Biol Cybern (2008) 98:371–395
DOI 10.1007/s00422-008-0221-5

ORIGINAL PAPER

Biological Cybernetics

Elements for a general memory structure: properties of recurrent neural networks used to form situation models

Valeri A. Makarov · Yongli Song · Manuel G. Velarde · David Hübner · Holk Cruse

Received: 24 August 2007 / Accepted: 6 February 2008 / Published online: 19 March 2008

Abstract We study how individual memory items are stored assuming that situations given in the environment can be represented in the form of synaptic-like couplings in recurrent neural networks. Previous numerical investigations have shown that specific architectures based on suppression or max units can successfully learn static or dynamic stimuli (situations). Here we provide a theoretical basis concerning the learning process convergence and the network response to a novel stimulus. We show that, besides learning "simple" static situations, a nD network can learn and replicate a sequence of up to n different vectors or frames. We find limits on the learning rate and show coupling matrices developing during training in different cases including expansion of the network into the case of nonlinear interunit coupling. Furthermore, we show that a specific coupling matrix provides low-pass-filter properties to the units, thus connecting networks constructed by static summation units with continuous-time networks. We also show under which conditions such networks can be used to perform arithmetic calculations by means of pattern completion.

Keywords Recurrent neural network · Situation model · Memory · Learning

V. A. Makarov · Y. Song · M. G. Velarde
Instituto Pluridisciplinar, Universidad Complutense,
Paseo Juan XXIII, 1, 28040 Madrid, Spain

V. A. Makarov
Department de Matemática Aplicada, Facultad de Matemáticas,
Universidad Complutense, Avda. Complutense s/n,
28040 Madrid, Spain
e-mail: vmakarov@mat.ucm.es

Y. Song
Department of Mathematics,
Tongji University, 200092 Shanghai, China

D. Hübner · H. Cruse (✉)
Department of Biological Cybernetics, Faculty of Biology,
University of Bielefeld, 33501 Bielefeld, Germany
e-mail: holk.cruse@uni-bielefeld.de

1 Introduction

How biological memories are organized is still a fairly open question, although a huge number of experimental studies have been reported dealing with features at different levels and using methods from different fields such as psychology or neurophysiology including brain imaging techniques. This situation has eventually been dubbed the crisis of the experimentalists. Is there a way to overcome this problem? It has been said that we only understand what we can construct (Feynman 2001). Accordingly, in order to understand brain functions, simulation studies appear to be a sensible solution. Such simulations on the one hand can originate new principles of information storage and engineering, and, on the other hand, may suggest experimental procedures to test novel hypotheses.

In the search for appropriate simulation models, recurrent neural networks (RNNs) have been intensively studied. This begun with Hopfield's seminal papers (Hopfield 1982, 1984) and has led to a vast literature. Significant architectures are Elman–Jordan networks (Elman 1990) or echo state networks (Jaeger and Haas 2004). Apart from many studies concentrating on monolithic architectures, sparsely coded networks (Palm and Sommer 1996) or expert networks (see, e.g., Tani 2003) have been investigated. In particular the latter show the advantage that they form separable

modules which can be treated in an easier way compared to monolithic structures, both with respect to studying the mathematical properties (Beer 2006; Pasemann 2002) and with respect to the way in which these modules might be implemented into a large memory structure. The latter, for example, implies problems of how memory contents can be organized to reflect hierarchical or contextual relationships. Another question concerns the structure of the basic RNN forming the individual modules, or neural assemblies. Two simple types of RNNs have recently been investigated and proposed as a possible basis for such elementary memory structures: multiple solutions of basic equations (MSBE) networks (Kühn et al. 2007; Kühn and Cruse 2007) and mean of multiple computations (MMC) networks (Kindermann and Cruse 2002; Steinkühler and Cruse 1998). It has been shown that such MSBE networks can be considered as building blocks to construct MMC networks. Specific RNNs have been used to produce holistic models of geometrical structures, for example, bodies with arms and legs showing a large number of degrees of freedom, and form models that can be used as forward models, inverse models or any mixed combination. Furthermore, RNNs based on linear or nonlinear differential equations have been studied (Kühn et al. 2007).

In Kühn et al. (2007) and Kühn and Cruse (2007) the authors concentrated on the RNNs as such, whereas in Cruse and Hübner (2008) a general memory architecture has been proposed, in which these relatively small networks can be embedded. This general architecture, inspired by the insect mushroom body system (Wessnitzer and Webb 2006), can be used for learning and controlling more-complex behaviors such as landmark-based navigation. The same architecture is currently being studied to form a general theory explaining many of the Pavlovian paradigms (Cruse and Sievers 2008).

These RNNs, although of simple structure, have been shown to be able to learn some arithmetic relations, but also dynamic features of linear and nonlinear oscillators. However, although attractive because of their simple structure, the dynamical properties of the learning process and of the resulting RNN have only been studied numerically, and a mathematical proof of convergence exists only for a very limited special case (Kühn et al. 2007). A general proof of convergence and of more specific aspects of the dynamic behavior, e.g., response to a novel stimulus, of trained networks is lacking. In addition, two slightly different versions of the input compensation (IC) units have been proposed that show a somewhat different behavior when connected to form an RNN. In the present work we offer new insight and give detailed proofs of several of the phenomenologically observed behaviors, thus providing a solid basis for further investigation and development of this simple but promising memory architecture.

2 IC units, IC network, basic equation, and learning rule

The networks considered here consist of n recurrently connected units. As mentioned above, two different types of "simple" nonlinear units have been proposed, which will be introduced first. These units have been called, respectively, the suppression unit or *Su* and the max unit or *Mu* (Fig. 1a, b). Both units operate in discrete time $t \in \mathbb{Z}$ and have an external input denoted as the signal $\xi_i(t)$ that we also call the activation, an internal (recurrent) input $s_i(t)$, and an output $x_i(t+1)$. The recurrent input is given by a weighted sum of the output of all units in the network

$$s_i(t) = \sum_{k=1}^{n} w_{ik} x_k(t), \tag{1}$$

where the matrix $W = \{w_{ij}\}$ plays the role of interunit coupling.

The nonlinear properties of the units arise from the treatment of the recurrent signal according to the signal at the external input. In *Su* the recurrent signal is simply suppressed and replaced by the external input if the latter is different from zero, or otherwise sent unchanged to the output

$$x^{Su}(t+1) = \begin{cases} \xi(t), & \text{if } \xi(t) \neq 0 \\ s(t), & \text{otherwise} \end{cases} \tag{2}$$

The output of the *Mu* is given by

$$x^{Mu}(t+1) = \begin{cases} \max(\xi(t), s(t)), & \text{if } s(t) \geq 0 \\ \min(\xi(t), s(t)), & \text{otherwise} \end{cases} \tag{3}$$

Another way of representing the transduction properties of the *Mu* is

$$x^{Mu}(t+1) = s(t) + \begin{cases} [\xi(t) - s(t)]_+, & \text{if } s(t) > 0 \\ -[s(t) - \xi(t)]_+, & \text{otherwise} \end{cases} \tag{4}$$

where the subscript plus denotes the rectifier operator

$$x_+ = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Although equivalent, representation (4) is better suited for hardware implementation, whereas (3) better serves for mathematical analysis.

Combining n IC units (either *Mu* or *Su*) into a recurrent network (Fig. 1c) yields a system whose dynamics is described by the following nD map

$$x_i(t+1) = F\left(\xi_i(t), \sum_{k=1}^{n} w_{ik} x_k(t)\right), \quad i = 1, \ldots, n, \tag{6}$$

where F is a nonlinear function determined by the type of units used in the network, given either by (2) for *Su* or (3) for *Mu*.

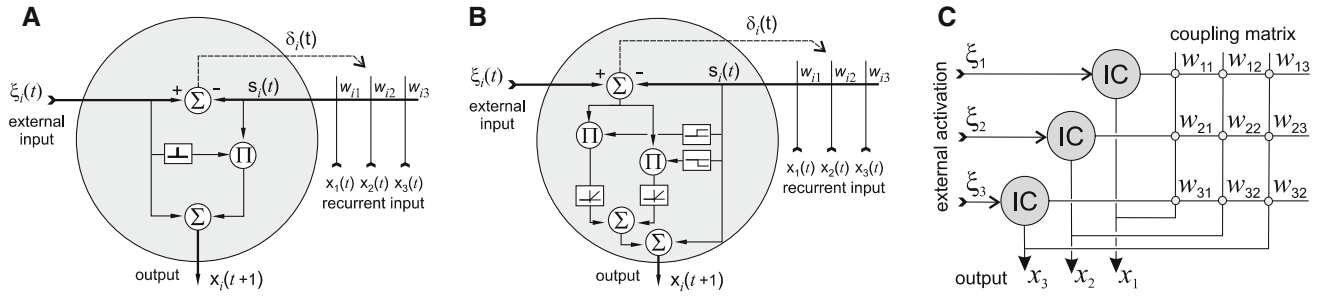

Fig. 1 Circuit implementation of a recurrent neural network composed of input compensation (nonlinear) units: **a** suppression IC unit (or *Su*) and **b** Max IC unit (or *Mu*). Both units have two inputs ($\xi_i(t)$ and $s_i(t)$) and one output ($x_i(t+1)$). The latter is given by a nonlinear function of the inputs (see main text). Blocks marked by Σ and Π perform input summation and multiplication, respectively. The other blocks are nonlinear elements with sketched transduction characteristics. **c** Neural network (case $n = 3$) composed of either suppression or max units recurrently coupled by the matrix $W = \{w_{ij}\}$

Before going further let us first introduce suitable notations. For any given vectors $x, y \in \mathbb{R}^n$, let

$$\langle x, y\rangle \equiv x^{\mathrm{T}} y$$

be a real number (T denotes transpose). Then $\|x\| = \sqrt{\langle x, x\rangle}$ is the length or modulus of the vector x. We shall also use the vector inequality

$$x < y \Leftrightarrow (x_i \le y_i,\ \forall i,\ \text{and}\ x \ne y)\,.$$

The recurrent IC network described by (6) can be trained to perform different tasks, for example to "remember" and reproduce given static or dynamic stimuli, or to perform simple algebraic associations. To illustrate the latter we introduce the so-called basic equation

$$\langle B, x\rangle = 0,\ \text{or}\ \sum_{i=1}^{n} B_i x_i = 0, \tag{7}$$

where $B \in \mathbb{R}^n$ is a constant vector. Equation (7) can be considered as an algebraic constraint to the network state; it defines a hyperplane passing through the origin in the nD phase space of the network. We note that the coefficients of the basic equation $B_{1,\ldots,n}$ are defined up to a constant, i.e., if $\langle B, x\rangle = 0$, then $\langle kB, x\rangle = 0$, where k is a constant. To rule out this uncertainty we can fix one of the coefficients, say $B_1 \equiv 1$.

As we shall show below the network can be trained in such a way that the basic Eq. (7) will be an attractor, i.e., any arbitrary activation of the network will relax to the aforementioned hyperplane. Once the network has been trained, i.e., the coefficients $B_{1,\ldots,n}$ have been learnt or somehow stored in the coupling matrix W, the network can be used to perform simple arithmetics. Indeed, giving a new incomplete activation (stimulus)

$$\xi_i(t) = e_i,\quad (i = 1, \ldots, n),\quad e_k = 0,$$

where e is an arbitrary nD vector ($\langle B, e\rangle \ne 0$), the network adopts a solution such that

$$x_i = e_i,\quad (i = 1, \ldots, n,\ i \ne k),\quad x_k = -\langle B, e\rangle / B_k,$$

i.e., satisfying the basic Eq. (7). This task can also be viewed as pattern completion. We notice that, even if more than one activation input is absent (equal to zero), the network will relax to a proper underdetermined solution.

The network can stay either in the learning or in the operational phase. In the operational phase the coupling matrix is fixed and the network reproduces the previously learnt stimulus or responds to a novel stimulus applied to the external input.

During learning the network is exposed to the training, in general dynamic, stimulus. We shall distinguish two types of situations to be learnt. One is so-called *static*, when external stimuli presented to the network are assumed to be (temporarily) independent pieces of a global picture; for example, the first vector a is presented and then, after resetting the activations of all units, another vector b is given. Then the learning requires two updating cycles at each vector presentation. Moreover, as we show below, the sequence of vectors can be arbitrary in this case. The other, *dynamic*, situation is characterized by essentially time-dependent stimuli, i.e., stimuli composed of different vectors whose sequence now indeed matters. Then we can write the situation on the external network input as a time function $\xi(t) = \xi(t + p)$, where p is the time-domain period of the training vector; for example, in the case $p = 1$, we train the network by a constant input vector $\xi(t) = a$; for $p = 2$ the network is trained by two alternating vectors: $\xi(t) = abababab\ldots$, and so on.

Training consists of appropriate adjustment of the coupling matrix W by using a learning rule (the same for *Su* and *Mu*), that can be described as teacher forcing based on the classical delta rule (Kühn et al. 2007). The updating of the

matrix elements is carried out according to

$$w_{ij}(t+1) = w_{ij}(t) + \varepsilon\delta_i(t)x_j(t), \tag{8}$$

where $\varepsilon > 0$ is the learning rate, and $\delta_i(t) = \xi_i(t) - s_i(t)$ is the error between the internal and external inputs (Fig. 1a, b). We assume that during the training of the weight matrix W the network has no internal dynamics, i.e., (2) and (3) are reduced to $x(t+1) = \xi(t)$. This is indeed true for *Su* with nonzero external activation. For *Mu* this can also be accomplished by starting the training process from $W = 0$ (and $s < \xi$).

For the static case the dynamics of the weight matrix is given by

$$W(t+1) = W(t)\left(I - \varepsilon\xi(t)\xi^{\mathrm{T}}(t)\right) + \varepsilon\xi(t)\xi^{\mathrm{T}}(t), \tag{9}$$

where $\xi(t)$ is the training (in general time-dependent) external activation applied to the network. We note that at each step the coupling matrix is updated by a single vector independently on the other elements of the training sequence.

For the dynamic situation the evolution of the weight matrix in the learning phase is given by

$$\begin{aligned} W(t+1) = {} & W(t)\left(I - \varepsilon\xi(t-1)\xi^{\mathrm{T}}(t-1)\right) \\ & + \varepsilon\xi(t)\xi^{\mathrm{T}}(t-1). \end{aligned} \tag{10}$$

The learning rule (10) can be considered as an $(n \times n)$ map driven by an external force with delay. Now, in contrast to (9), each training step uses two sequential vectors, hence their ordering becomes important.

The training is deemed finished when the total squared error $\sum \delta_i^2$ falls below a threshold. To quantify the learning performance we shall also use the normalized intermatrix distance

$$d(t) = \frac{\|W(t) - W_\infty\|}{\|W_\infty\|}, \tag{11}$$

where $W_\infty = \lim_{t\to\infty} W(t)$ is the limit (learnt) matrix. Note that W_∞ is not used for learning but only for a posteriori description of the learning dynamics.

We note that (9) or (10) in general may not converge to a fixed point, which means that the network is unable to learn (in terms of representation and replication) the corresponding external situation.

3 Learning static situations

Let us start with the case in which an RNN is used to learn a static situation. Then, as mentioned above, the learning follows the rule (9).

3.1 Convergence of the network training procedure

Let us assume that the static situation to be learnt is described by p nonzero vectors $\{a_1, a_2, \ldots, a_p\}$, and at each learning step the network is exposed to one of them. We also require that all training vectors appear sufficiently frequently in the learning sequence. The latter means that the occurrence frequency of the ith vector $f_i = t_i/t$ does not tend to zero for $t \to \infty$ (where t_i is the number of times the vector occurs up to time t). Examples of training sequences satisfying this condition are periodic (e.g., $\xi(t) = a_1, \ldots, a_p, a_1, \ldots, a_p, \ldots$), and probabilistic (vectors appear at random with nonzero probabilities).

3.1.1 Training by orthogonal vectors

First we consider the case when $a_1, \ldots, a_p$ are nonzero orthogonal vectors in $\mathbb{R}^n$. Then the following theorem on the convergence of the network learning process holds.

Theorem 1 *Assume that an n-unit RNN is trained by an arbitrary sequence of p nonzero orthogonal vectors $a_1, \ldots, a_p$, each of which appears in the sequence with nonzero frequency (i.e., an infinite number of times for an infinite sequence). If the learning rate satisfies*

$$0 < \varepsilon < \min\left\{\frac{2}{\|a_1\|^2}, \frac{2}{\|a_2\|^2}, \ldots, \frac{2}{\|a_p\|^2}\right\} \tag{12}$$

then for any initial conditions $W_0 = W(0)$ the learning process given by (9) converges to the coupling matrix

$$W_\infty \equiv \lim_{t\to\infty} W(t) = W_0(I - M_p) + M_p, \tag{13}$$

where

$$M_p = \sum_{i=1}^{p} \frac{a_i a_i^{\mathrm{T}}}{\|a_i\|^2}. \tag{14}$$

Particularly, for (i) $p = n$ or (ii) $p < n$ but with $W_0 = 0$.

$$W_\infty = M_p \tag{15}$$

The proof of the theorem is given in Appendix A.

We note that training by one vector, i.e., $\xi(t) = a$, reduces (14) to $M_1 = aa^{\mathrm{T}}/\|a\|^2$, which corresponds to the case considered in Kühn et al. (2007).

According to Theorem 1 the learning can always be achieved by using a small enough learning rate. The training starting with zero initial conditions ($W_0 = 0$) leads to a symmetric coupling matrix $w_{ij} = w_{ji}$. The learning result (in terms of W_∞) does not depend on the sequence of the presentation of the vectors $a_1, \ldots, a_p$ to the network. The latter, for instance, means that training by a periodic sequence of two vectors (e.g., $\xi(t) = a, b, a, b, a, \ldots$) gives the same matrix W_∞ as the training by a random sequence of these vectors (e.g., $\xi(t) = a, a, b, a, a, a, b, b, a, \ldots$), even if the

probability to find vector a is different from the probability to find vector b (provided that both are nonzero).

Finally, for practical implementation, we note that the learning time scales as

$$t \propto \frac{1}{\min\left(f_i \ln \frac{1}{1-\varepsilon\|a_i\|^2}\right)}.$$

Thus an excessively small (respectively, large) learning rate and/or small occurrence frequency of one of the training vectors deteriorates the learning performance. The latter particularly means that the network can learn equally well, say two vectors a_1 and a_2, even if the occurrence frequency of one of them is much smaller then that of the other (e.g., $f_1 \ll f_2$); however the training time in this case will be proportionally longer, i.e., $T_{\text{training}} \propto 1/f_1$.

3.1.2 Training by arbitrary vectors

The condition on the vector orthogonality used above is hardly satisfied by real-world stimuli. To extend our results we now relax this requirement.

Let us assume that the network is trained by an arbitrary sequence of p training vectors $\{a_1, \ldots, a_p\}$. We denote by $\{\gamma_1, \ldots, \gamma_r\}$ a maximal linearly independent subset of the complete training matrix. Then by the Gram–Schmidt orthogonalization procedure (Strang 2003) we define an orthogonal set $\{c_1, \ldots, c_r\}$, where

$$c_1 = \gamma_1, \quad c_k = \gamma_k - \sum_{j=1}^{k-1} \frac{\langle \gamma_k, c_j \rangle}{\| c_j \|^2} c_j \quad \text{for } 2 \le k \le r. \tag{16}$$

Now we are ready to formulate the following theorem:

Theorem 2 *Assume that an n-unit RNN is trained by an arbitrary sequence of p arbitrary nonzero vectors $a_1, \ldots, a_p$, each of which appears in the training sequence with nonzero frequency. If the learning process (9) starting from initial condition $W_0 = W(0)$ converges to the coupling matrix W_∞, then*

$$W_\infty = W_0(I - M_r) + M_r, \tag{17}$$

where

$$M_r = \sum_{k=1}^{r} \frac{c_k c_k^{\mathrm{T}}}{\|c_k\|^2}, \tag{18}$$

with $r = rank\{a_1, \ldots, a_p\}$. Particularly, for (i) $r = n$ or (ii) $r < n$ but with $W_0 = 0$,

$$W_\infty = M_r. \tag{19}$$

The proof of the theorem is given in Appendix B.

Remark 1 Although the chosen basis $\{c_1, \ldots, c_r\}$ is not unique, the matrix M_r does not depend on the choice, since an orthogonal projection (given by M_r) on the space spanned by $\{a_1, \ldots, a_p\}$ is unique (Strang 2003).

The independence of the learning result (matrix W_∞) on a particular sequence of the training vectors discussed above is also applied here. Another important point is that the learning operates on a *linearly independent subset* of the training vectors only. The latter, for instance, means that the training by an arbitrary sequence of three vectors a, b, and c, one of which is a linear combination of the other two (i.e., $c = k_1 a + k_2 b$), is equivalent to training by any pair of these vectors. In other words, the training by, e.g., $\xi(t) = a, b, c, a, b, c, \ldots$ gives the same coupling matrix W_∞ as the training by, e.g., $\xi(t) = a, c, a, c, a, \ldots$

3.2 Example of training by two vectors

Let us now give an example of the network training by two linearly independent vectors a and b satisfying a basic equation.

Using Theorem 2 with zero initial conditions ($W(0) = 0$), we obtain

$$W_\infty = \alpha\alpha^{\mathrm{T}} + \beta\beta^{\mathrm{T}}, \tag{20}$$

where

$$\alpha = \frac{a}{\|a\|}, \text{ and } \beta = \frac{\|a\|^2 b - \langle b, a\rangle a}{\|a\|\sqrt{\|a\|^2\|b\|^2 - \langle b, a\rangle^2}}. \tag{21}$$

We note that $\|\alpha\| = \|\beta\| = 1$ and $\langle \alpha, \beta \rangle = 0$, i.e., α and β are orthonormal vectors.

As an example let us train a 3D network to learn the basic equation

$$x + y - 2z = 0. \tag{22}$$

We chose as the training vectors $a = (1, 3, 2)^{\mathrm{T}}$ and $b = (1, 1, 1)^{\mathrm{T}}$, fulfilling (22). From (21) we obtain the equivalent vectors $\alpha = (1, 3, 2)^{\mathrm{T}}/\sqrt{14}$ and $\beta = (4, -2, 1)^{\mathrm{T}}/\sqrt{21}$. Finally, substituting them into (20) we get the limit trained matrix

$$W_{2\mathrm{v}} = \frac{1}{6}\begin{pmatrix} 5 & -1 & 2 \\ -1 & 5 & 2 \\ 2 & 2 & 2 \end{pmatrix}. \tag{23}$$

3.3 Response of a 3D *Su* RNN trained by two vectors to a novel external stimulus

As mentioned in the Introduction, the RNNs considered here are deemed to be used as building blocks in a larger memory framework, where different parts (individual networks) will interact with each other. Therefore, not only the network learning capabilities are of interest, but likewise its ability to recognize different stimuli and to complete incorrect stimulus patterns.

Let us consider a 3D *Su* network trained by two linearly independent vectors a and b. This type of network has been studied numerically in Kühn et al. (2007) and Cruse and Hübner (2008). Here we generalize and extend those results.

The learnt coupling matrix is given by (20). We apply to the network a novel stimulus e that does not satisfy the basic equation. Due to the *Su* network constraint, to obtain a dynamical response some of the elements of the novel stimulus should be equal to zero, e.g., $e = (0, 0, e_3)$. Then the network response follows from the theorem:

Theorem 3 *Assume that a 3D Su network has been previously trained by two linearly independent vectors a and b satisfying the basic equation. Then a novel incomplete external activation e is given to the network.*

(i) If $e = (0, e_2, e_3)^{\mathrm{T}}$, *and* $a_3 b_2 \neq a_2 b_3$ *then independently on the initial conditions the network state relaxes to*

$$x_1 = -B_2 e_2 - B_3 e_3, \; x_2 = e_2, \; x_3 = e_3, \tag{24}$$

where the constant $B_{2,3}$ *are given by the basic equation* ($B_1 \equiv 1$) *associated with the vectors a and b.*
For $a_3 b_2 = a_2 b_3$ *the network shows no dynamics:* $x(t) = (x_1^0, e_2, e_3)^{\mathrm{T}}$.

(ii) If $e = (0, 0, e_3)^{\mathrm{T}}$, *then the network state relaxes to*

$$x_1 = -C_2 B_2 - e_3 B_3, \; x_2 = C_2, \; x_3 = e_3, \quad C_2 = \frac{x_2^0 + B_2(e_3 B_3 - x_1^0)}{1 + B_2^2}, \tag{25}$$

where $x_{1,2}^0$ *are the network initial conditions.*

The proof is given in Appendix C.

From Theorem 3 we can derive the following statement concerning the memory content of the *Su* network. If the novel stimulus e has one or two zero components, then the output of the corresponding units evolves in time to a steady state such that the final network state always fulfills the basic equation. Indeed, using (24) for one zero or (25) for two zeros in the novel activation e we can verify that $\langle B, x_\infty \rangle = 0$. Thus the 3D network trained by two linearly independent vectors can perform algebraic calculations, or it can be used for pattern completion.

3.4 Damping factors ($n = 3$)

Let us consider a 3D *Su* network trained by two vectors a and b. The weights on the diagonal of the *learnt* coupling matrix (20) can be interpreted as damping elements. Then using that matrix we can formally construct a new damped matrix as follows

$$W_{\mathrm{damped}} = \begin{pmatrix} \frac{d_1}{1+d_1} & \frac{\alpha_1\alpha_2+\beta_1\beta_2}{(1+d_1)h_1} & \frac{\alpha_1\alpha_3+\beta_1\beta_3}{(1+d_1)h_1} \\ \frac{\alpha_2\alpha_1+\beta_2\beta_1}{(1+d_1)h_2} & \frac{d_2}{1+d_2} & \frac{\alpha_2\alpha_3+\beta_2\beta_3}{(1+d_2)h_2} \\ \frac{\alpha_3\alpha_1+\beta_3\beta_1}{(1+d_3)h_3} & \frac{\alpha_3\alpha_2+\beta_3\beta_2}{(1+d_3)h_3} & \frac{d_3}{1+d_3} \end{pmatrix}, \tag{26}$$

where $h_i = 1 - \alpha_i^2 - \beta_i^2$ and d_i are the so-called damping factors. We immediately see that for $d_i = (\alpha_i^2 + \beta_i^2)/h_i$ the damped matrix (26) is reduced to W_∞ as defined by (20), in other words, the training procedure (9) produces the damping factors $(\alpha_i^2+\beta_i^2)/h_i$. We note that (26) can be easily extended to the case of an nD network.

Clearly, for a set of arbitrary chosen damping factors ($d_i \neq -1$), for any vector x located in the attractor plane we have $W_{\mathrm{damped}} x = x$, which means that the damping factors do not affect the existence of the solution (interpreted as memory contents). However, their adjustment leads to a change in the relaxation provoked by a novel stimulus, as the following theorem states:

Theorem 4 *Assume that a novel incomplete stimulus e is applied to a 3D Su network with the modified coupling matrix* (26) *and* $d_{1,2} > 0$.

(i) If $e = (0, e_2, e_3)^{\mathrm{T}}$ *and* $a_3 b_2 \neq a_2 b_3$ *then independently on the initial conditions the network state relaxes to* (24). *Moreover, the relaxation rate is controlled by* $d_1/(1 + d_1)$.

(ii) If $e = (0, 0, e_3)^{\mathrm{T}}$, *then starting form zero initial conditions the network state relaxes to*

$$x_1 = \frac{-B_3 e_3 (1 + d_2)}{2 + d_1 + d_2}, \quad x_2 = \frac{-B_3 e_3 (1 + d_1)}{B_2 (2 + d_1 + d_2)}, \quad x_3 = e_3, \tag{27}$$

with the relaxation rate given by $\lambda_3 = (d_1 d_2 - 1)/(1 + d_1)(1 + d_2)$. *Moreover, the relaxation trajectory on the plane* $x_3 = e_3$ *follows the straight line*

$$x_1(t) = \frac{1 + d_2}{1 + d_1} B_2 x_2(t). \tag{28}$$

The proof is given in Appendix D.

Theorem 4 shows that, for a novel activation in the form $e = (0, e_2, e_3)$, damping factors affect only the relaxation rate, while the final network output is still given by (24), i.e., the network finds a solution satisfying the basic equation. The maximal relaxation velocity is achieved for $d_1 \to 0$.

For a novel activation in the form $e = (0, 0, e_3)$, damping factors affect both the relaxation rate and the final output of the network. For arbitrary (positive) values of the damping factors the network finds a solution satisfying the basic

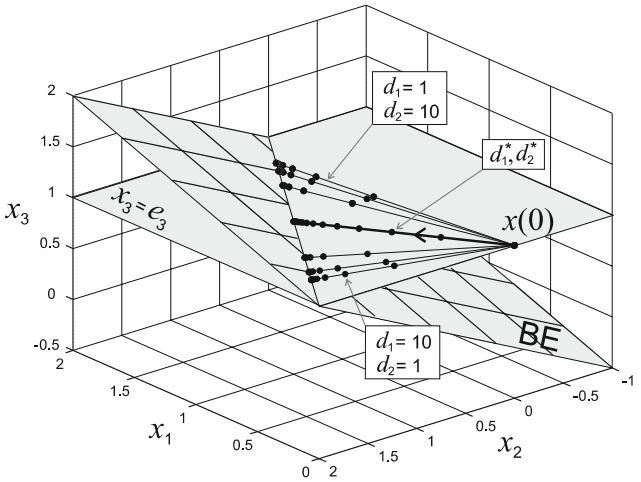

Fig. 2 Response of a 3D *Su* RNN trained by two vectors $a = (1, 3, 2)^{\mathrm{T}}$ and $b = (1, 1, 1)^{\mathrm{T}}$ to a novel stimulus $e = (0, 0, 1)$ for different values of the damping factors $d_{1,2}$. All trajectories start from the point (0, 0, 1) (marked by $x(0)$) and lie in the plane $x_3 = 1$ (marked by $x_3 = e_3$) given by the internal network constraint. According to (27) and (28) they follow straight lines tending to the plane (marked by BE) given by the basic Eq. (22). The *thick solid line* corresponds to the original coupling matrix with the damping factors $d^*_{1,2} = (\alpha^2_{1,2} + \beta^2_{1,2})/h_{1,2}$. It is orthogonal to the line obtained by the intersection of the basic plane and the plane $x_3 = e_3$

equation. Indeed, from (27) one can immediately see that $\langle B, x_\infty \rangle = 0$. However, the new solution in general differs from the original solution (25). The maximal velocity of the relaxation process is achieved for $d_1 d_2 = 1$, then $\lambda_3 = 0$ and the network jumps in a single step to the final state (27).

For $d_{1,2} = (\alpha^2_{1,2} + \beta^2_{1,2})/h_{1,2}$ using (28) one can show that the relaxation trajectory is orthogonal to the intersection of the basic plane and the plane given by $x_3 = e_3$.

To illustrate the results we have trained a 3D *Su* network by two vectors $a = (1, 3, 2)^{\mathrm{T}}$ and $b = (1, 1, 1)^{\mathrm{T}}$. In accordance with Theorem 2 the training process leads to the coupling matrix given by (23). Then, as defined by (26), we construct the damped matrix with different damping factors. Once a new damped matrix is obtained, we apply to the network a novel stimulus $e = (0, 0, 1)$, which does not satisfy the basic Eq. (22). According to Theorem 4 the network output will relax to a new state satisfying the basic equation.

Figure 2 shows trajectories in the network phase space for several values of $d_{1,2}$. The thick solid line corresponds to the original coupling matrix ($d_{1,2} = (\alpha^2_{1,2} + \beta^2_{1,2})/h_{1,2}$). The other six trajectories correspond to $(d_1, d_2) = \{(10,1), (7,1), (4,1), (1,4), (1,7), (1,10)\}$. As predicted, the trajectory obtained for the original coupling matrix (23) is orthogonal to the intersection of the basic plane and the plane given by $x_3 = e_3$. Thus the learning process given by (9) converges to the coupling matrix such that a novel stimulus will be processed by the network in a way that the network state will relax to the basic equation by the shortest trajectory. However, we note that setting damping factors such that $(1 + d_1) = B_2^2(1+d_2)$ we also obtain a trajectory going straightforwardly (orthogonal) from the initial perturbation to the basic plane. Moreover, minimizing also $|\lambda_3|$ (see Theorem 4), we obtain the optimal coupling matrix. For the basic Eq. (22) this can be obtained by setting $d_1 = d_2 = 1$. Then the network will relax to the basic equation (find a solution to the pattern completion problem) in just a single time iteration.

4 Dynamic situations: convergence of the network training procedure

In the previous section we considered learning of static situations. Let us now study the convergence properties of the network learning essentially dynamic situations. Then the training process is described by the rule (10). As we show below this leads to a more complex network behavior.

We assume that the training is done by exposing the RNN to an external input stimulus composed of periodically repeated sequences of vectors $a_i \in \mathbb{R}^n$

$$\xi(t) = a_1, a_2, \ldots, a_p, a_1, a_2, \ldots, a_p, a_1, a_2, \ldots, a_p \ldots, \tag{29}$$

where p is the period of the training stimulus; for example, a simple static situation corresponds to $p = 1$ and (29) is reduced to $\xi(t) = a_1$.

In general p is defined by the stimulus complexity (the external world) and can be rather arbitrary. Thus the network may receive as training input a stimulus with period shorter or equal to the number of units in the network ($p \leq n$), or during learning the network can be exposed to a sequence of vectors whose number exceeds the network order ($p > n$). By analogy to linear algebra we shall call the former situation underdetermined and the latter overdetermined. As we show below the training in these two cases may lead to essentially different results.

4.1 Network training: underdetermined case ($p \leq n$)

First we consider the case when $p \leq n$ and $a_1, \ldots, a_p$ are nonzero orthogonal vectors in $\mathbb{R}^n$. Then the following theorem on the convergence of the network learning process holds:

Theorem 5 *Assume that an n-unit RNN is trained by a periodic stimulus* (29) *composed of nonzero orthogonal vectors* $a_1, \ldots, a_p$ *(*$p \leq n$*) with the learning rate satisfying*

$$0 < \varepsilon < \min \left\{ \frac{2}{\|a_1\|^2}, \frac{2}{\|a_2\|^2}, \ldots, \frac{2}{\|a_p\|^2} \right\}. \tag{30}$$

Then for any initial conditions $W(0)$ the learning process given by (10) converges to the coupling matrix

$$W_\infty \equiv \lim_{t\to\infty} W(t) = \widetilde{W} + M_p, \tag{31}$$

where $\widetilde{W}$ is a constant matrix defined by the initial conditions and given in Appendix E; and

$$M_p = \sum_{i=1}^{p} \frac{a_{i+1}a_i^{\mathrm{T}}}{\|a_i\|^2}, \quad \text{with } a_{p+1} \equiv a_1. \tag{32}$$

Particularly, for (i) $p = n$ or (ii) $p < n$ but with $W(0) = 0$

$$W_\infty = M_p. \tag{33}$$

The proof of the theorem is given in Appendix E.

Theorem 5 states not only the convergence but it also provides the upper limit of the learning rate when the convergence still exists. For any set of nonzero vectors $\{a_i\}$ we can select the learning rate in such a way that the training process will always converge. From the implementation point of view an adjustment of the learning rate may improve the learning performance (speed up convergence).

As earlier noted, the requirement on orthogonality of the training stimulus in Theorem 5 is hardly satisfied by real-world stimuli. To extend our results we relax the condition on the orthogonal structure of the stimulus providing the following theorem:

Theorem 6 *Assume that an n-unit RNN is trained by a periodic stimulus (29) composed of linearly independent vectors $a_1, a_2, \ldots, a_p$ ($p \le n$) and that the learning process (10) starting from zero initial condition $W(0) = 0$ converges to the coupling matrix W_∞. Then*

$$W_\infty = (a_2, a_3, \ldots, a_p, a_1, \overbrace{0, \ldots, 0}^{n-p}) \times (a_1, \ldots, a_p, \alpha_1, \ldots, \alpha_{n-p})^{-1}, \tag{34}$$

where $0 \in \mathbb{R}^n$ is the zero vector and $\alpha_1, \ldots, \alpha_{n-p}$ are auxiliary nonzero linearly independent vectors such that the space spanned by $\{\alpha_1, \ldots, \alpha_{n-p}\}$ is orthogonal to the space spanned by $\{a_1, \ldots, a_p\}$.

The proof of the theorem is given in Appendix F.

Remark 2 For $p = n$ the resulting coupling matrix (34) is reduced to

$$W_\infty = (a_2, a_3, \ldots, a_n, a_1)\,(a_1, a_2, \ldots, a_n)^{-1}. \tag{35}$$

4.2 Network training: overdetermined case ($p > n$)

Let us now assume that the network of n units is subject to learn an external stimulus whose period is longer than the number of units in the network, i.e., $p > n$. Then we have:

Theorem 7 *Assume that an n-unit RNN is trained by a periodic stimulus (29) composed of vectors $a_1, a_2, \ldots, a_p$ ($p > n$) and that the learning process (10) starting from zero initial condition $W(0) = 0$ converges to the coupling matrix W_∞. Let $\{a_1, a_2, \ldots, a_n\}$ be the linearly independent subset of the training stimulus. Then*

$$W_\infty = (a_2, a_3, \ldots, a_n, a_{n+1})\,(a_1, a_2, \ldots, a_n)^{-1} \tag{36}$$

with

$$(W_\infty)^p = I \quad \text{and } a_{n+1} = W_\infty a_n, \ldots, a_p = W_\infty a_{p-1} \tag{37}$$

satisfied.

The theorem proof is given in Appendix G.

Note that in contrast to the underdetermined case considered above the learning convergence is not always possible here.

4.3 Examples of stimulus learning: underdetermined case ($p \le n$)

Let us now give a few examples illustrating the above stated theorems. In numerical simulations we use an RNN composed of three suppression units (Fig. 1a, c), though the results can be extended to the *Mu* network. In the underdetermined case such a network can be trained either by a static stimulus (one vector, $p = 1$) or by dynamic stimuli with period two (two vectors, $p = 2$) or period three (three vectors, $p = 3$).

4.3.1 Network training by one stimulus vector

We begin with training the network by a single vector. Strictly speaking this case can be ascribed to a static situation. Indeed, we supply to the network a *constant* input vector $\xi(t) = a$ and allow the coupling matrix to evolve. However, for completeness we also show this case here.

Starting from zero initial conditions $W(0) = 0$ the resulting trained coupling matrix is given by Kühn et al. (2007)

$$W_\infty = \frac{aa^{\mathrm{T}}}{\|a\|^2}. \tag{38}$$

The same result is obtained from Theorem 5. Indeed, setting $p = 1$ in (32) and using (33) we end up with (38). The same matrix is obtained from Theorem 2 using $c_1 = a$ and $r \equiv 1$.

To illustrate the training process let us set $a = (1, 3, 2)^{\mathrm{T}}$ and assume zero initial conditions $W(0) = 0$. Then using (38) the resulting trained matrix is

$$W_{1\mathrm{v}} = \frac{1}{14}\begin{pmatrix} 1 & 3 & 2 \\ 3 & 9 & 6 \\ 2 & 6 & 4 \end{pmatrix}. \tag{39}$$

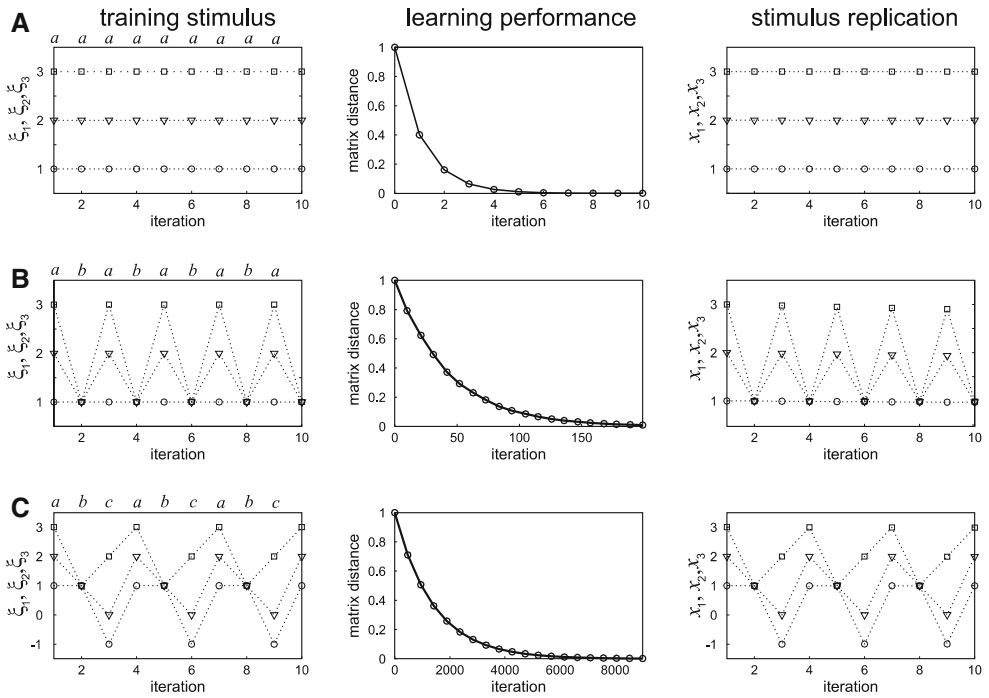

Fig. 3 Examples of stimulus learning and replication by RNN composed of three suppression units in the underdetermined case. **a** Network training by the constant stimulus vector $a = (1, 3, 2)^{\mathrm{T}}$. **b** Network training by alternating (period-two) sequence of vectors $a = (1, 3, 2)^{\mathrm{T}}$ and $b = (1, 1, 1)^{\mathrm{T}}$. **c** Network training by period-three stimulus composed of vectors $a = (1, 3, 2)^{\mathrm{T}}$, $b = (1, 1, 1)^{\mathrm{T}}$, and $c = (-1, 2, 0)^{\mathrm{T}}$. The learning rate has been fixed to $\varepsilon = 0.1$ for all three cases

Figure 3a (left and center panels) shows the training stimulus and the time evolution of the matrix distance (11) of the coupling matrix $W(t)$ to the theoretically predicted trained matrix (39). During the training the coupling matrix exponentially converges to (39) and in a few steps (at the seventh iteration) the error decreases below 1%. Then the external input can be withdrawn $\xi = 0$, and the RNN will replicate the previously learnt stimulus (Fig. 3a, right panel).

4.3.2 Network training by two alternating input vectors

Let us now consider the case in which, during training, several different vectors are presented to the three-unit network. As mentioned above this kind of training can be considered as *dynamic* in contrast to the *static* situation discussed in Sect. 4.3.1, since now the network activation varies in time.

In the simplest case we train the network with two vectors, say a and b (assuming they are linearly independent). Then using Theorem 6 from (34) we have

$$W_{\infty} = (b, a, 0)\,(a, b, \alpha)^{-1}, \tag{40}$$

where α can be chosen as

$$\alpha = \left(\begin{vmatrix} a_2 & a_3 \\ b_2 & b_3 \end{vmatrix}, \begin{vmatrix} a_3 & a_1 \\ b_3 & b_1 \end{vmatrix}, \begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix} \right)^{\mathrm{T}}.$$

We also note that, if the training vectors a and b satisfy the basic equation, i.e., $\langle B, a\rangle = 0$ and $\langle B, b\rangle = 0$, then $\alpha = B$.

Using $a = (1, 3, 2)^{\mathrm{T}}$ and $b = (1, 1, 1)^{\mathrm{T}}$ we obtain $\alpha = (1, 1, -2)^{\mathrm{T}}$ and finally

$$W_{2\mathrm{v}} = \frac{1}{6} \begin{pmatrix} 5 & -1 & 2 \\ 21 & -9 & 6 \\ 13 & -5 & 4 \end{pmatrix}. \tag{41}$$

Figure 3b shows the learning dynamics. As in the case of the training by a constant input vector, the coupling matrix also converges exponentially to the predicted matrix. However, the convergence rate is lower. To obtain an error below 1% of the initial intermatrix distance, 196 iteration steps are required.

4.3.3 Learning period-three stimulus

Let us now train the network of three units by a stimulus of period 3, i.e., $p = 3$ and $\xi = abcabcabc\ldots$ Using Remark 2 to Theorem 6 from (35) we have

$$W_{\infty} = (b, c, a)(a, b, c)^{-1}. \tag{42}$$

Setting as $a = (1, 3, 2)^{\mathrm{T}}$, $b = (1, 1, 1)^{\mathrm{T}}$, and $c = (-1, 2, 0)^{\mathrm{T}}$ we obtain

$$W_{3\mathrm{v}} = \begin{pmatrix} -5 & -2 & 6 \\ 9 & 6 & -13 \\ 0 & 1 & -1 \end{pmatrix}. \tag{43}$$

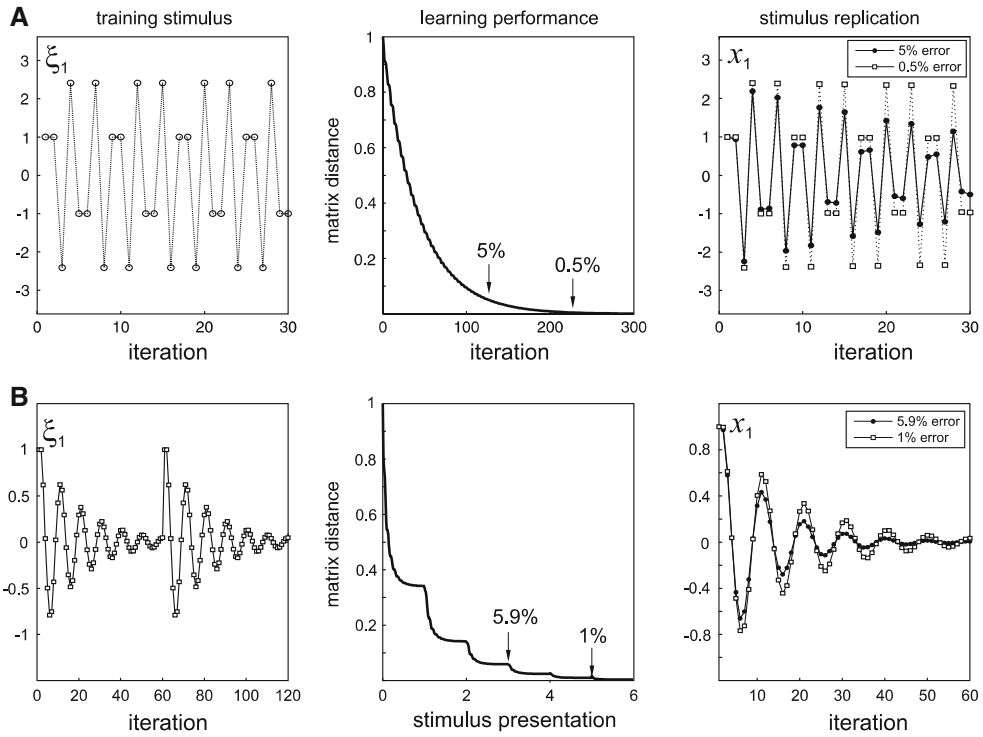

Fig. 4 Examples of learning and replication of dynamical stimuli by RNN consisting of two suppression units in the overdetermined case (i.e., stimulus period p exceeds the number of units in the network n). **a** Learning period of eight oscillations generated by the discrete counterpart of the harmonic oscillator (44). *Left panel*: first 30 vectors (only ξ_1 projection is shown) used for the network training. *Middle panel*: learning performance, i.e., dynamics of the distance of the coupling matrix $W(t)$ to the theoretically predicted matrix (47). *Right panel*: stimulus replication. Once the learning has been finished with the corresponding criteria (5% or 0.5% error) we use the network to replicate the stimulus. For 5% error the network output strongly deviates from the presented stimulus, whereas for 0.5% the RNN reproduces the stimulus fairly well. The learning rate was fixed at $\varepsilon = 0.05$. **b** The same as in (**a**) but now the network is subjected to learn decaying oscillations. We use the stimulus of 60 vectors long and present it to the network several times. The learning gradually improves each time we present the stimulus. After five presentations of the stimulus the learning reaches high quality and then the RNN replicates well the stimulus. The learning rate was fixed at $\varepsilon = 0.5$

Figure 3c shows the learning process dynamics. To obtain an error below 1% of the initial matrix distance, 6450 iteration steps are required.

Thus, the longer the stimulus period is, the slower the learning process is.

4.4 Examples of stimulus learning: overdetermined case ($p > n$)

To illustrate the stimulus learning in the overdetermined case we use a network of two coupled *Su* units ($n = 2$). As a physical model to learn we select the linear damped oscillator $\ddot{x} + r\dot{x} + \omega^2 x = 0$, where r is the damping constant and ω is the oscillation frequency. Direct discretization of the harmonic oscillator equation yields

$$X(t+1) = W_{\mathrm{LO}} X(t)$$
$$W_{\mathrm{LO}} = \begin{pmatrix} 1 & 1 \\ -\omega^2 & (1 - r - \omega^2) \end{pmatrix}. \tag{44}$$

Note that in general the solution of (44) may differ from the solution of its continuous counterpart.

4.4.1 Periodic oscillations

For $r = 0$, (44) with appropriate ω produces a periodic oscillation (in terms of a periodic sequence of 2D vectors X) whose period depends on ω. Using (37) from Theorem 7 for a solution of period p we have the following condition on ω

$$\det\left(W_{\mathrm{LO}}^{p}(\omega) - I\right) = 0. \tag{45}$$

For instance, for $p = 8$ from (44) and (45) we have

$$((\omega^2 - 4)\omega^2 + 2)^2(\omega^2 - 2)^2(\omega^2 - 4)\omega^2 = 0, \tag{46}$$

whose two solutions $\omega^2 = 2 \pm \sqrt{2}$ correspond to two different period-8 trajectories. Using either of them we can generate dynamical period-8 stimulus vectors $a_1, \ldots, a_8$. Presenting such a stimulus to the RNN, according to Theorem 7 the learning process should converge to the coupling matrix

$$W_{\mathrm{osc}} = (a_2, a_3)(a_1, a_2)^{-1}. \tag{47}$$

Using $a_1 = (1, 0)^{\mathrm{T}}$, $a_2 = (1, -2 - \sqrt{2})^{\mathrm{T}}$, and $a_3 = (-1 - \sqrt{2}, 2 + 2\sqrt{2})^{\mathrm{T}}$ we indeed arrive at

$$W_{8\mathrm{v}} = \begin{pmatrix} 1 & 1 \\ -2 - \sqrt{2} & -1 - \sqrt{2} \end{pmatrix}. \tag{48}$$

Figure 4a (left panel) shows the external stimulus of period 8 which has been applied to the network for training. The stimulus learning process, similarly to what we observed in the underdetermined case, exponentially converges to the theoretically predicted coupling matrix (48) (Fig. 4a, center panel). The intermatrix distance decreases below 5% in 127 iterations and in 227 iterations the matrix error becomes less than 0.5%. Once the external stimulus has been learnt (stored) we can use the network to replicate the stimulus. Figure 4a (right panel) illustrates the network output for two conditions to truncate the learning process for different error thresholds. With 5% error the output significantly diverges from the original stimulus already in the second period (i.e., $8 < t \leq 16$). However, with the more accurate (longer) learning (error 0.5%) the network reproduces the stimulus with high precision.

4.4.2 Damped oscillations

Let us now consider the damped oscillator (44) (i.e., $r > 0$). Now, strictly speaking, the stimulus is not periodic, since the oscillation amplitude decays in time. Nevertheless, the network can learn the stimulus, and Theorem 7 can be applied. To illustrate this case we use $\omega^2 = (3 - \sqrt{5})/2$ (which corresponds to period-10 undamped oscillations) and $r = 0.1$ to generate the training stimulus. Then we use the first three vectors a_1, a_2, and a_3 to evaluate the theoretical prediction for the learning convergence given by (47), which yields

$$W_{\text{damped oscill}} = \begin{pmatrix} 1 & 1 \\ -0.382 & 0.518 \end{pmatrix}. \tag{49}$$

Numerical simulation (Fig. 4b) indeed shows that the learning process converges to the coupling matrix (49). However, depending on the learning rate ε and the damping constant r, the learning may not converge in just a single stimulus presentation. This happens when the learning time scale is shorter than the scale of the oscillation decay, hence the network "has no time" to learn the stimulus, which disappears too fast. In this case, a solution to train the network is to present the same stimulus several times. In Fig. 4b we truncated the stimulus to 60 vectors; the left panel shows two stimulus presentations (epochs). At each stimulus presentation the network gradually improves the coupling matrix, approaching the theoretically predicted one (49). After three stimulus presentations the error falls below 6% and five presentations results in 1% error. The latter precision is sufficient to obtain quite a good stimulus replication (Fig. 4b, right panel).

Finally we note that nD network can learn the dynamical situations given by nD linear maps (i.e., loosely speaking by any linear differential equation).

5 Dynamic situations: response of trained IC-unit networks to a novel external stimulus

Let us consider an RNN of IC units when the learning process has been finished, i.e., the network now works in the operational phase with $W = \text{const}$. Then a novel, in general arbitrary, stimulus is given to the network, which provokes a response of the network state $x(t)$, whose dynamics we shall now describe.

For convenience we shall ascribe to the previously performed learning phase $t < 0$ (past) and to the operational phase $t > 0$ (future). Then the external network activation can be represented by

$$f(t) = \begin{cases} \xi(t), & t < 0 \\ e, & t \geq 0 \end{cases}, \tag{50}$$

where $\xi(t)$ is the learnt stimulus (e.g., a or ab), which fulfills the basic equation, and e is an arbitrary (including $e = 0$) postlearning network activation. Then the network response $x(t > 0)$ is expected to evolve in time to a new constant value (or diverge in time) according to (6). To complete the problem we also have to specify the initial condition for the network state, i.e., $x(t = 0) = x^0$. Two possibilities to define x^0 can be considered: (i) the so-called continuous case when the initial network state in the operational phase is the same (continues) as the state at the end of the learning phase, i.e., $x^0 = \xi(0)$, or (ii) the network state being reset, i.e., $x(0) = 0$.

If after learning we switch off the external activation from all units (i.e., $e = 0$) in the continuous case the network will replicate the previously learnt stimulus, e.g., $x(t > 0) = a$ for the training with single vector a, or $x(t > 0) = ababab\ldots$ for the training with two alternating vectors a and b (Fig. 3, right panels). Obviously, for $e = 0$ and $x(0) = 0$ (resetting) the network will stay at rest $x(t > 0) = 0$. However, for a nonzero novel stimulus ($\|e\| > 0$), the network will relax to a new state, in general different for *Su* and *Mu* networks.

5.1 Relaxation dynamics of *Su* networks ($n = 3$)

In the case of an RNN composed of *Su* units we have constraints $x_i(t > 0) = e_i$ for all $e_i \neq 0$. Thus only the units with zero external activations are allowed to evolve in time, while the others instantaneously (in one time step) adopt the new activation values. If nonzero activations are applied to all units then the network shows no dynamics at all. Thus we assume that the novel stimulus is incomplete, i.e., it has zero (empty) components. In this section for the sake of simplicity, we limit our analysis to the case of 3D networks. Moreover,

without loss of generality we assume that zero components are in the first elements of the novel stimulus vector e.

5.1.1 Stimulus completion

First, for convenience, let us introduce auxiliary parameters that can be evaluated using the training vectors a and b

$$\begin{aligned} \Gamma &= \|a\|^2\|b\|^2 - \langle a, b\rangle^2, \\ \Delta_{ij} &= a_i b_j \|a\|^2 + a_j b_i \|b\|^2 - (a_i a_j + b_i b_j)\langle a, b\rangle. \end{aligned} \tag{51}$$

We note that, for linearly independent a and b, $\Gamma > 0$.

The response of a trained RNN to a novel incomplete stimulus follows from the theorem:

Theorem 8 *Assume that a 3D Su network has been previously trained by a stimulus $\xi(t)$ satisfying the basic equation $\langle B, \xi(t)\rangle = 0$. Then a novel constant stimulus vector e with at least one zero component is applied.*

(1a) For the network trained by one vector ($\xi = a$) and $e = (0, e_2, e_3)$, the network state always relaxes to

$$x_1 = \frac{e_2 a_2 + e_3 a_3}{\|a\|^2 - a_1^2} a_1, \; x_2 = e_2, \; x_3 = e_3. \tag{52}$$

(1b) For the network trained by two linearly independent vectors ($\xi = abab\ldots$) and $e = (0, e_2, e_3)$ the network dynamics depends on the following condition

$$\left| \frac{\Delta_{11}}{\Gamma} \right| < 1. \tag{53}$$

If (53) is satisfied then the network state relaxes to

$$x_1 = \frac{\Delta_{12} e_2 + \Delta_{13} e_3}{\Gamma - \Delta_{11}}, \; x_2 = e_2, \; x_3 = e_3, \tag{54}$$

and diverges otherwise.

(2a) For the network trained by one vector ($\xi = a$) and $e = (0, 0, e_3)$, the network state always relaxes to

$$x = \frac{e_3}{a_3} a. \tag{55}$$

(2b) For the network trained by two linearly independent vectors ($\xi = abab\ldots$), and $e = (0, 0, e_3)$, the network dynamics depends on the following condition

$$\left| \frac{2(a_2 b_1 - a_1 b_2)^2}{\Delta_{33} \pm \sqrt{\Delta_{33}^2 + 4(a_2 b_1 - a_1 b_2)^2 \Gamma}} \right| < 1. \tag{56}$$

If (56) is satisfied then the network state relaxes to

$$x = \frac{e_3}{a_3 + b_3}(a + b), \tag{57}$$

and diverges otherwise.

Remark 3 The final network state (given for the corresponding case by (52), (54), (55), and (57)) does not depend on the initial network state x^0. The latter means that the network behavior is robust to internal perturbations.

The proof is given in Appendix H.

Response of an RNN trained by one vector Let us first consider the case of an RNN trained by one vector a. According to Theorem 8 any perturbation of the network state exponentially decays to a fixed point. This fixed point, however, may not belong to the plane defined by the basic equation.

If only one of the three components of the novel external stimuli e is zero, then the network will relax to a state that fulfills the basic equation only when

$$\frac{e_2}{e_3} = \frac{a_2}{a_3},$$

i.e., when the new activation has the same ratio between its second and third nonzero components as that of the training vector. Indeed, in this case $x_1(t) \to a_1 e_3/a_3$ and $\langle B, x_\infty\rangle = e_3/a_3\langle B, a\rangle = 0$. Thus the network recovers the "lost" part of the stimulus.

For the case of two zero components in the novel stimulus the network always finds a new solution satisfying the basic equation. Indeed, from (55) we have $\langle B, x_\infty\rangle = e_3/a_3\langle B, a\rangle = 0$. Moreover, the new network state represents a scaled version of the stimulus a previously learnt. The scaling factor can be adjusted by tuning the nonzero component e_3 of the novel stimulus. Thus starting from arbitrary initial conditions the network completes the previously learnt pattern and provides it on the output with the scale factor controlled by the novel stimulus.

Response of an RNN trained by two vectors Let us now consider the case when the network has been trained by two (linearly independent) vectors a and b. Note that now the stimulus vectors uniquely define the plane corresponding to the basic equation.

At variance with the training by one vector, now the novel stimulus e does not necessarily lead to a relaxation of the network state to a fixed point. If (53) or (56) is not satisfied by the training stimulus (e.g., for $a = (1.5, 3, 2)^{\mathrm{T}}$ and $b = (1, 1, 1)^{\mathrm{T}}$, $\Delta_{11}/\Gamma = 25/14 > 1$) the network state diverges. This means that for network stability an appropriate (safe) network training is required.

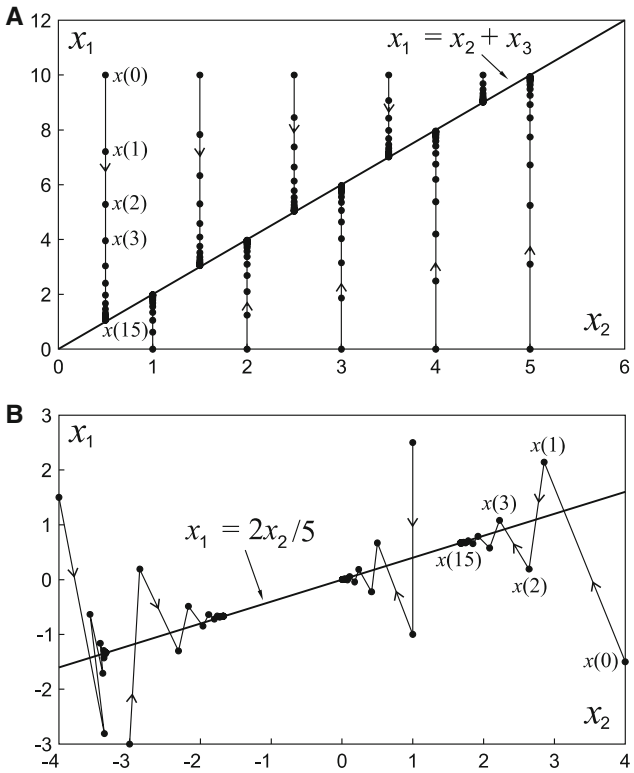

Fig. 5 Response of a trained 3D *Su* RNN to a novel stimulus. **a** Example of an algebraic task performed by the network (projection of the phase space on the plane (x_1, x_2) is shown). The network has first learnt the basic equation $x_1 = x_2 + x_3$ by using as a stimulus periodic sequence of two vectors $a = (1, 0.5, 0.5)^{\mathrm{T}}$ and $b = (1, 1.5, -0.5)^{\mathrm{T}}$. Then as a novel activation we use $(0, e_2, e_2)$. The initial value of the output on the first unit is either $x_1(0) = 0$ or $x_1(0) = 10$. Independently of the initial network state the network relaxes to the learnt basic equation $(e_2 + e_3, e_2, e_3) = (2e_2, e_2, e_2)$, thus dynamically evaluating the missing stimulus part. *Arrows* mark the direction of trajectories. **b** Example of stimulus summation and scaling. As in (**a**) the network has been trained by two vectors $a = (2, 1.5, 0.5)$ and $b = (-1, 1, -2)$. Then a novel activation in the form $(0, 0, e_3)$ is applied to the network. We use four values $e_3 = \{-1, 0, 1, 2\}$. Independently of the initial condition $x(0)$ the network sums and scales the learnt vectors $x = e_3(a+b)/(a_3+b_3)$. All trajectories end up at the line $x_1 = 2x_2/5$. *Arrows* mark the directions of the trajectories

If the novel (arbitrary) stimulus has one zero component, then the *necessary* and *sufficient* condition (Appendix H) for the network to always relax to the basic plane (i.e., $\langle B, x_\infty \rangle = 0$) is that $a_1 = b_1$ (which is always possible). Indeed, (53) is reduced to $(a_3 b_2 - a_2 b_3)^2 > 0$ and (54) to

$$x_1 = -\frac{B_2 e_2 + B_3 e_3}{B_1}, \quad x_2 = e_2, \; x_3 = e_3, \tag{58}$$

which satisfies the basic equation $\langle B, x \rangle = 0$. Therefore, the network can be used to perform simple algebraic tasks as, e.g.,

$$x_1 = x_2 + x_3. \tag{59}$$

To illustrate the algebraic capabilities of the network, first from the given algebraic Eq. (59) we have $B = (1, -1, -1)^{\mathrm{T}}$. Second we chose two vectors satisfying the basic equation with $a_1 = b_1$. Using as an example $a = (1, 0.5, 0.5)^{\mathrm{T}}$ and $b = (1, 1.5, -0.5)^{\mathrm{T}}$ we train a 3D *Su* network similarly as done in Fig. 3b. Once the coupling matrix (given in this case by (40)) has been learnt we can apply to the network novel incomplete stimuli. For illustration we use $e = (0, e_2, e_2)$ (i.e., with the same second and third components). Then according to Theorem 8 and (58), independently on the initial condition $x(0)$, the network output relaxes to $x = (2e_2, e_2, e_2)$, which means that the network finds the missing stimulus value $x_1 = 2e_2 = e_2 + e_2$. Figure 5a shows the network trajectories starting from different initial conditions for different values of the stimulus component e_2. All trajectories end up at the straight line $x_1 = x_2 + x_3$ fulfilling the task (59).

If the novel stimulus has two zero components and the training stimulus satisfies (56) then the network perturbation always relaxes to the basic plane. Moreover, the new adopted network state represents a weighted mean of the two training vectors. Again, as in the case of training by one vector, the scale factor is controlled by the nonzero stimulus component.

To illustrate this case we also use (59) as the basic equation and train the network by two vectors $a = (2, 1.5, 0.5)$ and $b = (-1, 1, -2)$. Then according to (57) the network state relaxes to $x = e_3(2/3, 5/3, -1)$, which yields $x_1 = 2x_2/5$. Figure 5b shows four network trajectories for four different values of the novel stimulus e_3 starting from different initial conditions $x(0)$ and converging to $x = e_3(2/3, 5/3, -1)$. Note that the trajectories exhibit damped oscillations (jump from one to the other side) around the attractor line $x_1 = 2x_2/5$.

5.1.2 Damping with constant input

Let us consider a network trained by a single vector a. Similar to in Sect. 3.4 the weights on the diagonal of the *learnt* coupling matrix $w_{ii} = a_i^2/\|a\|^2$ can be interpreted as damping elements and the damped matrix is given by

$$W_{\mathrm{damped}} = \begin{pmatrix} \frac{d_1}{1+d_1} & \frac{a_2 h_1}{1+d_1} & \cdots & \frac{a_n h_1}{1+d_1} \\ \frac{a_1 h_2}{1+d_2} & \frac{d_2}{1+d_2} & \cdots & \frac{a_n h_2}{1+d_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{a_1 h_n}{1+d_n} & \frac{a_2 h_n}{1+d_n} & \cdots & \frac{d_n}{1+d_n} \end{pmatrix}, \tag{60}$$

where $h_i = a_i/(\|a\|^2 - a_i^2)$. We immediately see that, for $d_i = a_i h_i$, the damped matrix (60) is reduced to W_∞ as defined by (38); in other words, the training procedure (10) produces the damping factors $a_i h_i$.

Clearly, for a set of arbitrary chosen damping factors ($d_i \neq -1$), $W_{\mathrm{damped}} a = a$, which means that the damping factors do not affect the existence of the solution (interpreted as memory contents). However, as the following theorem states,

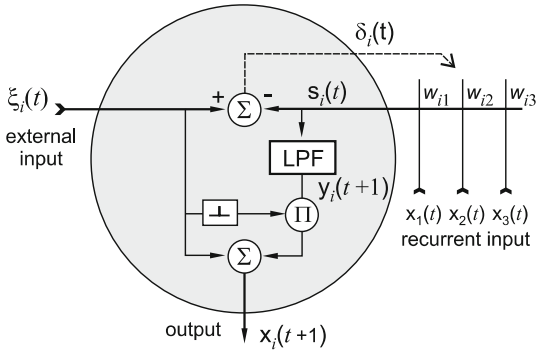

Fig. 6 Modified *Su* with the first-order low-pass filter (LPF) incorporated in the circuit

their adjustment leads to a change in the relaxation rate provoked by a perturbation of the network state:

Theorem 9 *Assume that a novel incomplete stimulus e is applied to a 3D Su network with the modified coupling matrix (60) and* $a_i \neq 0$, $d_i \geq 0$ $(i = 1, 2, 3)$.

(i) If $e = (0, e_2, e_3)^{\mathrm{T}}$ *then the network state relaxes to*

$$x_1 = (e_2 a_2 + e_3 a_3) h_1, \quad x_2 = e_2, \quad x_3 = e_3. \tag{61}$$

Moreover, the relaxation rate is given by $d_1/(1 + d_1)$, *i.e., the larger* d_1, *the slower the network approaches the stable solution.*

(ii) If $e = (0, 0, e_3)^{\mathrm{T}}$ *then the network state relaxes to*

$$x = \frac{e_3}{a_3} a \tag{62}$$

Damping factors affect the relaxation rate. Particularly, for $d_1 = d_2 = d$, *the larger d the slower the network approaches the stable solution.*

The proof of Theorem 9 is given in Appendix I.

According to the theorem the damping factors do not alter the result of the network response to a novel stimulus [compare (52) and (55) with (61) and (62), respectively]. However, the convergence rate can be adjusted by the damping factors.

5.1.3 Low-pass filter and damping elements

In several studies concerning the behavior of RNNs, instead of using simple summation units as done here, units containing blocks showing dynamical properties are used. An often applied extension is the use of a low-pass filter at the output (Fig. 6). The low-pass filter damps fast oscillations allowing the network to converge rapidly to a fixed point.

The dynamics of a first-order low-pass filter is given by

$$\tau \dot{y} = -y + I(t), \tag{63}$$

where I is the signal on the filter input, y is the filter output, and τ defines the filter time constant, i.e., the decay velocity of the filter output in response to delta-function input. Since our RNNs operate in discrete time we also discretize (63) and obtain the equation describing the dynamics of the LPF block in Fig. 6

$$y_i(t+1) = \frac{\tau - 1}{\tau} y_i(t) + \frac{1}{\tau} I_i(t). \tag{64}$$

A simple calculation shows that the behavior of an *Su* RNN based on the damped coupling matrix with damping elements d_i^{dmp} is identical to that of a network using the undamped matrix (i.e., with $d_i = 0$) but with each unit being equipped with a low-pass filter with the time constants $\tau_i = d_i^{\mathrm{dmp}} - 1$.

5.2 Relaxation dynamics of *Mu* networks ($n = 3$)

Let us now discuss the evolution of a *Mu* network under a general step-like perturbation (50). Once the training process described in Sect. 4.3.1 has been finished, the dynamics of the *Mu*-network obeys (3). Note that the new activation may not satisfy the basic Eq. (7), i.e., $\langle B, e\rangle \neq 0$, and the coupling matrix W is defined by (38).

When dealing with *Su* networks we could easily predict the evolution of the units with nonzero activation: $x_i(t > 0) = e_i$ for $e_i \neq 0$. For *Mu* networks the answer to the same question is not so trivial. Numerical simulation (not presented here) shows that, as for the *Su* network, one of the network output variables always appears to be fixed (unchanged) during the relaxation, i.e., $x_i(t > 0) = e_i$. However, there is no indication which of the units stays fixed.

First let us assume that the new stimulus vector e is given by rescaling the training stimulus $e = \rho a$ (ρ is a nonzero constant), and hence satisfies the basic equation. Then the dynamics of (3) is simply reduced to $x(t) = e$, i.e., there is no time evolution of the network state. Consequently, in the following we consider the case $e \neq \rho a$, i.e., we apply to the network a novel significantly different stimulus. Then we have the following result:

Theorem 10 *For a Mu network assume that the training vector a and the post-training stimulus vector e are either both positive* $(a, e > 0)$ *or negative* $(a, e < 0)$ *and* $e \neq \rho a$. *Let* $i \in \{1, 2, \ldots, n\}$ *be the only unit satisfying*

$$\frac{\langle a, e\rangle}{\|a\|^2} |a_i| \leq |e_i|, \quad a_i \neq 0. \tag{65}$$

Then $x_i(t > 0) = e_i$, *i.e., the output variable corresponding to this unit is fixed, while the others evolve in time according to*

$$\begin{aligned} x_j(t) &= \lambda(t) a_j \\ \lambda(t) &= \left(\frac{\langle a, e\rangle}{\|a\|^2} - \frac{e_i}{a_i}\right)\left(1 - \frac{a_i^2}{\|a\|^2}\right)^{t-1} + \frac{e_i}{a_i} \end{aligned} \tag{66}$$

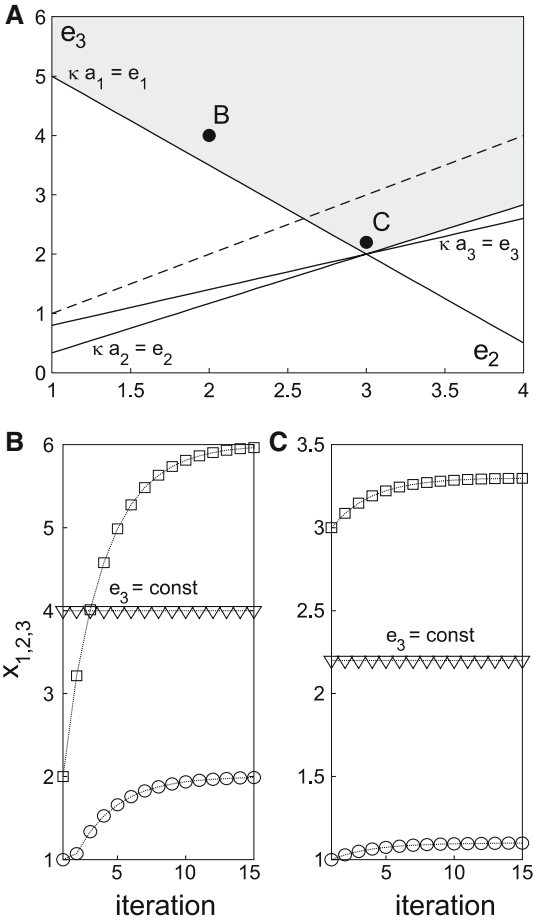

Fig. 7 Relaxation dynamics of a 3D *Mu* network. **a** Graphical solution for the constraint (68) with $a = (1, 3, 2)^{\mathrm{T}}$ and $e_1 = 1$. The *gray* area corresponds to the possible values of e_2 and e_3. The *dashed line* is the bisectrix. The *two filled circles* marked by letters B and C correspond to stimulus values used in simulations of the relaxation dynamics of the network shown in panels B and C, respectively. **b**, **c** Time evolution of $x_{1,2,3}$ (marked by *circles*, *squares*, and *triangles*, respectively). In both cases the solution converges to $x = ae_3/a_3$, satisfying the basic equation

for $j \neq i$. The final network state is given by

$$\lim_{t\to\infty} x(t) = \frac{e_i}{a_i} a. \tag{67}$$

The proof is given in Appendix J.

Theorem 10 provides the answer for the unit number whose output will be fixed in time. Note that it usually, *but not always*, corresponds to the highest element of the post-training stimulus vector. Moreover, the final network state (67) reproduces a rescaled version of the training vector a.

Let us illustrate the theorem in the 3D case considering positive training and post-training vectors ($a, e > 0$), such that the condition (65) is satisfied for the $i = 3$ unit, i.e.,

$$\kappa a_1 > e_1,\ \kappa a_2 > e_2,\ \kappa a_3 \leq e_3, \tag{68}$$

where $\kappa = \langle a, e\rangle / \|a\|^2$. As earlier, we use $a = (1, 3, 2)^{\mathrm{T}}$, and for the sake of simplicity we set $e_1 = 1$. Then Fig. 7a shows a graphical solution for the inequalities (68). We can have either $e_3 > e_2$ or $e_3 < e_2$ (above or below the bisector line). In both cases, according to Theorem 10, the third unit of the network will have no dynamics, i.e., $x_3(t) = e_3$. Thus as mentioned above, the unit with constant output usually, but not always, corresponds to the highest element of the post-training vector. Indeed, Fig. 7b,c confirms the theorem predictions: (i) $x_3(t > 0) = e_3$, and (ii) $x_{1,2}(t) \to e_3 a_{1,2}/a_3$, which for the parameter values used in the figure gives $x_{1,2} \to (2, 6)$ for Fig. 7b, and $x_{1,2} \to (1.1, 3.3)$ for Fig. 7c. Thus the final network state is $x = 2a$ in the first case and $x = 1.1a$ in the second case.

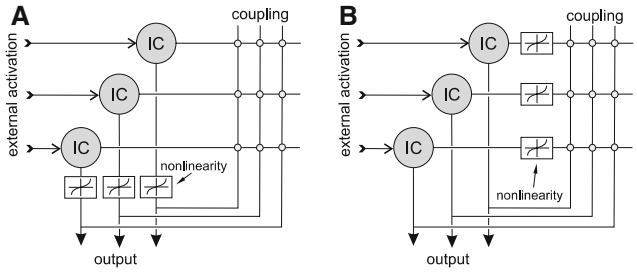

Fig. 8 IC networks with nonlinear blocks (shown as *boxes*) in the recurrent pathway (case $n = 3$). **a** The nonlinear blocks are placed before the coupling. **b** The nonlinear blocks appear after the coupling

6 IC networks with nonlinear recurrent coupling

In the previous sections we investigated the dynamical behavior of "simple" nonlinear networks, when each IC unit on its internal input receives a linear weighted sum of outputs of all units (1). In this context we may refer to such recurrent input as a linear coupling. The use of nonlinear coupling can greatly enhance the flexibility of RNNs to represent and model more complex external stimuli or situations, e.g., nonlinear algebraic relationships or pattern completion. Specific cases of nonlinear RNNs could successfully be trained (Kühn et al. 2007), but there was no general statement concerning the conditions under which such training is possible. Let us therefore investigate up to what extent an RNN can be trained when "strong" nonlinear properties are introduced in the recurrent coupling.

To generalize the network architecture shown in Fig. 1c we include nonlinear blocks into the recurrent pathway, i.e., elements whose output is a nonlinear function of the input. Figure 8 shows two possible network architectures with nonlinear blocks inserted between the unit outputs and their internal inputs. The difference between these architectures is in the order of operation, i.e., first nonlinearity and then coupling (Fig. 8a), or vice versa (Fig. 8b). Denoting the nonlinearity used in the network by $g(x)$, in the first case we have the generalized version of (1) in the form

$$s_i(t) = \sum_{k=1}^{n} w_{ik} g\left(x_k(t)\right), \tag{69}$$

whereas in the second case

$$s_i(t) = g\left(\sum_{k=1}^{n} w_{ik} x_k(t)\right). \tag{70}$$

6.1 Training the network with nonlinear blocks preceding the coupling

Let us start with the network where the nonlinear blocks are placed before the coupling (Fig. 8a). We shall consider the learning of a static stimulus, i.e., when during the training the network receives a constant external input $\xi(t) = a$. In this case, generalizing the above described learning algorithm (9) or (10) and using the constant external input, we obtain

$$W(t+1) = W(t)\left(I - \varepsilon g(a) a^{\mathrm{T}}\right) + \varepsilon a a^{\mathrm{T}}, \tag{71}$$

where $g(a) = (g(a_1), \ldots, g(a_n))^{\mathrm{T}}$. For the learning process described by (71) the following theorem holds:

Theorem 11 *Assume that network A (Fig. 8a) is trained by a constant stimulus a such that $g^{\mathrm{T}}(a)a \neq 0$ and the learning rate satisfies*

$$0 < \varepsilon < \frac{2}{g^{\mathrm{T}}(a)a}. \tag{72}$$

Then the learning process given by (71) converges. Moreover, for zero initial conditions $W(0) = 0$ we have

$$W_\infty \equiv \lim_{t\to\infty} W(t) = \frac{a a^{\mathrm{T}}}{g^{\mathrm{T}}(a)a}. \tag{73}$$

The theorem proof is given in Appendix K. We also note that for $g(x) = x$ (73) is reduced to (38).

Using Theorem 11 we see that the learning convergence can be achieved by an appropriate choice of the learning rate ε for any nonlinear function g satisfying $g(x)x > 0$. The latter condition for instance is true for odd functions like $\tanh(x)$, x^3, $\mathrm{sign}(x)$, etc.

6.2 Training the network with nonlinear blocks following the coupling

Let us now consider the case when the nonlinear blocks are included after the coupling (Fig. 8b), hence we have the following learning rule

$$W(t+1) = W(t) - \varepsilon g(W(t)a) a^{\mathrm{T}} + \varepsilon a a^{\mathrm{T}}, \tag{74}$$

where $g(W(t)a) = (g(\langle W_1, a\rangle), \ldots, g(\langle W_n, a\rangle))^{\mathrm{T}}$, with W_i being the ith row of W.

The convergence of the learning algorithm (74) is given by the following theorem.

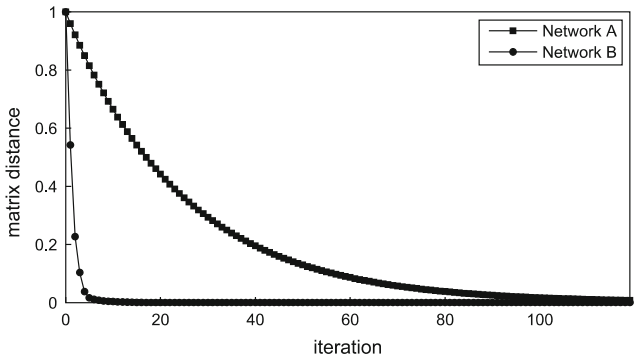

Fig. 9 Performance of learning constant stimulus by IC networks with nonlinear blocks in the recurrent pathway shown in Fig. 8 ($g(x) = x^3$, $a = (1, 3, 2)^{\mathrm{T}}$, and $\varepsilon = 0.02$)

Theorem 12 *Assume that network B (Fig. 8b) is trained by a constant stimulus a and $g(x) \in \mathbb{C}^1(\mathbb{R})$ is a monotonic function on $\mathbb{R}$, $g'(g^{-1}(a_i)) \neq 0$. If the learning rate satisfies*

$$0 < \varepsilon < \min_{1\le i\le n}\left(\frac{2}{g'(g^{-1}(a_i))\|a\|^2}\right), \tag{75}$$

then the learning process given by (74) converges. Moreover, for zero initial conditions $W(0) = 0$, we have

$$W_\infty \equiv \lim_{t\to\infty} W(t) = \frac{g^{-1}(a)a^{\mathrm{T}}}{\|a\|^2}. \tag{76}$$

The theorem proof is given in Appendix L. Again (76) is reduced to (38) for $g = x$. We also note that, if $g^{-1}(a)$ does not exist (e.g., $\tanh^{-1}(3)$), then the learning process diverges.

6.3 Simulation results

To illustrate the above stated theorems we use $g(x) = x^3$ as the nonlinearity, and $a = (1, 3, 2)^{\mathrm{T}}$ as the training stimulus.

Let us first consider the case of the nonlinearity preceding the coupling (Fig. 8a). We obtain $g(a) = (1, 27, 8)^{\mathrm{T}}$. Then from (72) we get $\varepsilon_{\max} = 1/49 \approx 0.020$ and from (73)

$$W_{\mathrm{A}} = \frac{1}{98}\begin{pmatrix} 1 & 3 & 2 \\ 3 & 9 & 6 \\ 2 & 6 & 4 \end{pmatrix}. \tag{77}$$

For network B with nonlinearity, following the coupling (Fig. 8b) using the same training vector $a = (1, 3, 2)^{\mathrm{T}}$, we find $g^{-1}(a) = a^{1/3} = (1, 3^{1/3}, 2^{1/3})$ and $g'(g^{-1}(a)) = 3(a^{1/3})^2$. Then using Theorem 12 we obtain $\varepsilon_{\max} \approx 0.023$ and

$$W_{\mathrm{B}} \approx \begin{pmatrix} 0.071 & 0.214 & 0.142 \\ 0.103 & 0.309 & 0.206 \\ 0.090 & 0.270 & 0.180 \end{pmatrix}. \tag{78}$$

Figure 9 shows the learning performance for the two network configurations shown in Fig. 8. In simulations we used

the same nonlinearity and the same learning rate. For network A (coupling follows nonlinearity, Fig. 8a) the matrix distance goes below 1% in 114 iterations, whereas for network B the same precision is achieved in 8 steps. Thus for the given nonlinearity the use of network architecture B is more beneficial.

7 Discussion

If, following Fuster (1995), one interprets the term memory to comprise not only declarative and/or procedural memory, but also one does not separate individual memory from species memory, then the task of understanding the memory does not mean less than understanding the brain. Therefore, investigation of the memory organization and functions is a challenging task. The goal of understanding memory function primarily requires solution of two basic problems. One concerns the question of how individual memory items are stored in the form of neural networks. The second question is how such memory elements may be connected to form large temporal or contextual structures. In this paper we have dealt with the first problem, assuming that situations given in the environment are represented by RNNs of as-yet unspecified structure.

Among the different situations to learn we have *static*, represented by one or more stimulus vectors that are considered to be temporarily independent, and *dynamic* situations represented by a sequence of temporarily ordered stimulus vectors; for instance, situations described by linear or nonlinear differential equations belong to the latter case.

Following the hypothesis that stimuli are stored in the interunit couplings, it has been shown that specific RNN architectures based on *Su* or *Mu* units can be used to learn and represent static and dynamic situations. Moreover, we have shown how the learning rule should be adjusted (selecting (9) or (10)) according to the situation. The obtained situation models might then be used to represent the learnt stimuli or to control behavioral output. Examples of the tasks to be solved include pattern completion, solution of simple algebraic problems, learning and finding the position of a home relatively to visible landmarks or representing a model of its own body. Most of these investigations had been previously performed based on numerical studies only. Therefore no proof concerning the generality or limits of these proposals had been given. In this paper, this gap has now been closed in such a way that many of the qualitative statements could be proven or the quantitative limits be defined.

Using the teacher forcing method and the traditional delta rule applied locally within a neuron, training RNNs is possible for situations being described by linear basic equations, by linear differential equations, and also by nonlinear versions of both types. Based on the results presented here it is shown (Cruse and Hübner 2008) that also linear MMC networks can be learned this way. Concerning the training, there is no difference between the use of *Su* or *Mu* units in the network; the difference appears in the network responses to a novel stimulus.

During the training of a RNN with linear couplings, the network learns the weight matrix W, whose damping factors can be the same or may be different for all units. Learning of both static and dynamic situations has been studied. We have shown that the learning of static situations works on a linearly independent subset of the training vectors and that the limit coupling matrix does not depend on their particular sequence. This is obviously not the case for dynamic situations where the time ordering is indeed important. Theorems 1 and 2 for the static case and Theorems 5, 6, and 7 for the dynamic case provide coupling matrices that will be formed during the learning and what are the appropriate limits for the learning rate in general, i.e., for the case of arbitrary number of units in the network and arbitrary stimulus complexity. As an example for the static case, matrix (20) shows the weights (elements of W) of a three-unit network trained statically with two vectors. Matrix (26) shows how the weights can be interpreted to include damping factors that define the relaxation characteristics of the network after a disturbance. As an example for the dynamic case, the weights are given by (38), (40), or (42) for learning a periodic sequence of one, two or three stimulus vectors, respectively. If, after the learning of the periodic sequence, the external input is then switched off, the network reproduces this input, either a constant vector or a temporal pattern (Fig. 3). We have shown that nD networks can learn a dynamic situation consisting of up to n different vectors. Moreover, the network will sequentially reproduce the learnt vectors in the order they have been shown to the network. Such network ability can be used, for instance, to store a movie, where each frame can be considered as a training vector.

Theorems 11 and 12 extend results of the learning static situations onto the case of nonlinear coupling. They provide general conditions on the type of nonlinearities, weight matrices developed during the training, and limits for the learning rate for different arrangements of the nonlinear blocks in the interunit coupling. For instance, the nonlinearity preceding the linear part of the coupling (as shown in Fig. 8a) should satisfy the condition $xg(x) > 0$, which is fulfilled by odd functions such as $g(x) = \tanh(x)$. The nonlinearity following the coupling (Fig. 8b) should satisfy the condition $g'(g^{-1}(x)) > 0$. We have shown that placing the nonlinearities at the output of the units requires more learning steps to converge compared to placing the same nonlinear functions at the internal input of the units. However, the latter network may cause problems if the inverse of the nonlinear function does not exist for the whole range of possible values of the stimulus vector, e.g., $\tanh^{-1}(x)$ exists only for $|x| < 1$.

Generally, artificial neural networks are investigated in two versions using either (i) simple summation units, or (ii) units equipped with dynamic properties, usually a low-pass filter. The latter cases are eventually called continuous-time recurrent neural networks (Beer 2006). Steinkühler and Cruse (1998) have indicated that using summation units with appropriate positive weights at the diagonal of the coupling matrix may endow the network with low-pass-filter-like properties. Using the units with low-pass-filter blocks (Fig. 6) here we have shown that indeed damping factors d_i in *Su* units correspond to the decaying time constants $\tau_i = d_i - 1$ of the first-order low-pass filter introduced at each unit. Recently, the inclusion of such low-pass filter units instead of setting the diagonal weights to zero was shown to be advantageous in the case of a network used for landmark navigation (Cruse and Hübner 2008).

Apart from the learning situation models, an important question is how such networks behave after the learning process has been finished. If a novel stimulus is then applied, the network will relax to a new state. As has been discussed by Kühn et al. (2007) these networks can be used to represent short- and long-term memories. They can be used to perform simple algebraic tasks or more abstract pattern separation such as ABB or ABA. Applying the property of pattern completion these representations could be used for reconstruction of missing inputs, or for pattern recognition. For example the missing value of x_1 can be recovered given x_2 and x_3. These properties have now been investigated in quantitative detail for the case of three-unit networks.

Regarding the static case for a network consisting of three *Su* units and being trained by two linearly independent vectors, Theorem 3 shows that after a disturbance the network relaxes to the attractor plane defined by the basic equation. Therefore, the network can be used to compute simple algebraic equations. Theorem 4 provides information concerning the relaxation dynamics. The damping factors learnt led to the relaxation trajectory being orthogonal to the plane defined by the basic equation, i.e., the relaxation follows the shortest possible path. We have also shown how one can tune the damping factors to speed up the relaxation (decrease the number of the required time steps down to a single iteration). Similarly, with respect to dynamic situations, Theorem 8 considers the response of a network with *Su* units, which has been trained by either one or two input vectors. For the network trained by one vector providing a novel incomplete input to two units that corresponds to the training vector, while setting the other stimulus component to zero, the network will restore this missing stimulus part. If two of three inputs are set to zero, in general, the new network output will represent a scaled version of the learned vector. Moreover, the scaling factor can be controlled by the nonzero input. For the network trained by two vectors, application of the incomplete novel stimulus with two zero components leads to a new state on the network output satisfying the basic equation. Furthermore, we have shown that the new state represents a scaled sum of the training vectors. If specific conditions are fulfilled by the training vectors and then one component of the novel stimulus is set to zero, the three-neuron network can be used to perform simple algebraic calculations, e.g., $x_1 = x_2 + x_3$ (Fig. 5) if some constraints (not being necessary when applying the learning procedure for the static case) are fulfilled. Using *Mu* units, in the same situation the network after disturbance will always relax to a scaled version of the training vector. One unit of the network will maintain its value during the relaxation and the conditions defining which unit will be selected are given for specific cases in Theorem 10.

Having investigated basic properties of the recurrent networks constructed by *Su* or *Mu* units, we now have a solid basis to approach the second goal when searching for a memory structure, namely how to connect different situation models in a sensible way in order to represent temporal sequences of such situation models and to search for possibilities of how such situation models could be arranged within a dynamical hierarchy.

Acknowledgements This research has been sponsored by the EU grant SPARK (FP6-2003-IST-004690), by the Spanish Ministry of Education and Science (grant FIS2007-65173, and by a Ramón y Cajal grant), and by a Santander–Complutense grant (PR41/06-15058).

Appendix A: Proof of Theorem 1

Let $V_j(t)$ denote the jth column of the matrix $W^{\mathrm{T}}(t)$. Then from (9) we have

$$V_j(t+1) = \left(I - \varepsilon\xi(t)\xi^{\mathrm{T}}(t)\right) V_j(t) + \varepsilon\xi_j(t)\xi(t). \tag{79}$$

For $p < n$ we can select $(n-p)$ nonzero orthogonal vectors $\alpha_1, \ldots, \alpha_{n-p} \in \mathbb{R}^n$ such that the space spanned by them is orthogonal to the space spanned by $\{a_1, \ldots, a_p\}$. Then we can introduce two orthogonal subspaces

$$\begin{aligned} X_1 &= \left\{k_{p+1}\alpha_1 + \cdots + k_n\alpha_{n-p} : \quad k_{p+1}, \ldots, k_n \in \mathbb{R}\right\}, \\ X_2 &= \left\{k_1 a_1 + \cdots + k_p a_p : \quad k_1, \ldots, k_p \in \mathbb{R}\right\}. \end{aligned} \tag{80}$$

Clearly, $\mathbb{R}^n$ is the direct sum of X_1 and X_2, i.e., $\mathbb{R}^n = X_1 \oplus X_2$. Using (14) we verify that

$$M_p^{\mathrm{T}} a_i = a_i, \quad M_p^{\mathrm{T}} M_p^{\mathrm{T}} = M_p^{\mathrm{T}}, \tag{81}$$

hence M_p^{T} is an orthogonal projector onto X_2. Then decomposition of $V_j \in \mathbb{R}^n$ is given by

$$V_j = V_j^1 \oplus V_j^2 \tag{82}$$

with $V_j^1 = (I - M_p^{\mathrm{T}})V_j \in X_1$ and $V_j^2 = M_p^{\mathrm{T}} V_j \in X_2$.

Denoting the jth column of the matrix M_p^{T} by M_{pj}^{T} we have

$$a_i^{\mathrm{T}} M_{pj}^{\mathrm{T}} = a_{i,j}, \quad i = 1, \ldots, p, \tag{83}$$

where $a_{i,j}$ is the jth component of the vector a_i.

Now we define a linear map L from $\mathbb{R}^n$ into itself

$$L(V_j) = \left(I - \varepsilon \xi(t) \xi^{\mathrm{T}}(t)\right) V_j. \tag{84}$$

Then for $V_j \in X_1$ we have

$$L(V_j) = V_j, \tag{85}$$

since $\xi(t) \in X_2$ and X_1 is orthogonal to X_2.

Let $V_j = k_1 a_1 + \cdots + k_p a_p \in X_2$ and, for example, $\xi(t) = a_1, a_2, \ldots$ Then from (84) we get the first two iterations

$$\begin{aligned} L^1(V_j) &= \left(I - \varepsilon a_1 a_1^{\mathrm{T}}\right)(k_1 a_1 + k_2 a_2 + \cdots + k_p a_p) \\ &= \left(1 - \varepsilon \|a_1\|^2\right) k_1 a_1 + \textstyle\sum_{i=2}^{p} k_i a_i \end{aligned}$$

and

$$\begin{aligned} &L^2(V_j) = L(L^1(V_j)) = \left(I - \varepsilon a_2 a_2^{\mathrm{T}}\right) L^1(V_j) \\ &= (1 - \varepsilon \|a_1\|^2) k_1 a_1 + (1 - \varepsilon \|a_2\|^2) k_2 a_2 + \textstyle\sum_{i=3}^{p} k_i a_i. \end{aligned}$$

Similarly, we can proceed with the other iterations. Then, in the general case, for the tth iteration and nonzero probabilities of occurrences of the training vectors $\{a_1, \ldots, a_p\}$ we obtain

$$L^t(V_j) = \sum_{i=1}^{p} \left(1 - \varepsilon \|a_i\|^2\right)^{t_i} k_i a_i \in X_2, \tag{86}$$

where t_i is the number of occurrences of the ith training vector a_i $\left(\sum t_i = t\right)$. For ε satisfying (12)

$$\lim_{t \to \infty} L^t(V_j) = 0. \tag{87}$$

Now let us consider the following map, equivalent to (79),

$$V_j(t+1) = L(V_j(t)) + \varepsilon \xi_j(t) \xi(t) \tag{88}$$

with the initial condition $V_j(0)$. Using the decomposition (82) and (85) we have

$$V_j(t+1) = V_j^1(0) + L^{t+1}(V_j^2(0)) + \sum_{k=0}^{t} L^{t-k}\left(\varepsilon \xi_j(k) \xi(k)\right).$$

Thus, if (88) in X_2 converges to $\hat{V}_j \in X_2$, then in $\mathbb{R}^n$ for any $V_j(0)$ it converges to $V_j^1(0) + \hat{V}_j$.

Using (83) and (84) we can directly verify that

$$M_{pj}^{\mathrm{T}} - L(M_{pj}^{\mathrm{T}}) = M_{pj}^{\mathrm{T}} - (I - \varepsilon \xi(t) \xi^{\mathrm{T}}(t)) M_{pj}^{\mathrm{T}} = \varepsilon \xi_j(t) \xi(t),$$

i.e., $M_{pj}^{\mathrm{T}} \in X_2$ is a steady state of the map (88) in X_2. Then representing the solution of (88) in the form $V_j(t) = M_{pj}^{\mathrm{T}} + \widetilde{V}_j(t)$ and using (87) we immediately see that $\widetilde{V}_j(t) \to 0$, i.e., $M_{pj}^{\mathrm{T}} \in X_2$ is the unique globally asymptotically stable steady state in X_2 (provided that (12) is satisfied). Thus $V_j(t) \to V_j^1(0) + M_{pj}^{\mathrm{T}}$ and hence we have

$$\lim_{t \to \infty} V(t) = V^1(0) + M_p^{\mathrm{T}}.$$

Using (82) we obtain $V^1(0) = (I - M_p^{\mathrm{T}}) V(0)$. Therefore (13) holds.

For $p = n$ we have $X_2 = \mathbb{R}^n$, which implies that $W_\infty = M_p$.

Appendix B: Proof of Theorem 2

The proof is similar to the proof of Theorem 1.

For $r < n$ we can select $(n - r)$ linearly independent vectors $\alpha_1, \ldots, \alpha_{n-r} \in \mathbb{R}^n$ such that the space spanned by them is orthogonal to the space spanned by $\{a_1, \ldots, a_p\}$. Consequently, we can introduce two orthogonal subspaces of $\mathbb{R}^n$

$$\begin{aligned} X_1 &= \{k_1 \alpha_1 + \cdots + k_{n-r} \alpha_{n-r} : \quad k_1, \ldots, k_{n-r} \in \mathbb{R}\} \\ X_2 &= \operatorname{span}\{a_1, \ldots, a_p\} \end{aligned} \tag{89}$$

Using (18) we check for any $i \in \{1, \ldots, p\}$

$$M_r^{\mathrm{T}} c_i = c_i, \quad M_r^{\mathrm{T}} M_r^{\mathrm{T}} = M_r^{\mathrm{T}}. \tag{90}$$

The latter, together with $\operatorname{span}\{a_1, \ldots, a_p\} = \operatorname{span}\{c_1, \ldots, c_r\}$, imply that M_r^{T} is an orthogonal projector onto X_2 and the decomposition of $V_j \in \mathbb{R}^n$ is given by

$$V_j = V_j^1 \oplus V_j^2 \tag{91}$$

with $V_j^1 = (I - M_r^{\mathrm{T}}) V_j \in X_1$ and $V_j^2 = M_r^{\mathrm{T}} V_j \in X_2$. In addition, since $a_i \in X_2$, we also have $M_r^{\mathrm{T}} a_i = a_i$. Then

$$a_i^{\mathrm{T}} M_{rj}^{\mathrm{T}} = a_{i,j}, \quad i = 1, \ldots, p, \tag{92}$$

where M_{rj}^{T} is the jth column of M_r^{T} and $a_{i,j}$ is the jth component of a_i. Therefore, $M_{rj}^{\mathrm{T}} \in X_2$ is a steady state of the map (88). Further, if the learning process (9) starting from initial condition $W(0)$ converges to the coupling matrix W_∞, then $W_\infty = W(0)(I - M_r) + M_r$.

For $r = n$, we have $X_2 = \mathbb{R}^n$ and consequently, for any initial conditions $W_\infty = M_r$, which completes the proof.

Appendix C: Proof of Theorem 3

The dynamics of a *Su* network can be described by the map:

$$x(t+1) = W_s x(t) \tag{93}$$

with the appropriately selected coupling matrix W_s and initial conditions $x(0) = x^0$.

(i) If the novel stimulus has the form $e = (0, e_2, e_3)^{\mathrm{T}}$, then the dynamics of the network is given by the map (93) with $x^0 = (x_1^0, e_2, e_3)^{\mathrm{T}}$ and

$$W_s = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{94}$$

where w_{ij} are the corresponding elements of the matrix (20).

First let us assume that $a_3b_2 = a_2b_3$. Then $W_s = I$, which implies that $x(t) = (x_1^0, e_2, e_3)^{\mathrm{T}}$. For $a_3b_2 \neq a_2b_3$ the eigenvalues of (94) are $(1, 1, \lambda_3)$. Moreover, $\lambda_3 = (B_2^2 + B_3^2)/(1 + B_2^2 + B_3^2)$, hence $0 \leq \lambda_3 < 1$. The corresponding eigenvectors are $(-B_3, 0, 1)^{\mathrm{T}}$, $(-B2, 1, 0)^{\mathrm{T}}$, and $(1, 0, 0)^{\mathrm{T}}$. Then the general solution of (93) is

$$x_1(t) = (B_2e_2 + B_3e_3)(\lambda_3^t - 1) + x_1^0\lambda_3^t, \quad x_{2,3}(t) = e_{2,3}, \tag{95}$$

which for $t \to \infty$ converges to (24).

(ii) If $e = (0, 0, e_3)^{\mathrm{T}}$, then the dynamics of the network is given by (93) with $x^0 = (x_1^0, x_2^0, e_3)^{\mathrm{T}}$ and

$$W_s = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ 0 & 0 & 1 \end{pmatrix}, \tag{96}$$

where w_{ij} are the corresponding elements of the matrix (20).

It can be shown that the eigenvalues of (96) are $\lambda = (1, 1, \lambda_3)$. Moreover, $\lambda_3 = B_3^2/(1 + B_2^2 + B_3^2)$, hence $0 \leq \lambda_3 < 1$. The corresponding eigenvectors are $\chi_1 = (-B_3, 0, 1)^{\mathrm{T}}$, $\chi_2 = (-B_2, 1, 0)^{\mathrm{T}}$, and $\chi_3 = (1/B_2, 1, 0)^{\mathrm{T}}$. Then the general solution of (93) is given by

$$x(t) = e_3\chi_1 + C_2\chi_2 + C_3\chi_3\lambda_3^t, \tag{97}$$

where

$$C_2 = \frac{x_2^0 + B_2(e_3B_3 - x_1^0)}{1 + B_2^2}, \quad C_3 = \frac{B_2(x_1^0 + x_2^0B_2 + e_3B_3)}{1 + B_2^2}. \tag{98}$$

For $t \to \infty$ (97) converges to (25).

Appendix D: Proof of Theorem 4

The proof is similar to the proof of Theorem 3.

(i) If the novel stimulus has the form $e = (0, e_2, e_3)^{\mathrm{T}}$, then the eigenvalues of (94) with damping factors are $(1, 1, d_1/(1 + d_1))$ with the corresponding eigenvectors $(-B_3, 0, 1)^{\mathrm{T}}$, $(-B2, 1, 0)^{\mathrm{T}}$, and $(1, 0, 0)^{\mathrm{T}}$. Thus the general solution of (93) is given by (95) with $\lambda_3 = d_1/(1 + d_1)$, which defines the relaxation rate.

(ii) If $e = (0, 0, e_3)^{\mathrm{T}}$, then the eigenvalues of (96) with damping factors are

$$\lambda_{1,2} = 1, \ \lambda_3 = \frac{d_1d_2 - 1}{(1 + d_1)(1 + d_2)}. \tag{99}$$

For positive damping factors $0 < \lambda_3 < 1$. The corresponding eigenvectors are

$$\chi_1 = (-B_3, 0, 1)^{\mathrm{T}}, \ \chi_2 = (-B_2, 1, 0), \\ \chi_3 = \left(\tfrac{1+d_2}{1+d_1}B_2, 1, 0\right)^{\mathrm{T}}. \tag{100}$$

Then the general solution of (93) is given by

$$x_1(t) = B_3e_3\tfrac{1+d_2}{2+d_1+d_2}\left(\lambda_3^t - 1\right), \\ x_2(t) = \tfrac{B_3e_3}{B_2}\tfrac{1+d_1}{2+d_1+d_2}\left(\lambda_3^t - 1\right), \\ x_3(t) = e_3. \tag{101}$$

For $t \to \infty$ (101) converges to (27). λ_3 defines the relaxation rate. (28) immediately follows from (101).

Appendix E: Proof of Theorem 5

Let $V_j(t)$ denote the jth column of the matrix $W^{\mathrm{T}}(t)$. Then from (10) we have

$$V_j(t + 1) = \left(I - \varepsilon\xi(t - 1)\xi^{\mathrm{T}}(t - 1)\right) V_j(t) \\ + \varepsilon\xi_j(t)\xi(t - 1). \tag{102}$$

Since $a_1, \ldots, a_p$ are nonzero orthogonal vectors, hence for $p < n$ we can select $n - p$ nonzero orthogonal vectors $\alpha_1, \ldots, \alpha_{n-p} \in \mathbb{R}^n$ such that the space spanned by them is orthogonal to the space spanned by $\{a_1, \ldots, a_p\}$. Consequently, we can introduce two orthogonal subspaces

$$X_1 = \left\{k_{p+1}\alpha_1 + \cdots + k_n\alpha_{n-p} : \quad k_{p+1}, \ldots, k_n \in \mathbb{R}\right\}, \\ X_2 = \left\{k_1a_1 + \cdots + k_pa_p : \quad k_1, \ldots, k_p \in \mathbb{R}\right\}. \tag{103}$$

Clearly, $\mathbb{R}^n = X_1 \oplus X_2$. From (32) it follows that

$$M_{pj}^{\mathrm{T}} = \frac{a_{2j}}{\|a_1\|^2}a_1 + \cdots + \frac{a_{pj}}{\|a_{p-1}\|^2}a_{p-1} + \frac{a_{1j}}{\|a_p\|^2}a_p \in X_2.$$

Now we define a linear map L from $\mathbb{R}^n$ into itself

$$L(V_j) = \left(I - \varepsilon\xi(t - 1)\xi^{\mathrm{T}}(t - 1)\right) V_j, \tag{104}$$

and then for $V_j \in X_1$ we have

$$L(V_j) = V_j, \tag{105}$$

since $\xi(t) \in \{a_1, \ldots, a_p\}$ and X_1 is orthogonal to X_2. For $V_j = k_1a_1 + \cdots + k_pa_p \in X_2$, from (104) the first two iterations for $\xi(0) = a_1$ and $\xi(1) = a_2$ are

$$L^1(V_j) = \left(I - \varepsilon a_1a_1^{\mathrm{T}}\right)(k_1a_1 + k_2a_2 + \cdots + k_pa_p) \\ = \left(1 - \varepsilon\|a_1\|^2\right)k_1a_1 + \textstyle\sum_{i=2}^p k_ia_i$$

and

$$L^2(V_j) = L(L^1(V_j)) = \left(I - \varepsilon a_2a_2^{\mathrm{T}}\right)L^1(V_j) \\ = \left(1 - \varepsilon\|a_1\|^2\right)k_1a_1 + \left(1 - \varepsilon\|a_2\|^2\right)k_2a_2 + \textstyle\sum_{i=3}^p k_ia_i.$$

Proceeding further the ptth iteration is given by

$$L^{pt}(V_j) = \sum_{i=1}^p \left(1 - \varepsilon\|a_i\|^2\right)^t k_ia_i \in X_2. \tag{106}$$

If (30) is satisfied, then $\lim_{t\to\infty} L^{pt}(V_j) = 0$.

Now let us consider the following map, equivalent to (102),

$$V_j(t + 1) = L(V_j(t)) + \varepsilon\xi_j(t)\xi(t - 1) \tag{107}$$

with the initial condition $V_j(0)$. We obtain

$$V_j(t+1) = V_j^1(0) + L^{t+1}(V_j^2(0)) + \sum_{k=0}^{t} L^{t-k}\left(\varepsilon\xi_j(k)\xi(k-1)\right),$$

where we have used (105) and the decomposition $V_j(0) = V_j^1(0)+V_j^2(0)$ associated with the direct sum $\mathbb{R}^n = X_1 \oplus X_2$. Thus, if the iteration (107) in X_2 converges to $\hat{V}_j \in X_2$, then for any $V_j(0)$ it converges to $V_j^1(0) + \hat{V}_j$.

Notice that

$$a_i^{\mathrm{T}} M_{pj}^{\mathrm{T}} = a_{i+1,j}, \quad i = 1, \ldots, p \ (i \bmod p).$$

Consequently, $M_{pj}^{\mathrm{T}} \in X_2$ is a steady state of the map (107) in X_2 since

$$M_{pj}^{\mathrm{T}} - L(M_{pj}^{\mathrm{T}}) = M_{pj}^{\mathrm{T}} - (I - \varepsilon\xi(t-1)\xi^{\mathrm{T}}(t-1))M_{pj}^{\mathrm{T}} = \varepsilon\xi(t-1)\xi^{\mathrm{T}}(t-1)M_{pj}^{\mathrm{T}} = \varepsilon\xi_j(t)\xi(t-1).$$

Now substituting $V_j(t) = M_{pj}^{\mathrm{T}} + \bar{V}_j(t)$ into (107) we get

$$\bar{V}_j(t+1) = L(\bar{V}_j(t)), \quad \bar{V}_j(t) \in X_2. \tag{108}$$

Equations (108) and (106) imply that $\bar{V}_j(t) \to 0$, and hence the steady state M_{pj}^{T} of the map (107) restricted to X_2 is globally asymptotically stable (provided that (30) is satisfied). Thus in $\mathbb{R}^n$ $V_j(t) \to V_j^1(0) + M_{pj}^{\mathrm{T}}$, as $t \to \infty$ and hence

$$\lim_{t\to\infty} V(t) = V^1(0) + M_p^{\mathrm{T}}.$$

Therefore, we have

$$\lim_{t\to\infty} W(t) = \widetilde{W} + M_p,$$

where $\widetilde{W} = V^1(0)^{\mathrm{T}}$.

Obviously, for zero initial conditions $W(0) = 0$, $V^1(0) = 0$, thus $W_\infty = M_p$. Besides, if $p = n$, then $X_2 = \mathbb{R}^n$, and for any initial conditions $\widetilde{W} = 0$, which completes the proof.

Appendix F: Proof of Theorem 6

For a periodic stimulus (29) composed of linearly independent vectors $a_1, \ldots, a_p$, $(p \le n)$, we can select auxiliary nonzero linearly independent vectors such that the space spanned by $\{\alpha_1, \ldots, \alpha_{n-p}\}$ is orthogonal to the space spanned by $\{a_1, \ldots, a_p\}$. Thus we can also introduce two orthogonal subspaces X_1 and X_2, which are formally the same as in (103). Then we note that the matrix W_∞ defined by (34) is the steady state of the map (10) and $W_\infty \in X_2$. Thus, similar arguments as those given in the proof of Theorem 5 complete the proof of Theorem 6.

Appendix G: Proof of Theorem 7

Setting

$$X_1 = \{k_1 a_1 : \ k_1 \in \mathbb{R}\}$$
$$X_2 = \{k_2 a_2 + \cdots + k_{n+1} a_{n+1} : \ k_2, \ldots, k_{n+1} \in \mathbb{R}\}$$

we have $\mathbb{R}^n = X_1 \oplus X_2$. Then under the assumption (37), the matrix W_∞ defined by (36) is the steady state of the learning process (10) and $W_\infty \in X_2$. The rest of the proof is very similar to the proof of Theorem 5.

Appendix H: Proof of Theorem 8

The proof is similar to the proof of Theorem 3. The dynamics of a *Su* network can be described by the map (93) with the appropriately selected coupling matrix W_s and initial conditions $x(0) = x^0$.

(1) If the novel stimulus has the form $e = (0, e_2, e_3)^{\mathrm{T}}$, then the dynamics of the network is given by the map (93) with $x^0 = (x_1^0, e_2, e_3)^{\mathrm{T}}$ and (94), where the matrix elements w_{1j} are given by the corresponding elements of: (a) the matrix (38) for the training by one vector, or (b) the matrix (40) for the training by two vectors. The eigenvalues of (94) are $(1, 1, w_{11})$ with the corresponding eigenvectors $(w_{13}/(1 - w_{11}), 0, 1)$, $(w_{12}/(1 - w_{11}), 1, 0)$, and $(1, 0, 0)$. Then the general solution of (93) is

$$x_1(t) = \frac{w_{12}e_2 + w_{13}e_3}{1 - w_{11}}\left(1 - w_{11}^t\right) + x_1^0 w_{11}^t, \quad x_{2,3}(t) = e_{2,3}. \tag{109}$$

Thus, if

$$|w_{11}| < 1 \tag{110}$$

then (109) converges and diverges otherwise.

(1a) For the training by one vector a

$$w_{11} = \frac{a_1^2}{\|a\|^2}, \quad w_{12} = \frac{a_1 a_2}{\|a\|^2}, \quad w_{13} = \frac{a_1 a_3}{\|a\|^2}. \tag{111}$$

The convergence condition (110) is always satisfied and the network relaxes to (52).

(1b) For the training by two linearly independent vectors a and b from (40) we obtain the matrix elements w_{11}, w_{12}, and w_{13}. Then using (110) and (109) we derive the stability condition (53) and the fixed point (54), respectively.

The fixed point (54) satisfies the basic equation if

$$\left(\frac{\Delta_{12}}{\Gamma - \Delta_{11}} + B_2\right) e_2 + \left(\frac{\Delta_{13}}{\Gamma - \Delta_{11}} + B_3\right) e_3 = 0. \tag{112}$$

Without loss of generality in (112) we used $B_1 = 1$. For an arbitrary novel activation the values in parentheses of (112) should be together equal to zero. This leads to

$$\begin{aligned} (a_1 - b_1)(a_3 + b_3)\Gamma &= 0 \\ (a_1 - b_1)(a_2 + b_2)\Gamma &= 0 \end{aligned} \tag{113}$$

For linearly independent a and b (113) is fulfilled if and only if $a_1 = b_1$.

(2) If the novel stimulus has the form $e = (0, 0, e_3)^{\mathrm{T}}$, then the dynamics of the network is given by the map (93) with $x^0 = (x_1^0, x_2^0, e_3)^{\mathrm{T}}$ and (96), where the matrix elements w_{ij} are given by the corresponding elements of: (a) the matrix (38) for the training by one vector, or (b) the matrix (40) for the training by two vectors.

(2a) For the training by one vector $w_{ij} = a_i a_j / \|a\|^2$. Then the matrix (96) has three eigenvalues $\lambda_1 = 1$, $\lambda_2 = 0$, and $\lambda_3 = (a_1^2 + a_2^2)/\|a\|^2$ with the corresponding eigenvectors a/a_3, $(-a_2/a_1, 1, 0)^{\mathrm{T}}$, and $(a_1/a_2, 1, 0)^{\mathrm{T}}$, respectively. Using the initial condition we obtain the general solution

$$x(t) = \frac{e_3}{a_3} a + \left(\frac{x_1^0 a_1 + x_2^0 a_2}{a_1^2 + a_2^2} - \frac{e_3}{a_3} \right) \times (a_1, a_2, 0)^{\mathrm{T}} \left(\frac{a_1^2 + a_2^2}{\|a\|^2} \right)^t \tag{114}$$

and (55) immediately follows.

(2b) For the training by two vectors one eigenvalue of (96) is $\lambda_1 = 1$ with the corresponding eigenvector $\chi_1 = (a + b)/(a_3 + b_3)$. Let us denote the other two eigenvalues with the corresponding eigenvectors by $\lambda_{2,3}$ and $\chi_2 = (\chi_{21}, 1, 0)^{\mathrm{T}}$, $\chi_3 = (\chi_{31}, 0, 1)^{\mathrm{T}}$, respectively. Then the general solution is given by

$$x(t) = \frac{e_3}{a_3 + b_3}(a + b) + C_2 \chi_2 \lambda_2^t + C_3 \chi_3 \lambda_3^t, \tag{115}$$

where $C_{2,3}$ are constants determined by the initial condition $x(0)$. If $|\lambda_{2,3}| < 1$ then (115) converges to (57), otherwise it diverges. Simple but tedious calculation shows that the condition $|\lambda_{2,3}| < 1$ is reduced to (56), which completes the proof.

Appendix I: Proof of Theorem 9

(i) Similar to the proof of Theorem 8 the dynamics of a *Su* network with damping can be described by the map (93) with the initial condition $x(0) = (0, e_2, e_3)^{\mathrm{T}}$ and

$$W_s = \begin{pmatrix} \frac{d_1}{1+d_1} & \frac{a_2 h_1}{1+d_1} & \frac{a_3 h_1}{1+d_1} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{116}$$

A simple computation shows

$$x_1(t) = (e_2 a_2 + e_3 a_3) h_1 \left[1 - \left(\frac{d_1}{1 + d_1} \right)^t \right], \quad x_{2,3}(t) = e_{2,3}. \tag{117}$$

From (117) the relaxation rate and (61) follow.

(ii) The dynamics of a *Su* network with damping can be described by the map (93) with the initial condition $x(0) = (0, 0, e_3)^{\mathrm{T}}$ and

$$W_s = \begin{pmatrix} \frac{d_1}{1+d_1} & \frac{a_2 h_1}{1+d_1} & \frac{a_3 h_1}{1+d_1} \\ \frac{a_1 h_2}{1+d_2} & \frac{d_2}{1+d_2} & \frac{a_3 h_2}{1+d_2} \\ 0 & 0 & 1 \end{pmatrix}. \tag{118}$$

One eigenvalue of (118) is $\lambda_3 = 1$ while the other two $\lambda_{1,2}$ are roots of the quadratic equation $h(\lambda) = 0$ with

$$h(\lambda) = \lambda^2 - \left(\frac{d_1}{1 + d_1} + \frac{d_2}{1 + d_2} \right) \lambda + \frac{d_1 d_2 - a_1 h_1 a_2 h_2}{(1 + d_1)(1 + d_2)}. \tag{119}$$

First we note that $\lambda_{1,2}$ are real. Then the fixed point of (93) will be stable if $|\lambda_{1,2}| < 1$. The latter is valid if the first two damping constants $d_{1,2}$ satisfy the condition $h(\pm 1) > 0$, which (for $d_{1,2} > 0$) is reduced to $a_1 h_1 a_2 h_2 < 1$, which in turn is always true for $a_i \neq 0$.

We denote by $\chi = (\chi_1, \chi_2, 0)^{\mathrm{T}}$ and $\eta = (\eta_1, \eta_2, 0)^{\mathrm{T}}$ the eigenvectors associated with eigenvalues λ_1 and λ_2, respectively. Moreover, the vector $b_3/a_3 a$ is the eigenvector associated with $\lambda_3 = 1$. Consequently, the general solution of the map (93) is

$$x(t) = C_1 \chi \lambda_1^t + C_2 \eta \lambda_2^t + \frac{e_3}{a_3} a, \tag{120}$$

where the constants $C_{1,2}$ can be determined from the initial condition $x(0) = (0, 0, e_3)^{\mathrm{T}}$. Solution (120) converges to

$$\lim_{t \to \infty} x(t) = \frac{e_3}{a_3} a, \tag{121}$$

which is independent of the damping factors, although they affect the relaxation velocity. To show this we assume that $d_1 = d_2 = d$. Then we have

$$\lambda_{1,2} = \frac{d \pm \sqrt{a_1 h_1 a_2 h_2}}{1 + d}, \quad \frac{\partial \lambda_{1,2}}{\partial d} = \frac{1 \mp \sqrt{a_1 h_1 a_2 h_2}}{(1 + d)^2} > 0, \tag{122}$$

which, together with (120), implies that the for large d, the network slowly approaches the stable solution.

Appendix J: Proof of Theorem 10

(i) Let us first to consider the case of positive vectors $a, b > 0$. At $t = 0$ from (1) and (38) we have

$$s(0) = \frac{a a^{\mathrm{T}}}{\|a\|^2} e = \lambda_1 a \geq 0,$$

where $\lambda_1 = \langle a, e \rangle / \|a\|^2$ is a constant. Without loss of generality we assume that

$$\lambda_1 a_j > e_j \ (j = 1, 2, \ldots, n-1), \ \lambda_1 a_n \leq e_n. \tag{123}$$

This yields

$$s_j(0) > e_j \ (j = 1, 2, \ldots, n-1), \ s_n(0) \leq e_n \tag{124}$$

Inequality (124) together with (3) implies

$$x(1) = (\lambda_1 a_1, \ldots, \lambda_1 a_{n-1}, b_n)^{\mathrm{T}}. \tag{125}$$

For the next iteration the recurrent input is given by

$$s(1) = \frac{aa^{\mathrm{T}}}{\|a\|^2} x(1) = \lambda_2 a \geq 0, \tag{126}$$

where using (123)

$$\begin{aligned} \lambda_2 &= \tfrac{1}{\|a\|^2}\left(a_n e_n + \lambda_1 \sum_{k=1}^{n-1} a_k^2\right) \\ &= \lambda_1 + \tfrac{e_n - \lambda_1 a_n}{\|a\|^2} a_n \geq \lambda_1 \end{aligned} \tag{127}$$

From the inequality (127) and (125), (126) we have

$$s_j(1) \geq x_j(1) > e_j \ (j = 1, 2, \ldots, n-1).$$

Comparing the last element of the output and the recurrent vectors we obtain

$$x_n(1) - s_n(1) = (e_n - \lambda_1 a_n)\left(1 - \frac{a_n^2}{\|a\|^2}\right) \geq 0,$$

which, together with (125), lead to

$$s_n(1) \leq e_n.$$

Thus we get the second iteration of the map (3)

$$x(2) = (\lambda_2 a_1, \ldots, \lambda_2 a_{n-1}, e_n)^{\mathrm{T}}.$$

Repeating the above described procedure we obtain

$$x(t) = (\lambda_t a_1, \ldots, \lambda_t a_{n-1}, e_n)^{\mathrm{T}}, \tag{128}$$

where λ_t is

$$\lambda_{t+1} = \left(1 - \frac{a_n^2}{\|a\|^2}\right)\lambda_t + \frac{a_n e_n}{\|a\|^2}, \ \text{for } t \geq 1, \tag{129}$$

which has a globally stable fixed point $\lambda^* = e_n/a_n$. Thus $x(t)$ converges to

$$x = \frac{e_n}{a_n} a.$$

Using (129) we obtain

$$\lambda_t = \left(\frac{\langle a, e\rangle}{\|a\|^2} - \frac{e_n}{a_n}\right)\left(1 - \frac{a_n^2}{\|a\|^2}\right)^{t-1} + \frac{e_n}{a_n}. \tag{130}$$

Combining (128) and (130) we derive the relaxation trajectory in explicit form.

(ii) For the case $a, e < 0$, the proof is obtained by substituting $x \to -x$, $a \to -a$, and $b \to -b$.

Appendix K: Proof of Theorem 11

Let $W_j(t)$ denote the jth column vector of the matrix $W^{\mathrm{T}}(t)$. Then from (71) we have

$$W_j(t+1) = \left(I - \varepsilon a g^{\mathrm{T}}(a)\right) W_j(t) + \varepsilon a_j a. \tag{131}$$

The characteristic matrix of (131) is give by

$$A(\lambda) = (\lambda - 1)\, I + \varepsilon a g^{\mathrm{T}}(a).$$

A straightforward computation shows that

$$\det A(\lambda) = (\lambda - 1)^{n-1}\left(\lambda - 1 + \varepsilon g^{\mathrm{T}}(a) a\right).$$

It gives one simple and $(n-1)$ multiplicative roots

$$\lambda_{1,2,\ldots,n-1} = 1, \ \lambda_n = 1 - \varepsilon g^{\mathrm{T}}(a) a.$$

Moreover, one can verify that

$$\xi_1 = \begin{pmatrix} g(a_2) \\ -g(a_1) \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ldots, \xi_{n-1} = \begin{pmatrix} g(a_n) \\ 0 \\ 0 \\ \vdots \\ -g(a_1) \end{pmatrix} \tag{132}$$

are $(n-1)$ linearly independent eigenvectors of A corresponding to the multiplicative eigenvalues and the vector a is the eigenvector of A corresponding to the simple eigenvalue λ_n. Thus the general solution of the map (131) is given by

$$W_j(t) = \sum_{i=1}^{n-1} k_i \xi_i + k_n \left(1 - \varepsilon g^{\mathrm{T}}(a) a\right)^t a + \widetilde{V}, \tag{133}$$

where the constants $k_1, \ldots, k_n \in \mathbb{R}$ can be determined using initial activation $W_j(0)$, and $\widetilde{V}$ is a special solution of the map satisfying

$$g^{\mathrm{T}}(a)\widetilde{V} = a_j.$$

For instance, we can choose

$$\widetilde{V} = \frac{a_j}{m}(h(a_1), h(a_2), \ldots, h(a_n))^{\mathrm{T}}, \tag{134}$$

where m is the sum of nonzero values in $\{g(a_1), \ldots, g(a_n)\}$ and

$$h(a_i) = \begin{cases} 1/g(a_i), & \text{if } g(a_i) \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{135}$$

If $W_j(0) = 0$, then using (133)–(135), we obtain

$$\begin{aligned} k_i &= \tfrac{a_j}{m} h(a_1) h(a_{i+1}) - \tfrac{a_j a_{i+1}}{g^{\mathrm{T}}(a) a} h(a_1) \ (i = 1, \ldots, n-1), \\ k_n &= -\tfrac{a_j}{g^{\mathrm{T}}(a) a}. \end{aligned} \tag{136}$$

Combing (132)–(136) leads to

$$\lim_{t \to \infty} W_j(t) = W_j = \sum_{i=1}^{n-1} k_i \xi_i + \widetilde{W} = \frac{a_j}{g^{\mathrm{T}}(a) a} a \tag{137}$$

since $|1-\varepsilon g^{\mathrm{T}}(a)a|<1$ holds if and only if the inequality (72) is satisfied. From (137) it follows that

$$W_\infty^{\mathrm{T}}=(W_1,W_2,\ldots,W_n)=\frac{aa^{\mathrm{T}}}{g^{\mathrm{T}}(a)a}. \tag{138}$$

Since W_∞^{T} is a symmetric matrix (73) follows from (138).

Appendix L: Proof of Theorem 12

Let $W_i(t)$ be the ith row vector of the matrix $W(t)$. Then (74) can be rewritten in the form

$$W_i(t+1)=W_i(t)-\varepsilon g(W_i(t)a)a^{\mathrm{T}}+\varepsilon a_i a^{\mathrm{T}}. \tag{139}$$

The set of fixed points of the map (139) is given by

$$S=\{W_i|W_i^{\mathrm{T}}=k_1\xi_1+\ldots+k_{n-1}\xi_{n-1}+\tfrac{g^{-1}(a_i)}{m}(h(a_1),\ldots,h(a_n))^{\mathrm{T}}\},$$

where $k_1,\ldots,k_{n-1}\in\mathbb{R}$, m is the sum of nonzero numbers a_i in $\{a_1,\ldots,a_n\}$,

$$h(a_i)=\begin{cases}1/a_i, & \text{if } a_i\neq 0\\ 0, & \text{otherwise}\end{cases}$$

and

$$\xi_1=\begin{pmatrix}a_2\\-a_1\\0\\\vdots\\0\end{pmatrix},\ldots,\xi_{n-1}=\begin{pmatrix}a_n\\0\\0\\\vdots\\-a_1\end{pmatrix}.$$

Expanding $g(W_i(t)a)$ in Taylor series at $W_ia=g^{-1}(a_i)$ up to the first order, we obtain

$$g(W_i(t)a)=g(W_ia)+g'(g^{-1}(a_i))(W_i(t)-W_i)a+O((W_i(t)-W_i)a).$$

Substituting this into (139) and denoting $V_i(t)=W_i^{\mathrm{T}}(t)-W_i^{\mathrm{T}}$, we get

$$V_i(t+1)=V_i(t)-\varepsilon g'(g^{-1}(a_i))aa^{\mathrm{T}}V_i(t)+O(a^{\mathrm{T}}V_i(t))a. \tag{140}$$

Then the map (140) has one simple and $(n-1)$ multiplicative roots

$$\lambda_{1,2,\ldots,n-1}=1,\ \lambda_n=1-\varepsilon g'(g^{-1}(a_i))a^{\mathrm{T}}a$$

with the corresponding eigenvectors $\xi_1,\ldots,\xi_{n-1}$ and a.

Generally speaking the linear approximation is not sufficient to determine stability if there exists an eigenvalue equal to 1. However, we can handle the problem as follows. In order to determine the local stability of the fixed points of (140) we first put the linear part of (140) in the block-diagonal form. The matrix associated with the linear transformation has columns consisting of the eigenvectors of the linearized map. From the above, it is given by

$$C=(\xi_1,\ldots,\xi_{n-1},a).$$

Thus, setting $V_i(t)=CY(t)$ with $Y(t)=(y_1(t),\ldots,y_n(t))^{\mathrm{T}}$ it follows that

$$a^{\mathrm{T}}C=(0,0,\ldots,\|a\|),\ C^{-1}a=(0,0,\ldots,0,1)^{\mathrm{T}}$$

and then the map (140) becomes

$$Y(t+1)=AY(t)+O(\|a\|^2y_n(t))(0,0,\ldots,0,1)^{\mathrm{T}} \tag{141}$$

with $A=\mathrm{diag}\{1,\ldots,1,1-\varepsilon g'(g^{-1}(a_i))a^{\mathrm{T}}a\}$, which is equivalent to

$$y_i(t+1)=y_i(t)\ (i=1,2,\ldots,n-1),$$
$$y_n(t+1)=(1-\varepsilon g'(g^{-1}(a_i))a^{\mathrm{T}}a)y_n(t)+O(y_n(t)).$$

If $|1-\varepsilon g'(g^{-1}(a_i))a^{\mathrm{T}}a|<1$, then

$$Y(t)\to(y_1(0),y_2(0),\ldots,y_{n-1}(0),0)^{\mathrm{T}}\text{ as }t\to\infty.$$

Applying the initial condition $W_i(0)=0$ and following the same procedure as described in the proof of Theorem 11, we obtain

$$\lim_{t\to\infty}W_i(t)=\frac{ag^{-1}(a_i)}{a^{\mathrm{T}}a}$$

and (76) immediately follows.

References

Beer RD (2006) Parameter space structure of continuous-time recurrent neural networks. Neural Comput 18:3009–3051

Cruse H, Hübner D (2008) Selforganizing memory: active learning of landmarks used for navigation. Biol Cybern (submitted)

Cruse H, Sievers K (2008) A general network structure for learning Pavlovian paradigms (in preparation)

Elman JL (1990) Finding structure in time. Cogn Sci 14:179–211

Feynman R (2001) In: Hawking SW (ed) The universe in a nutshell. Bantam Press, New York

Fuster JM (1995) Memory in the cerebral cortex: an empirical approach to neural networks in the human and nonhuman primate. MIT Press, Cambridge

Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci 79:2554–2558

Hopfield JJ (1984) Neurons with graded response have collective computational properties like those of two state neurons. Proc Natl Acad Sci 81:3088–3092

Jaeger H, Haas H (2004) Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science 2:78–80

Kühn S, Beyn WJ, Cruse H (2007) Modelling memory functions with recurrent neural networks consisting of input compensation units: I. Static situations. Biol Cybern 96:455–470

Kühn S, Cruse H (2007) Modelling memory functions with recurrent neural networks consisting of input compensation units: II. Dynamic situations. Biol Cybern 96:471–486

Kindermann T, Cruse H (2002) MMC—a new numerical approach to the kinematics of complex manipulators. Mech Mach Theory 37:375–394

Palm G, Sommer FT (1996) Associative data storage and retrieval in neural networks. In: Domany E, van Hemmen JL, Schulten K (eds) Models of neural networks III. Association, generalization, and representation. Springer, New York, pp 79–118

Pasemann F (2002) Complex dynamics and the structure of small neural networks. Netw: Comput Neural Syst 13:195–216

Steinkühler U, Cruse H (1998) A holistic model for an internal representation to control the movement of a manipulator with redundant degrees of freedom. Biol Cybern 79:457–466

Strang G (2003) Introduction to linear algebra. Wellesley–Cambridge Press, Cambridge

Tani J (2003) Learning to generate articulated behavior through the bottom-up and the top-down interaction processes. Neural Netw 16:11–23

Wessnitzer J, Webb B (2006) Multimodal sensory integration in insects—towards insect brain control architectures. Bioinspir Biomim 1:63–75