 582–601 (2018)