Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2013, Article ID 480560, 19 pages
http://dx.doi.org/10.1155/2013/480560

Research Article

Discrete Pseudo-SINR-Balancing Nonlinear Recurrent System

Zekeriya Uykan[1,2]

[1] *Control and Automation Engineering Department, Doğuş University, Acibadem, Kadikoy, 34722 Istanbul, Turkey*
[2] *Aalto University School of Electrical Engineering, Department of Communications and Networking (COMNET), PL 13000 Aalto, 00076 Espoo, Finland*

Correspondence should be addressed to Zekeriya Uykan; zuykan@dogus.edu.tr

Received 24 October 2012; Revised 7 February 2013; Accepted 5 March 2013

Academic Editor: Kwok-Wo Wong

Being inspired by the Hopfield neural networks (Hopfield (1982) and Hopfield and Tank (1985)) and the nonlinear sigmoid power control algorithm for cellular radio systems in Uykan and Koivo (2004), in this paper, we present a novel discrete recurrent nonlinear system and extend the results in Uykan (2009), which are for autonomous linear systems, to nonlinear case. The proposed system can be viewed as a discrete-time realization of a recently proposed continuous-time network in Uykan (2013). In this paper, we focus on discrete-time analysis and provide various novel key results concerning the discrete-time dynamics of the proposed system, some of which are as follows: (i) the proposed system is shown to be stable in synchronous and asynchronous work mode in discrete time; (ii) a novel concept called Pseudo-SINR (pseudo-signal-to-interference-noise ratio) is introduced for discrete-time nonlinear systems; (iii) it is shown that when the system states approach an equilibrium point, the instantaneous Pseudo-SINRs are balanced; that is, they are equal to a target value. The simulation results confirm the novel results presented and show the effectiveness of the proposed discrete-time network as applied to various associative memory systems and clustering problems.

1. Introduction

Artificial neural networks have been an important research area since 1970s. Since then, various biologically inspired neural network models have been developed. Hopfield Neural Networks [1, 2] have been one of the most widely used neural networks since the early 1980s whose applications vary from combinatorial optimization (e.g., [3, 4]) to image restoration (e.g., [5]) and from various control engineering optimization problems in robotics (e.g., [6]) to associative memory systems (e.g., [7, 8]). For a tutorial and further references about Hopfield neural networks, see, for example, [9, 10].

In [11], we introduce a novel pseudo-signal-to-interference-noise ratio concept for discrete-time autonomous linear systems. Our main motivation in this paper is to investigate a nonlinear extension of [11]. Furthermore, the proposed system can be viewed as a discrete-time realization of a very recently proposed continuous-time network called double-sigmoid continuous-time Hopfield neural network in a brief letter [12]. And our investigations in this paper yield various interesting key novel results in discrete time, some of which are as follows: (i) a novel concept called Pseudo-SINR (pseudo-signal-to-interference-noise ratio) is introduced for discrete-time nonlinear systems; (ii) it is shown that when the network approaches to one of its equilibrium points, the instantaneous Pseudo-SINRs are balanced; that is, they are equal to a target value; (iii) the proposed network outperforms its Hopfield neural network counterpart as applied to various associative memory systems and clustering applications. The disadvantage of the proposed network is that it increases the computational burden.

The paper is organized as follows. The proposed recurrent network and its stability features are analyzed in Section 2. Simulation results are presented in Section 3, followed by conclusions in Section 4.

2. Discrete Pseudo-SINR-Balancing Recurrent Neural Networks

Being inspired by the nonlinear sigmoid power control algorithm for cellular radio systems in [13] and the Hopfield

neural networks [2], we propose the following discrete nonlinear recurrent network:

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \alpha(k)\,\mathbf{f}_1\left(-\mathbf{A}\mathbf{x}(k) + \mathbf{W} f_2(\mathbf{x}(k)) + \mathbf{b}\right), \tag{1}$$

where k represents the iteration step, $\mathbf{A}$, $\mathbf{W}$, and $\mathbf{b}$ are defined as in (2), and $\mathbf{f}_m(\cdot)$, $m = 1, 2$, represents a vectoral mapping from $\Re^N$ to $\Re^N$. For an N-dimensional vector $\mathbf{e} = [e_1 e_2 \cdots e_N]^T$, $\mathbf{f}_m(\mathbf{e}) = [f_m(e_1) f_m(e_2) \cdots f_m(e_N)]^T$ where $f_m(\cdot)$ is chosen as the sigmoid function; that is, for a real number e_i, the output is $f_m(e_i) = \kappa_m(1-(2/(1+\exp(-\sigma_m e_i))))$, where $\kappa_m > 0$, $\sigma_m > 0$. We will call the network in (1) as discrete sigmoid-pseudo-SINR-balancing recurrent neural network (D-SP-SNN). The name comes from the fact that the proposed network balances an entity called Pseudo-SINR, as will be seen in the following. In this paper, we choose sigmoid function because it's used both in Hopfield neural network and the power control algorithm in [13]. Furthermore, the proposed D-SP-SNN can be viewed as a discrete-time implementation of a very recently proposed continuous-time network called double-sigmoid continuous-time Hopfield neural network in the brief [12]. In this paper, we focus on discrete-time analysis and provide various novel key results concerning the discrete-time dynamics of the proposed system:

$$\mathbf{A} = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & & \ddots & 0 \\ 0 & 0 & \cdots & a_{NN} \end{bmatrix},$$

$$\mathbf{W} = \begin{bmatrix} 0 & w_{12} & \cdots & w_{1N} \\ w_{21} & 0 & \cdots & w_{2N} \\ \vdots & & \ddots & \vdots \\ w_{N1} & w_{N2} & \cdots & 0 \end{bmatrix}, \tag{2}$$

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}.$$

In (2), $\mathbf{A}$ shows the self-state-feedback matrix, $\mathbf{W}$ with zero diagonal shows the interneurons connection weight matrix, and $\mathbf{b}$ is a threshold vector.

The proposed network includes both the sigmoid power control in [13] and the traditional Hopfield neural network (HNN) as its special cases by choosing the $f_1(\cdot)$ and $f_2(\cdot)$ appropriately. The Euler approximation of the continuous-time HNN is given as

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \alpha(k)\left(-\mathbf{A}\mathbf{x}(k) + \mathbf{W} f_2(\mathbf{x}(k)) + \mathbf{b}\right). \tag{3}$$

Let us call the network in (3) HNN-Euler, which is a special case of the proposed D-SP-SNN in (1). From (1),

$$x_j(k+1) = x_j(k) + \alpha(k) \times f_1\left(-a_{jj}x_j(k) + b_j + \sum_{i=1, i\neq j}^{N} w_{ij} f_2(x_i(k))\right), \quad j = 1, \ldots, N, \tag{4}$$

where $\alpha(k)$ is the step size at time k. Let's define the error signal $e_i(k)$ as

$$e_i(k) = -a_{ii}x_i + I_i(k), \quad \text{where } I_i(k) = b_i + \sum_{j=1, j\neq i}^{N} w_{ij} f_2\left(x_j(k)\right), \quad i = 1, \ldots, N. \tag{5}$$

Then, the performance index is defined as l_1-norm of the error vector in (5) as follows:

$$V(k) = \|\mathbf{e}(k)\|_1 = \sum_{i}^{N} |e_i(k)| \tag{6}$$

$$= \sum_{i}^{N} \left|-a_{ii}x_i + I_i\right|, \quad \text{where } I_i = b_i + \sum_{j=1, j\neq i}^{N} w_{ij} f_2\left(x_j\right). \tag{7}$$

In what follows, we examine the evolution of the energy function in (6) in synchronous and asynchronous work modes. Asynchronous mode means that at every iteration step, at most only one state is updated, whereas synchronous mode refers to the case that all the states are updated at every iteration step according to (4).

Proposition 1. *In asynchronous mode of the proposed network D-SP-SNN in* (4) *with a symmetric matrix* $\mathbf{W}$*, for a nonzero error vector, the* l_1*-norm of the error vector in* (6) *decreases at every step; that is, the error vector goes to zero for any* $\alpha(k)$ *such that*

$$\left|e_j(k)\right| > \left|a_{jj}\alpha(k)\, f_1\left(e_j(k)\right)\right| \tag{8}$$

if

$$\left|a_{jj}\right| \geq k_2 \sum_{i=1, (i\neq j)}^{N} \left|w_{ij}\right|, \tag{9}$$

where $k_2 = 0.5\sigma_2$ *is the global Lipschitz constant of* $f_2(\cdot)$ *as shown the in Appendix A.*

Proof. In asynchronous mode, only one state is updated at an iteration time. Let j show the index of the state which is updated at time k whose error signal is different than zero;

that is, $e_j = -a_{jj}x_j + I_j \neq 0$, where $I_j = b_j + \sum_{i=1, i\neq j}^{N} w_{ji} f_2(x_i)$, as defined in (5). Writing (5) in vector form for steps k and $k+1$ results in

$$\mathbf{e}(k+1) - \mathbf{e}(k) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ -a_{jj}\left(x_j(k+1) - x_j(k)\right) \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} w_{1j} \\ w_{2j} \\ \vdots \\ 0 \\ \vdots \\ w_{Nj} \end{bmatrix} \left(f_2\left(x_j(k+1)\right) - f_2\left(x_j(k)\right)\right). \tag{10}$$

Using the error signal definition of (5) in (4) gives

$$x_j(k+1) - x_j(k) = \alpha(k) f_1\left(e_j(k)\right). \tag{11}$$

So, the error signal for state j is obtained using (10) and (11) as follows:

$$e_j(k+1) - e_j(k) = -a_{jj}\left(x_j(k+1) - x_j(k)\right) \tag{12}$$

$$= -a_{jj}\alpha(k) f_1\left(e_j(k)\right). \tag{13}$$

From (12) and (13), if $\alpha(k)$ is chosen to satisfy $|e_j(k)| > |a_{jj}\alpha(k) f_1(e_j(k))|$, then

$$\left|e_j(k+1)\right| < \left|e_j(k)\right|, \quad \text{for } \left|e_i(k)\right| \neq 0, \tag{14}$$

where $f_1(\cdot)$ is a sigmoid function, which is lower and upper bounded. Since the sigmoid function $f_1(\cdot)$ has the same sign as its argument and $f_1(e_j) = 0$ if and only if $e_j = 0$, then it is seen that $\alpha(k)$ can easily be chosen small enough to satisfy $|e_j(k)| > \alpha(k)a_{jj}|f_1(e_j(k))|$ according to the parameter a_{jj} and the slope of sigmoid function $f_1(\cdot)$.

Above, we examined only the state j and its error signal $e_j(k)$. In what follows, we examine the evolution of the norm of the complete error vector $\mathbf{e}(k+1)$ in (10). From the point of view of the l_1 norm of the $\mathbf{e}(k+1)$, the worst case is that when $|e_j(k)|$ decreases, all other elements $|e_i(k)|$, $i \neq j$, increase. So, using (10), (12), and (14), we obtain that if

$$\left|-a_{jj}\left(x_j(k+1) - x_j(k)\right)\right| \geq \left|f_2\left(x_j(k+1)\right) - f_2\left(x_j(k)\right)\right| \times \sum_{i=1,(i\neq j)}^{N} \left|w_{ij}\right|, \tag{15}$$

then

$$\|\mathbf{e}(k+1)\|_1 \begin{cases} < \|\mathbf{e}(k)\|_1 & \text{if } \|\mathbf{e}(k)\|_1 \neq \mathbf{0}, \\ = 0 & \text{if } \|\mathbf{e}(k)\|_1 = \mathbf{0}. \end{cases} \tag{16}$$

The sigmoid function $f_2(\cdot)$ is a Lipschitz continuous function as shown in Appendix A. So,

$$k_2\left|x_j(k+1) - x_j(k)\right| \geq \left|f_2\left(x_j(k+1)\right) - f_2\left(x_j(k)\right)\right|, \tag{17}$$

where $k_2 = 0.5\sigma_2$ is $f_2(\cdot)$'s global Lipschitz constant as shown in Appendix A. From (15) and (17), choosing $|a_{jj}| > k_2 \sum_{i=1,(i\neq j)}^{N} |w_{ij}|$ yields (15), which implies (16). This completes the proof. □

Definition 2 (pseudo-SINR). We define the following system variable, which will be called pseudo-SINR, for the D-SP-SNN in (4):

$$\frac{\overline{\theta}_i}{\theta_i^{\text{tgt}}} = \frac{a_{ii} f(x_i)}{b_i + \sum_{j=1, j\neq i}^{N} w_{ij} f\left(x_j\right)}, \quad i = 1, \ldots, N, \tag{18}$$

where $f(\cdot)$ represents the sigmoid function, that is, $f(e) = \kappa(1 - (2/(1+\exp(-\sigma e))))$, and θ_i^{tgt} is a constant, which we call target θ_i.

Examining the $\overline{\theta}_i$ in (18), we observe that it resembles the traditional signal-to-interference-noise ratio (SINR) definition in cellular radio systems (see, e.g., [14, 15]); therefore we call it Pseudo-SINR.

Definition 3 (prototype vectors). Prototype vectors are defined as those $\mathbf{x}$'s which make $\theta_i = \theta_i^{\text{tgt}}$, $i = 1, \ldots, N$, in (18). So, from (18) and (5), the prototype vectors make the error signal zero; that is, $e_i = 0$, $i = 1, \ldots, N$ given that $x_i \neq 0$ and $I_i \neq 0$.

Proposition 4. *In asynchronous mode, choosing the slope of $f_2(\cdot)$ relatively small as compared to $f_1(\cdot)$ and choosing $a_{jj} > 0$ and $\alpha(k)$ satisfying* (8)*, the D-SP-SNN in* (4) *with a symmetric matrix* $\mathbf{W}$ *is stable and there exists a finite time constant such that the l_1-norm of the error vector in* (6) *approaches to an ϵ-vicinity of the zero as its steady state, where ϵ is a relatively small positive number. If $\overline{\theta}_i = \theta_i^{tgt}$ at the converged point, then it corresponds to a prototype vector as defined above.*

Proof. Since it is asynchronous mode, (10)–(14) hold where $a_{jj} > 0$. So, if $\alpha(k)$ at time k is chosen to satisfy $|e_j(k)| > |a_{jj}\alpha(k) f_1(e_j(k))|$ as in (8), then

$$\left|e_j(k+1)\right| < \left|e_j(k)\right|, \quad \text{for } \left|e_i(k)\right| \neq 0. \tag{19}$$

Note that it is straightforward to choose a sufficiently small $\alpha(k)$ to satisfy (8) according to a_{jj} and the slope σ_1 of sigmoid $f_1(\cdot)$. Using (10), (12), and (19), it is seen for $e_j(k) \neq 0$ that if

$$\begin{aligned} &\left|-a_{jj}\left(x_j(k+1) - x_j(k)\right)\right| \\ &\quad = \left|-a_{jj}\alpha(k) f_1\left(e_j(k)\right)\right| \\ &\quad > \left|f_2\left(x_j(k+1)\right) - f_2\left(x_j(k)\right)\right| \end{aligned} \tag{20}$$

$$\times \sum_{i=1,(i\neq j)}^{N} \left|w_{ij}\right|, \tag{21}$$

then

$$\|\mathbf{e}(k+1)\|_1 < \|\mathbf{e}(k)\|_1. \tag{22}$$

We observe from (12), (20), (21), and (22) the following.

(1) If the $x_i(k)$, $i = 1, \ldots, N$, approach to either of the saturation regimes of its sigmoid function $f_2(\cdot)$, then

$$\left|f_2\left(x_j(k+1)\right) - f_2\left(x_j(k)\right)\right| \sum_{i=1,(i\neq j)}^{N} \left|w_{ij}\right| \approx 0, \quad j = 1, \ldots, N, \tag{23}$$

since $|f_2(x_j(k+1)) - f_2(x_j(k))| \approx 0$, $i = 1, \ldots, N$. That satisfies (20) and (21). Therefore, the norm of the error vector in (6) does not go to infinity and is finite for any $\mathbf{x}$.

(2) $\mathbf{x}(k+1) = \mathbf{x}(k)$ if and only if $\mathbf{e}(k) = \mathbf{0}$; that is,

$$x_j(k+1) = x_j(k) \quad \text{if and only if } f_1\left(e_j(k)\right) = 0, \quad j = 1, \ldots, N. \tag{24}$$

(3) Examining the (11), (12), and (13) taking the observations (1) and (2) into account, we conclude that any of the $x_j(k)$, $j = 1, \ldots, N$, does not go to infinity and is finite for any k. So, the D-SP-SNN in (4) with a symmetric matrix $\mathbf{W}$ is stable for the assumptions in Proposition 4. Because there is a finite number of insaturation states (i.e., the number of all possible insaturation state combinations is finite), which is equal to 2^N, there exists a finite time constant such that the l_1-norm of the error vector in (6) approaches to an ϵ-vicinity of the zero as its steady state, where ϵ is a relatively small positive number.

From (18), if $\overline{\theta}_i = \theta_i^{\text{tgt}}$ at the converged point, then it corresponds to a prototype vector as defined in the previous section, which completes the proof. □

In what follows, we examine the evolution of pseudo-SINR $\overline{\theta}_i(k)$ in (18). From (18), let us define the following error signal at time k:

$$\xi_j(k) = -\theta_j(k) + \theta_j^{\text{tgt}}, \quad j = 1, \ldots, N. \tag{25}$$

Proposition 5. *In asynchronous mode, in the D-SP-SNN in* (4) *with a sufficiently small* $\alpha(k)$ *and with a symmetric matrix* $\mathbf{W}$*, the* $\theta_j(k)$ *is getting closer to* θ_j^{tgt} *at those iteration steps* k *where* $I_j(k) \neq 0$*; that is,* $|\xi_j(k+1)| < |\xi_j(k)|$*, where index* j *shows the state being updated at iteration* k*.*

Proof. Let j show the state which is updated at time k. The pseudo-SINR defined by (18) for nonzero $I_j(k)$ is equal to

$$\overline{\theta}_j(k) = \frac{a_{ii}x_j(k)}{I_j(k)}, \quad \text{where } I_j(k) = b_j + \sum_{i=1, i\neq j}^{N} w_{ji} f\left(x_j(k)\right). \tag{26}$$

Without loss of generality, and for the sake of simplicity, let is take $\theta_i^{\text{tgt}} = 1$. Then, from (26) and (25)

$$\xi_j(k) = -\overline{\theta}_j(k) + 1 = \frac{-a_{ii}x_j(k) + I_j(k)}{I_j(k)}. \tag{27}$$

In asynchronous mode, from (26), $I_m(k) = I_m(k+1)$. Using this observation and (27),

$$\xi_j(k+1) - \xi_j(k) = \frac{-a_{ii}\left(x_j(k+1) - x_j(k)\right)}{I_j(k)}. \tag{28}$$

From (4) and (28),

$$\xi_j(k+1) - \xi_j(k) = \frac{-a_{ii}\alpha(k) f_1\left(e_j(k)\right)}{I_j(k)}. \tag{29}$$

Provided that $I_j(k) \neq 0$, we write, from (5) and (27),

$$e_j(k) = I_j(k)\xi_j(k). \tag{30}$$

Writing (30) in (29) gives

$$\xi_j(k+1) - \xi_j(k) = \frac{-a_{ii}\alpha(k) f_1\left(I_j(k)\xi_j(k)\right)}{I_j(k)}. \tag{31}$$

From (31), since sigmoid function $f_1(\cdot)$ is an odd function, and $a_{ii} > 0$ and $\alpha(k) > 0$,

$$\xi_j(k+1) = \xi_j(k) - \beta \operatorname{sign}\left(\xi_j(k)\right), \quad \text{where } \beta = \left|\frac{-a_{ii}\left(\alpha(k) f_1\left(I_j(k)\xi_j(k)\right)\right)}{I_j(k)}\right|. \tag{32}$$

As seen from (32), for a nonzero $\xi_j(k)$, choosing a sufficiently small $\alpha(k)$ satisfying $|\xi_j(k)| > \beta$ assures that

$$\left|\xi_j(k+1)\right| < \left|\xi_j(k)\right| \quad \text{if } I_m(k) \neq 0 \tag{33}$$

which completes the proof. □

Proposition 6. *The results in Propositions 1 and 4 for asynchronous mode hold also for synchronous mode.*

In synchronous mode, all the states are updated at every step k according to (5). So, from (5)

$$\begin{aligned}\mathbf{e}(k+1) - \mathbf{e}(k) &= \sum_{i=1}^{N} \left(\begin{bmatrix} 0 \\ 0 \\ \vdots \\ -a_{11}\left(x_i(k+1) - x_i(k)\right) \\ \vdots \\ 0 \end{bmatrix} \right. \\ &\quad \left. + \begin{bmatrix} w_{1i} \\ w_{2i} \\ \vdots \\ 0 \\ \vdots \\ w_{Ni} \end{bmatrix} \left(f_2\left(x_i(k+1)\right) - f_2\left(x_i(k)\right)\right) \right).\end{aligned} \tag{34}$$

Using (5) in (34) and writing it elementwise give

$$e_i(k+1) = e_i(k) - a_{ii}\alpha(k) f_1(e_i(k)) + \sum_{j=1,(j\neq i)}^{N} w_{ij}\left(f_2\left(x_j(k+1) - f_2\left(x_j(k)\right)\right)\right), \quad i = 1,\ldots,N. \tag{35}$$

From (34) and (35), we obtain

$$\left|-a_{ii}\left(x_i(k+1) - x_i(k)\right)\right| = \left|-a_{ii}\alpha(k) f_1(e_i(k))\right| > \left|f_2(x_i(k+1)) - f_2(x_i(k))\right| \times \sum_{j=1,(j\neq i)}^{N} \left|w_{ji}\right|, \quad i = 1,\ldots,N, \tag{36}$$

which is equal to (15) in Proposition 1 and (20) in Proposition 4.

It is well known that the performance of Hopfield network may highly depend on the parameter setting of the weight matrix (e.g., [8]). There are various ways for determining the weight matrix of the Hopfield networks: gradient-descent supervised learning (e.g., [16]), solving linear inequalities (e.g., [17, 18] among others), Hebb learning rule [19, 20], and so forth. How to design D-SP-SNN is out of the scope of this paper. The methods used for traditional Hopfield NN can also be used for the proposed networks D-SP-SNN. As far as the simulation results in Section 3 are concerned, we determine the matrices **A**, **W**, and **b** by using a Hebb learning-based algorithm [19] presented in Appendix B.

3. Simulation Results

In the simulation part, we examine the performance of the proposed D-SP-SNN in the area of associative memory systems and clustering problem. In Examples 7 and 8, we present some toy examples one with 8 neurons and one with 16 neurons, respectively, where the desired vectors are orthogonal. Lyapunov function of the HNN at time k is given as

$$L(k) = -\mathbf{x}(k)^T \mathbf{W}\mathbf{x}(k) + \mathbf{x}(k)^T \mathbf{b}. \tag{37}$$

In Examples 7 and 8, we use discrete-time HNN just for comparison reasons, which is given by

$$\mathbf{x}(k+1) = \operatorname{sign}(\mathbf{W}\mathbf{x}(k)), \tag{38}$$

where $\mathbf{W}$ is the weight matrix and $\mathbf{x}(k)$ is the state at time k, and at most one state is updated at a time.

Example 7. In this example of discrete-time networks, there are 8 neurons. The desired prototype vectors are as follows:

$$\mathbf{D} = \begin{bmatrix} 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}. \tag{39}$$

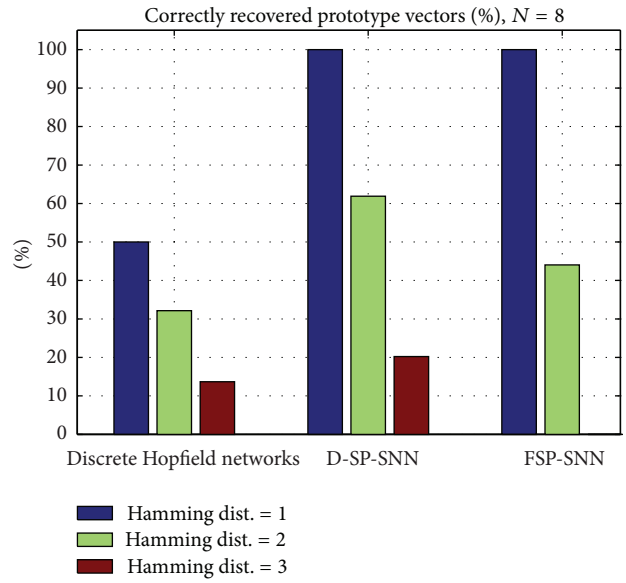

Figure 1: The figure shows the percentage of correctly recovered desired patterns for all possible initial conditions in Example 7 for the proposed D-SP-SNN and FSP-SNN as compared to traditional Hopfield network (8-neuron case).

The weight matrices **A** and **W** and the threshold vector **b** are obtained as follows by using the outer-product-based design (Hebb-learning [19]) presented in Appendix B and the slopes of sigmoid functions $f_1(\cdot)$ and $f_2(\cdot)$ are set to $\sigma_1 = 10$, $\kappa_1 = 10$, and $\sigma_2 = 2$, $\kappa_2 = 1$, respectively, and $\rho = 0$, $\alpha = 0.1$:

$$\mathbf{A} = 3\mathbf{I},$$

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 1 & -1 & 1 & -1 & -1 & -3 \\ 1 & 0 & -1 & 1 & -1 & 1 & -3 & -1 \\ 1 & -1 & 0 & 1 & -1 & -3 & 1 & -1 \\ -1 & 1 & 1 & 0 & -3 & -1 & -1 & 1 \\ 1 & -1 & -1 & -3 & 0 & 1 & 1 & -1 \\ -1 & 1 & -3 & -1 & 1 & 0 & -1 & 1 \\ -1 & -3 & 1 & -1 & 1 & -1 & 0 & 1 \\ -3 & -1 & -1 & 1 & -1 & 1 & 1 & 0 \end{bmatrix}, \tag{40}$$

$$\mathbf{b} = \mathbf{0}.$$

Figure 1 shows the percentages of correctly recovered desired patterns for all possible initial conditions $\mathbf{x}(k) \in (-1,+1)^8$, for the proposed networks D-SP-SNN as compared to traditional discrete Hopfield network. In the proposed network D-SP-SNN, $f_1(\cdot)$ is a sigmoid function. Establishing an analogy to the traditional fixed step 1-bit increase/decrease power control algorithm (e.g. [21, 22]), we replace the sigmoid function by the sign function and call corresponding network as fixed-step pseudo-SINR neural network (FSPSNN). For comparison reason its performance is also shown in Figure 1.

As seen from Figure 1 the performance of the proposed network D-SP-SNN is remarkably better than that of the traditional discrete Hopfield network for all Hamming distance cases. The FSP-SNN also considerably outperforms the

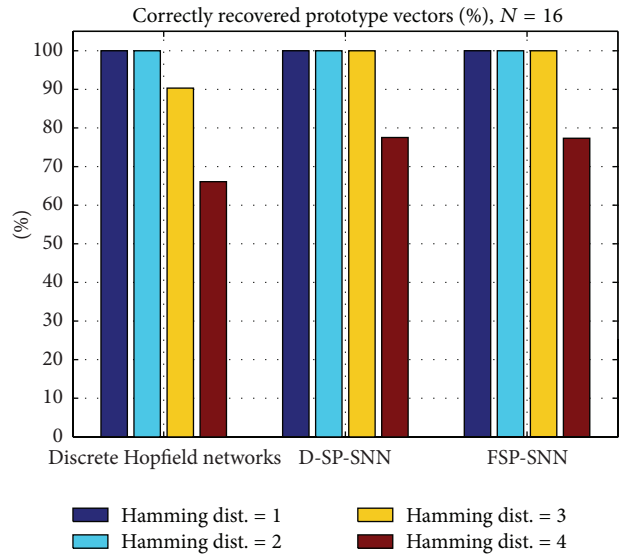

FIGURE 2: The figure shows the percentage of correctly recovered desired patterns for all possible initial conditions in Example 8 for the proposed D-SP-SNN and its 1-bit version FSP-SNN as compared to traditional Hopfield network (16-neuron case).

Hopfield network for 1 and 2 Hamming distance cases while the all the networks perform poorly (less than 20%) at 3-Hamming distance case.

Example 8. The desired prototype vectors are

$$\mathbf{D} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}. \tag{41}$$

The weight matrices $\mathbf{A}$ and $\mathbf{W}$ and threshold vector $\mathbf{b}$ are obtained as follows by using the outer-product-based design (Hebb-learning [19]) in Appendix B:

$$\mathbf{A} = 4\mathbf{I}, \qquad \mathbf{W} = \begin{bmatrix} 0 & 2 & 2 & 0 & 2 & 0 & 0 & -2 & 2 & 0 & 0 & -2 & 0 & -2 & -2 & -4 \\ 2 & 0 & 0 & 2 & 0 & 2 & -2 & 0 & 0 & 2 & -2 & 0 & -2 & 0 & -4 & -2 \\ 2 & 0 & 0 & 2 & 0 & -2 & 2 & 0 & 0 & -2 & 2 & 0 & -2 & -4 & 0 & -2 \\ 0 & 2 & 2 & 0 & -2 & 0 & 0 & 2 & -2 & 0 & 0 & 2 & -4 & -2 & -2 & 0 \\ 2 & 0 & 0 & -2 & 0 & 2 & 2 & 0 & 0 & -2 & -2 & -4 & 2 & 0 & 0 & -2 \\ 0 & 2 & -2 & 0 & 2 & 0 & 0 & 2 & -2 & 0 & -4 & -2 & 0 & 2 & -2 & 0 \\ 0 & -2 & 2 & 0 & 2 & 0 & 0 & 2 & -2 & -4 & 0 & -2 & 0 & -2 & 2 & 0 \\ -2 & 0 & 0 & 2 & 0 & 2 & 2 & 0 & -4 & -2 & -2 & 0 & -2 & 0 & 0 & 2 \\ 2 & 0 & 0 & -2 & 0 & -2 & -2 & -4 & 0 & 2 & 2 & 0 & 2 & 0 & 0 & -2 \\ 0 & 2 & -2 & 0 & -2 & 0 & -4 & -2 & 2 & 0 & 0 & 2 & 0 & 2 & -2 & 0 \\ 0 & -2 & 2 & 0 & -2 & -4 & 0 & -2 & 2 & 0 & 0 & 2 & 0 & -2 & 2 & 0 \\ -2 & 0 & 0 & 2 & -4 & -2 & -2 & 0 & 0 & 2 & 2 & 0 & -2 & 0 & 0 & 2 \\ 0 & -2 & -2 & -4 & 2 & 0 & 0 & -2 & 2 & 0 & 0 & -2 & 0 & 2 & 2 & 0 \\ -2 & 0 & -4 & -2 & 0 & 2 & -2 & 0 & 0 & 2 & -2 & 0 & 2 & 0 & 0 & 2 \\ -2 & -4 & 0 & -2 & 0 & -2 & 2 & 0 & 0 & -2 & 2 & 0 & 2 & 0 & 0 & 2 \\ -4 & -2 & -2 & 0 & -2 & 0 & 0 & 2 & -2 & 0 & 0 & 2 & 0 & 2 & 2 & 0 \end{bmatrix}, \qquad \mathbf{b} = \mathbf{0}. \tag{42}$$

Figure 2 shows the percentage of correctly recovered desired patterns for all possible initial conditions $\mathbf{x}(k) \in (-1, +1)^{16}$, in the proposed D-SP-SNN and FSP-SNN as compared to discrete Hopfield network.

The total number of different possible combinations for the initial conditions for this example is 64, 480, 2240 and 7280 for 1-, 2-, 3-, and 4-Hamming distance cases, respectively, which could be calculated by $m_d \times C(16, K)$, where $m_d = 4$ and $K = 1, 2, 3$, and 4.

As seen from Figure 2 the performance of the proposed networks D-SP-SNN and FSP-SNN is the same as that of discrete Hopfield Network for 1-Hamming and 2-Hamming

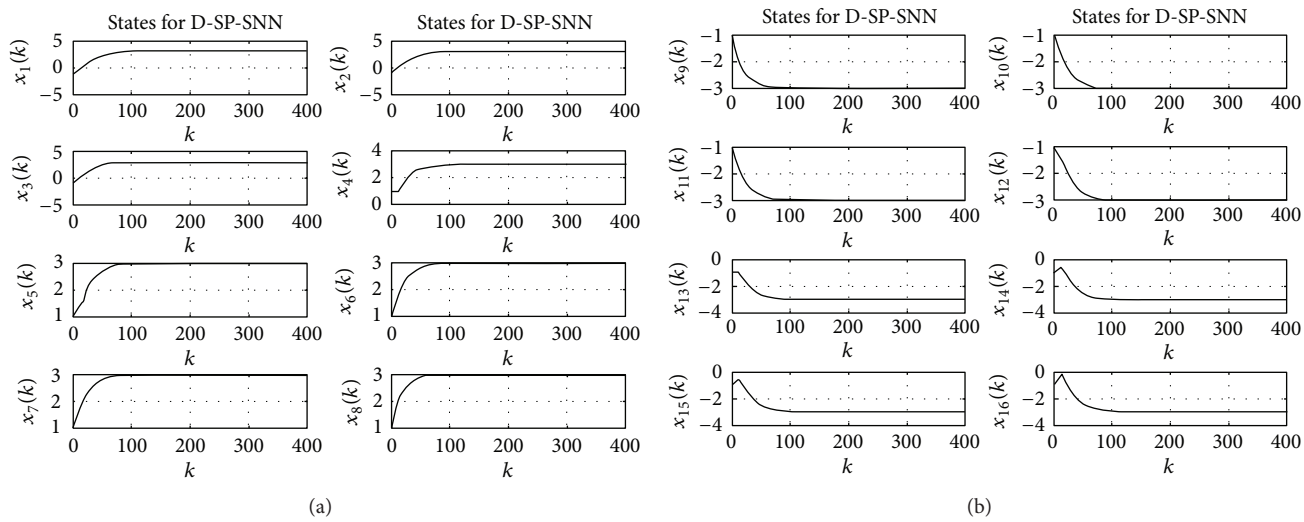

Figure 3: Typical plot for evolutions of states (a) 1 to 8 and (b) 9 to 16 in Example 8 by the D-SP-SNN.

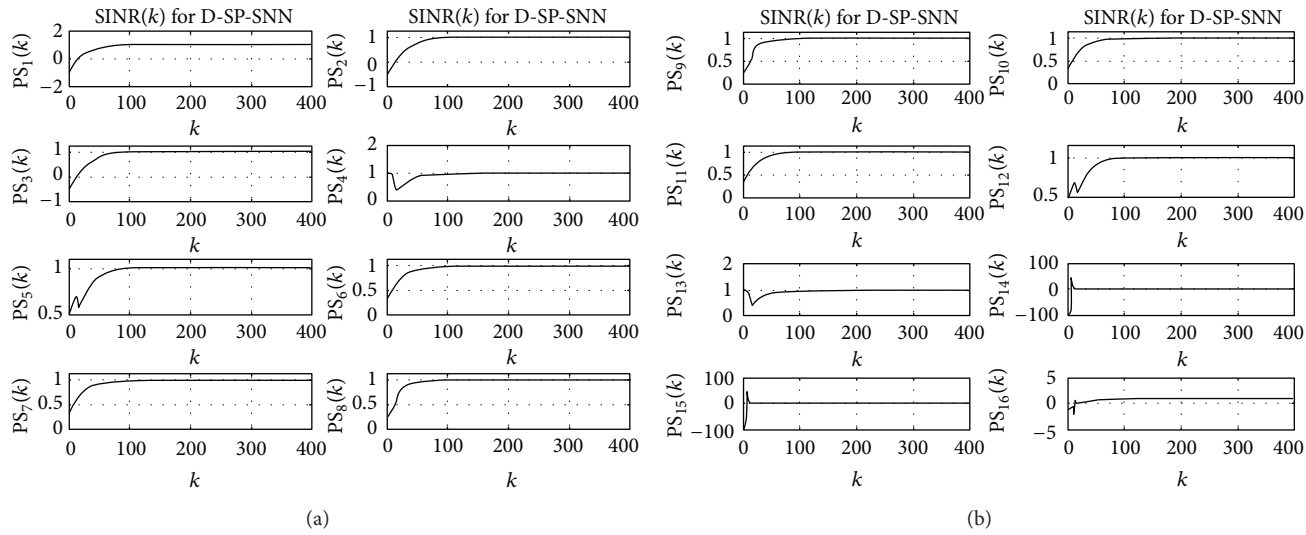

Figure 4: Evolutions of pseudo-SINRs for the states in Figure 3 in Example 8 by the D-SP-SNN. (a) Pseudo-SINRs 1 to 8 and (b) pseudo-SINRs 9 to 16.

distance cases (%100 for all networks). However, the D-SP-SNN and FSP-SNN give better performance than the discrete Hopfield network does for 3- and 4- Hamming distance cases.

Typical plots for evolution of states in Example 8 by the D-SP-SNN are shown in Figure 3. The evolution of corresponding pseudo-SINRs is given by Figure 4. The figure shows that the pseudo-SINRs approach to constant value 1 as states converge to the equilibrium point.

Evolutions of the Lyapunov function in (37) for the states of Figure 3 in Example 8 are given in Figure 5. The figure shows that the proposed D-SP-SNN minimizes the Lyapunov function of Hopfield neural network with the same weight matrix.

Example 9. In Examples 7 and 8, the desired vectors are orthogonal. In this example, the desired vectors represent numbers 1, 2, 3, and 4, which are not orthogonal to each other. The numbers are represented by 25 neurons. The weight matrix is determined by the Hebb learning as in the previous examples. In the rest of the examples in this paper, we set $\sigma_1 = 1$, $\kappa_1 = 10$, $\sigma_2 = 10$, $\kappa_2 = 1$, and $\alpha(k) = 0.01$, for all k.

Figure 6 shows desired pattern 1, a distorted pattern 1 where the Hamming Distance (HD) is 5, the result of HNN-Euler, and the result of the D-SP-SNN using the distorted pattern as initial condition. As seen from the figure, the proposed D-SP-SNN succeeds to recover the number while the HNN-Euler fails for the same parameters and weight matrix.

The evolutions of the Lyapunov function in (37) and the norm of the difference between the state vector and equilibrium point for pattern 1 in Figure 6 are shown in Figure 7. As seen from the figure, (i) the proposed D-SP-SNN

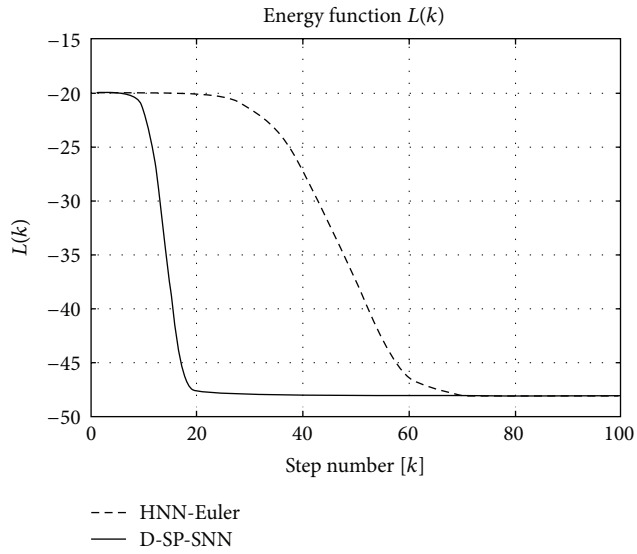

Figure 5: Evolution of Lyapunov function in (37) in Example 8 ($N = 16$).

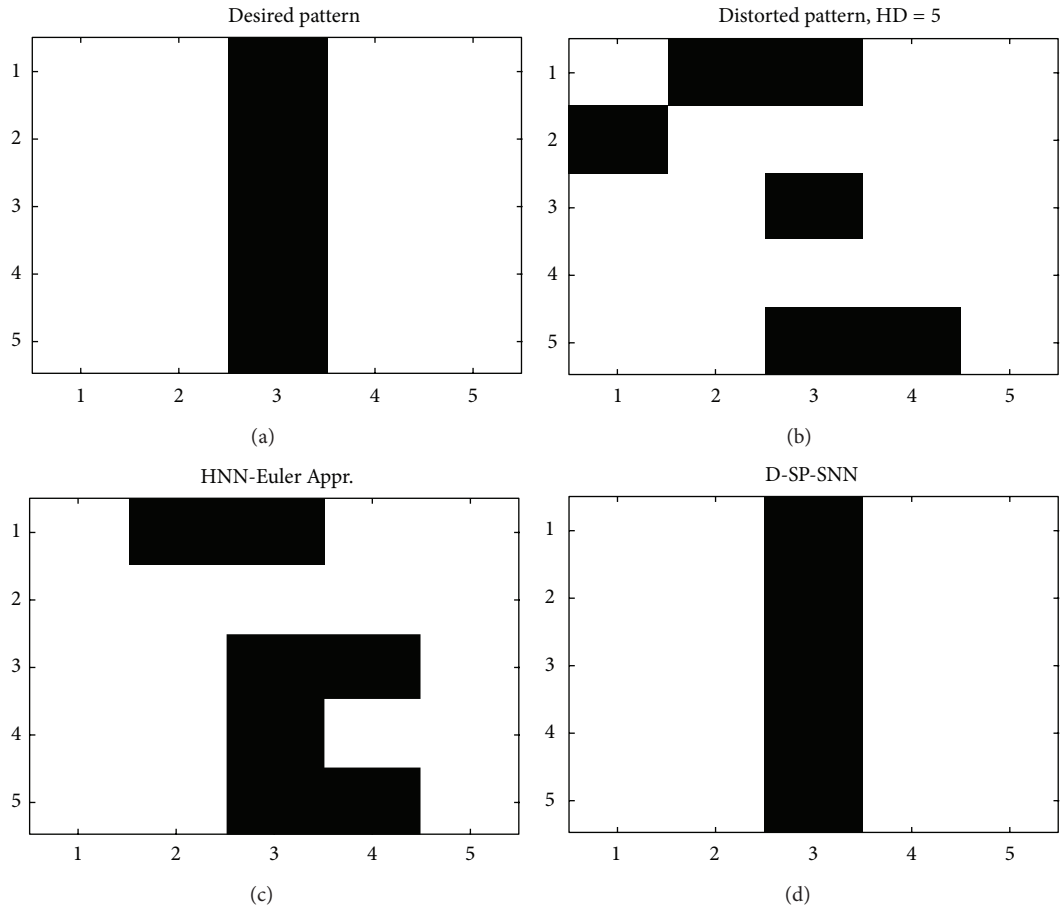

Figure 6: (a) Desired pattern 1, distorted pattern 1 (HD = 5), result of HNN-Euler, and result of D-SP-SNN in Example 9.

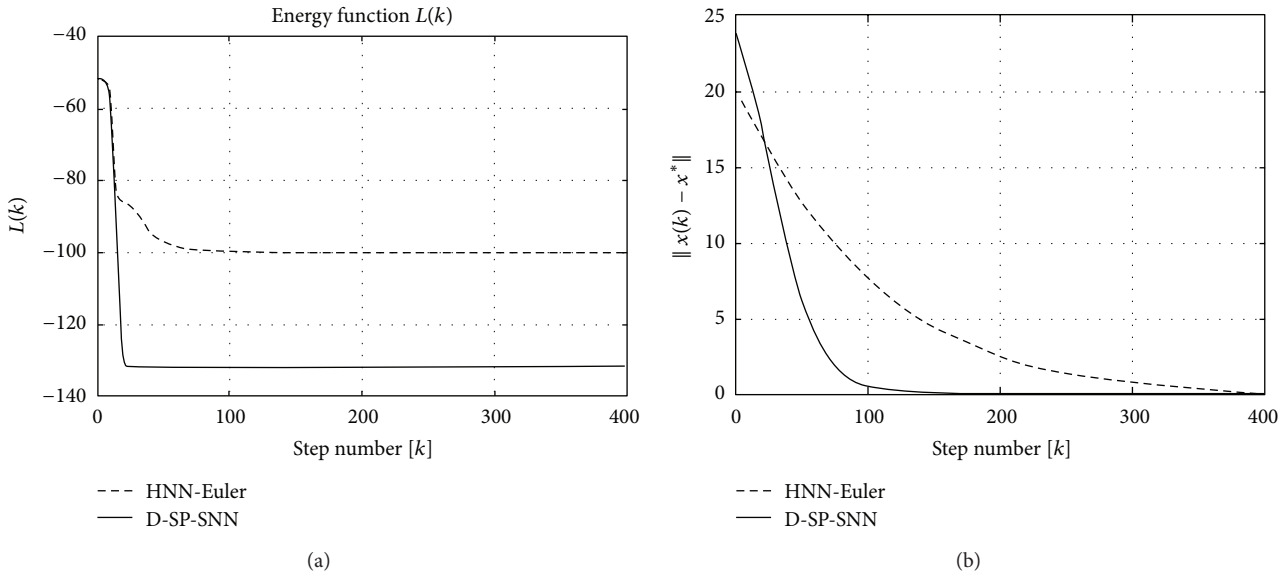

Figure 7: Evolution of (a) Lyapunov function in (37) and (b) norm of the difference between the state vector and equilibrium point in Example 9 for pattern 1 ($N = 25$).

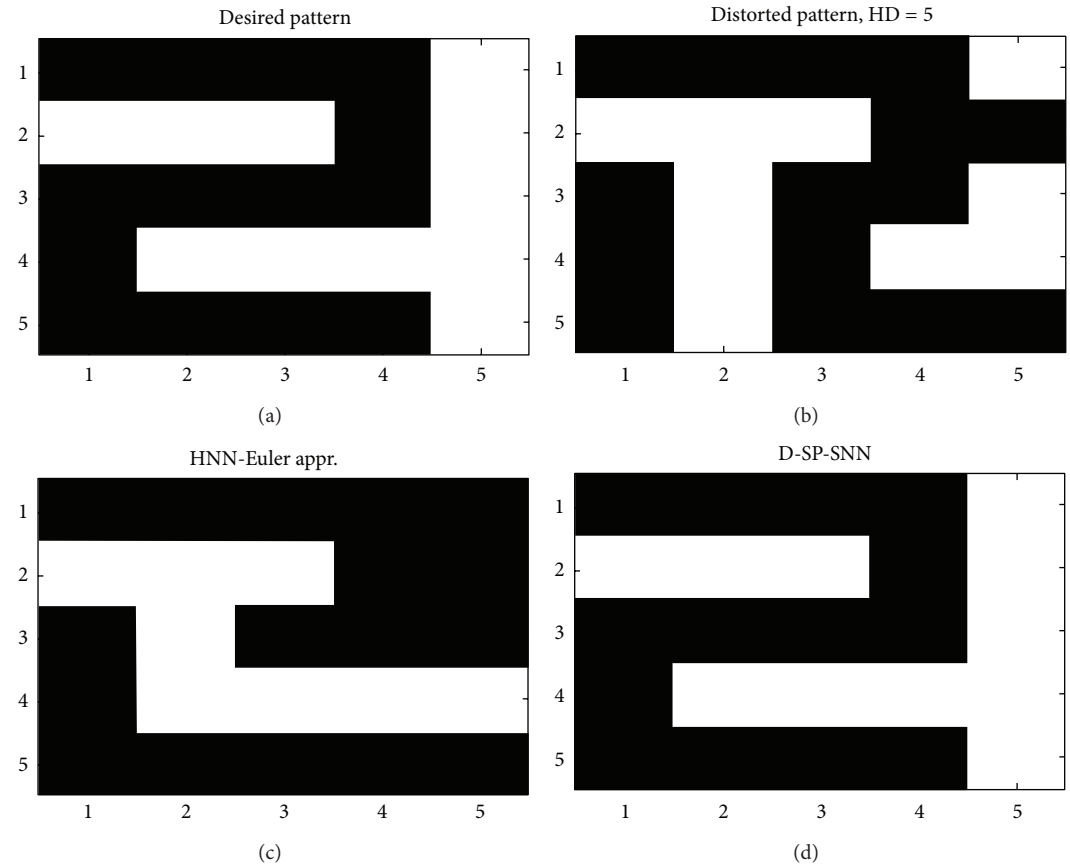

Figure 8: (a) Desired pattern 2, (b) distorted pattern 2 (HD = 5), (c) result of HNN-Euler, and (d) Result of D-SP-SNN in Example 9.

minimizes the Lyapunov function of Hopfield neural network, and (ii) the proposed D-SP-SNN converges faster than its HNN-Euler counterpart with the same weight matrix for this example.

Figure 8 shows desired pattern 2, a distorted pattern 2 where the HD is 5, the result of HNN-Euler, and the result of D-SP-SNN using the distorted pattern as initial condition. As seen from the figure, the proposed D-SP-SNN succeeds to

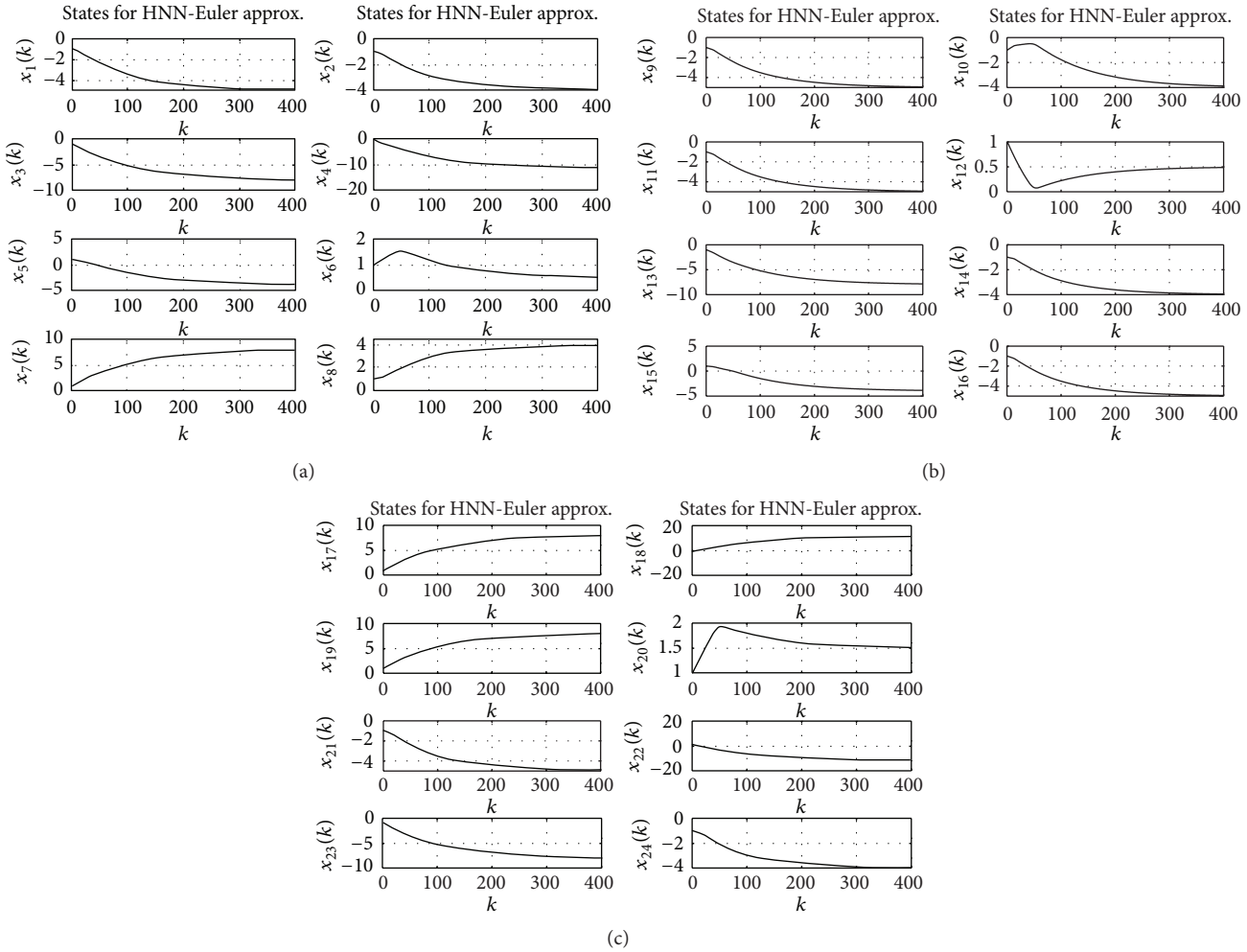

Figure 9: Evolutions of states (a) 1 to 8, (b) 9 to 16, and (c) 17 to 24 in Example 9 for pattern 2 by HNN-Euler.

recover the number while the HNN-Euler fails for the same parameters and weight matrix.

Evolutions of states in Example 9 for pattern 2 by HNN-Euler and by D-SP-SNN are shown in Figures 9 and 10, respectively. The figures show that the states of proposed D-SP-SNN converge faster than those of its HNN-Euler counterpart for the same parameter settings.

Figure 11 shows the evolutions of pseudo-SINRs of states in Example 9 for pattern 2 by D-SP-SNN. The figure shows that the pseudo-SINRs approach to constant value 1 as states converge to the equilibrium point.

The evolutions of the norm of the difference between the state vector and equilibrium point for pattern 2 in Figure 8 are shown in Figure 12. As seen from the figure, the proposed D-SP-SNN converges much faster than its HNN-Euler counterpart.

Figure 13 shows desired pattern 3, a distorted pattern 3 where the HD is 5, the result of HNN-Euler, and the result of the D-SP-SNN using the distorted pattern as initial condition. As seen from the figure, the proposed D-SP-SNN succeeds to recover the number while its HNN-Euler counterpart fails for the same parameters and weight matrix.

The evolutions of the Lyapunov function and the norm of the difference between the state vector and equilibrium point for pattern 3 in Figure 13 are shown in Figure 14. As seen from the figure, (i) the proposed D-SP-SNN minimizes the Lyapunov function of Hopfield neural network, and (ii) the proposed D-SP-SNN converges faster than its HNN-Euler counterpart with the same weight matrix for this example.

Figure 15 shows desired pattern 4, a distorted pattern 4 where the HD is 5, the result of HNN-Euler, and the result of the D-SP-SNN using the distorted pattern as initial condition. As seen from the figure, the proposed D-SP-SNN succeeds to recover the number while its HNN-Euler counterpart fails for the same parameters settings.

The evolutions of the Lyapunov function and the norm of the difference between the state vector and equilibrium point for pattern 4 in Figure 15 are shown in Figure 16. As seen from the figure, (i) the proposed D-SP-SNN minimizes the Lyapunov function of Hopfield Neural Network, and (ii) the proposed D-SP-SNN converges faster than its HNN-Euler counterpart with the same weight matrix for this example.

Example 10. In this and in the following example, we examine the performance of the proposed D-SP-SNN in clustering

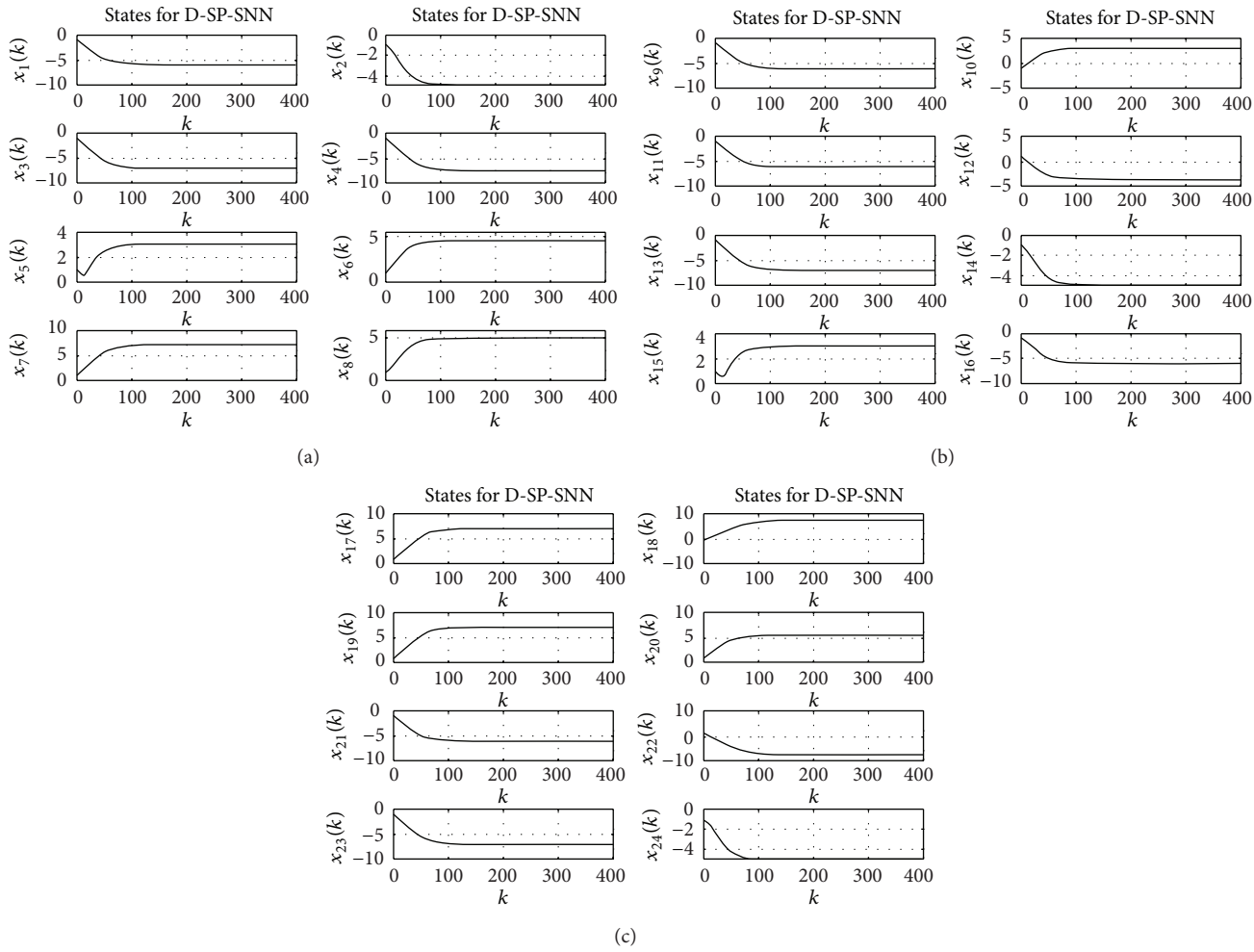

Figure 10: Evolutions of states (a) 1 to 8, (b) 9 to 16, and (c) 17 to 24 in Example 9 for pattern 2 by D-SP-SNN.

problem. Clustering is used in a wide range of applications, such as engineering, biology, marketing, information retrieval, social network analysis, image processing, text mining, finding communities, influencers, and leaders in online or offline social networks. Data clustering is a technique that enables dividing large amounts of data into groups/clusters in an unsupervised manner such that the data points in the same group/cluster are similar and those in different clusters are dissimilar according to some defined similarity criteria. The clustering problem is an NP-complete, and its general solution even for 2-clustering case is not known. It is well known that the clustering problem can be formulated in the form of the Lyapunov function of the HNN. The weight matrix is chosen as the distance matrix of the dataset and is the same for both HNN-Euler and D-SP-SNN.

In what follows, we compare the performance of the proposed D-SP-SNN as compared to its HNN-Euler counterpart as applied to clustering problems for the very same parameter settings. Two-dimensional 16 data points to be bisected are shown in Figure 17. The clustering results are also shown in Figure 17. As seen from the figure, the D-SP-SNN finds the optimum solution for this toy example. HNN-Euler also gives the same solution.

The evolutions of states in the clustering by HNN-Euler and by D-SP-SNN are shown in Figures 18 and 19, respectively. As seen from the figures, the states of the proposed D-SP-SNN converge faster that those of its HNN-Euler counterpart.

The evolutions of psuedo-SINRs of states in the clustering by D-SP-SNN in Example 10 ($N = 16$) are given by Figure 20. The figure shows that the pseudo-SINRs approach to constant value 1 as states converge to the equilibrium point.

The evolutions of Lyapunov function and the norm of the difference between the state vector and equilibrium point in Example 10 are given in Figure 21. The figure confirms that (i) the proposed D-SP-SNN minimizes the Lyapunov function of Hopfield neural network and (ii) the proposed D-SP-SNN converges faster than its HNN-Euler counterpart with the same weight matrix.

Example 11. In this example, there are 40 data points as shown in Figure 22. The figure also shows the bisecting clustering results by k-means algorithm and the proposed D-SP-SNN.

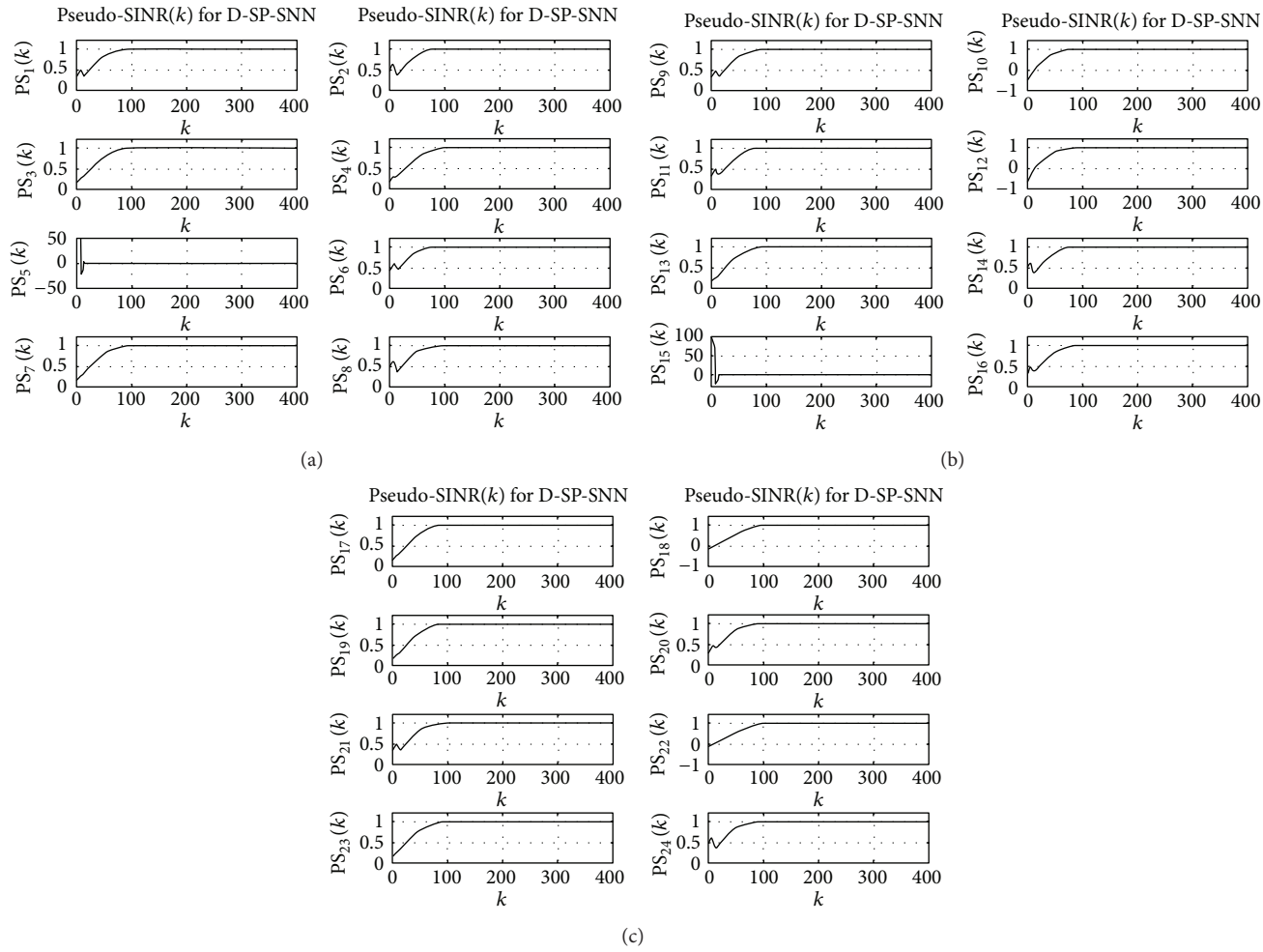

Figure 11: Evolutions of pseudo-SINRs of states (a) 1 to 8, (b) 9 to 16, and (c) 17 to 24 in Example 9 for pattern 2 by D-SP-SNN.

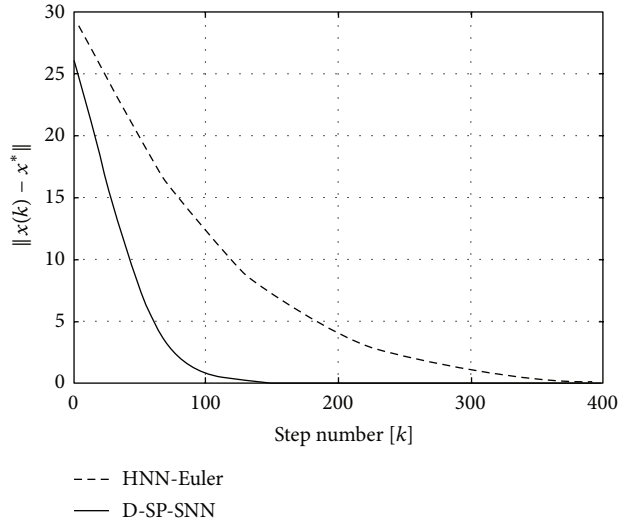

Figure 12: Evolutions of the norm of the difference between the state vector and equilibrium point in Example 9 for pattern 2 ($N = 25$).

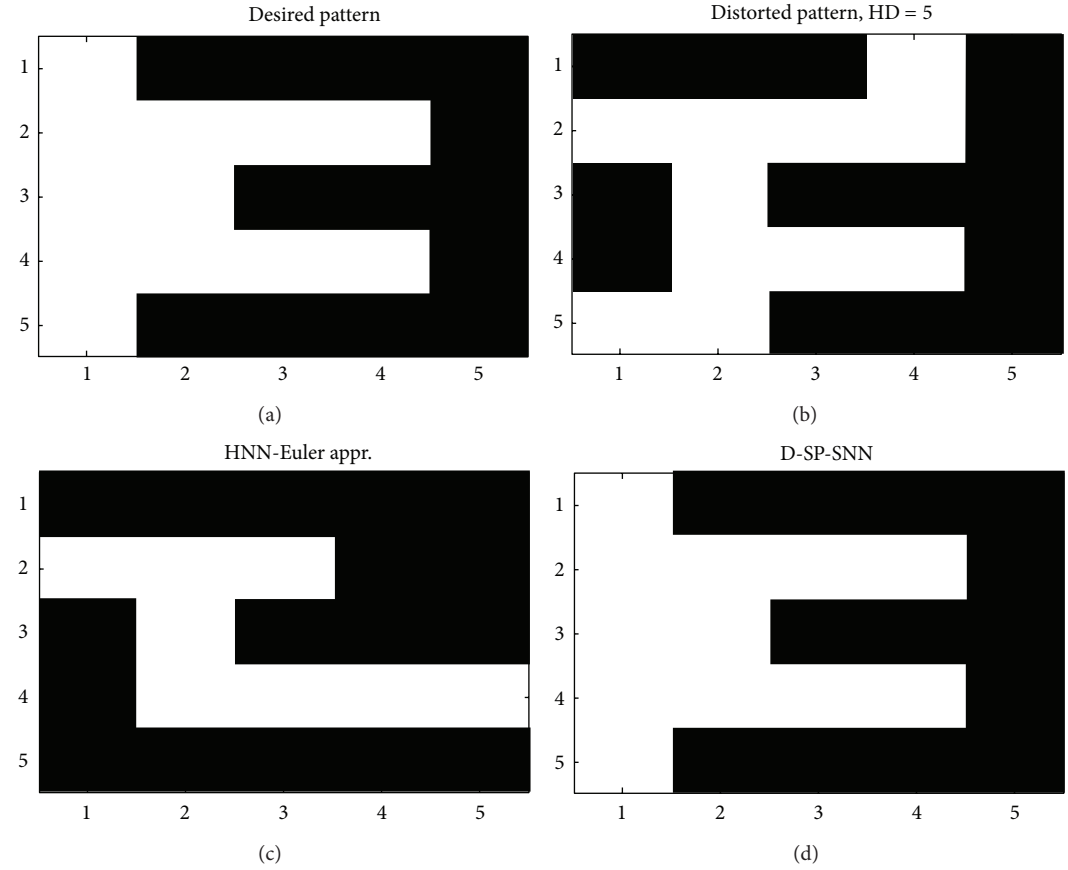

Figure 13: (a) Desired pattern 3, (b) distorted pattern 3 (HD = 5), (c) result of HNN-Euler, and (d) result of D-SP-SNN in Example 9.

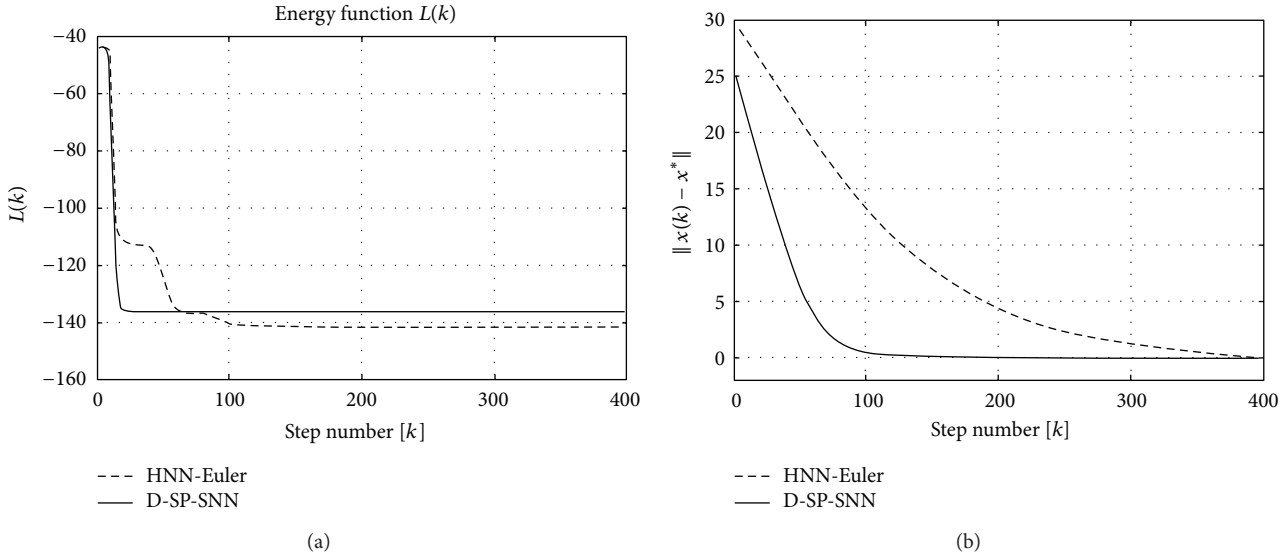

Figure 14: Evolution of (a) Lyapunov function and (b) norm of the difference between the state vector and equilibrium point in Example 9 for pattern 3 ($N = 25$).

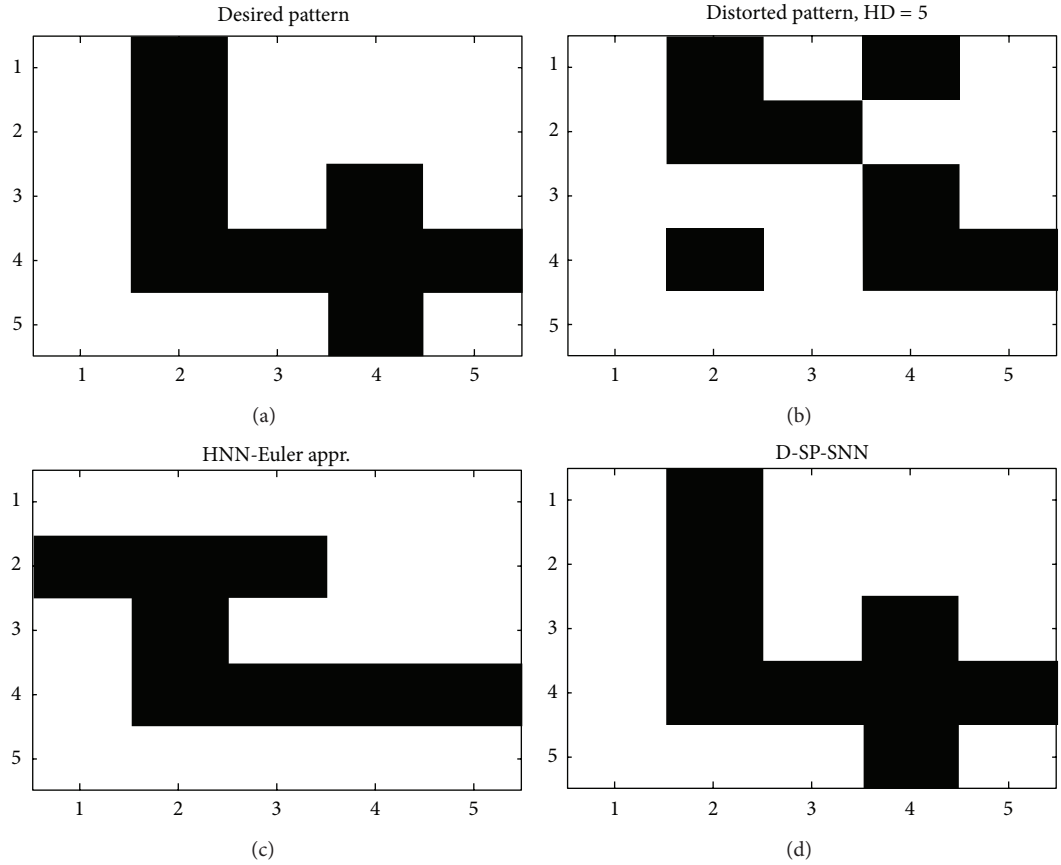

Figure 15: (a) Desired pattern 4, (b) distorted pattern 4 (HD = 5), (c) result of HNN-Euler, and (d) result of D-SP-SNN in Example 9.

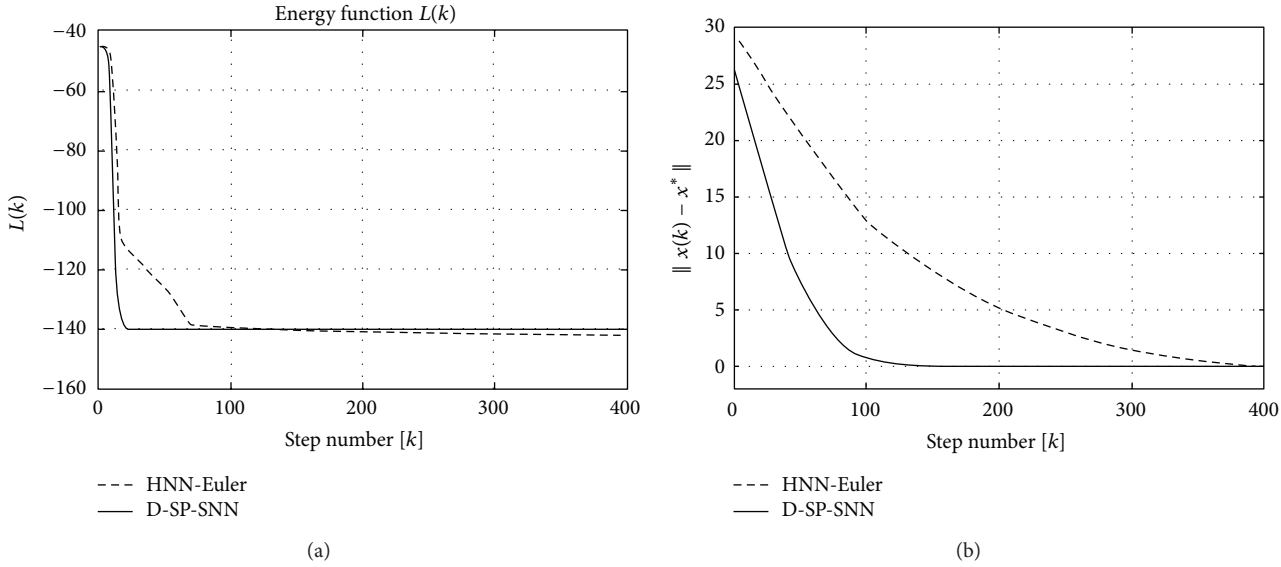

Figure 16: Evolutions of (a) Lyapunov function and (b) norm of the difference between the state vector and equilibrium point in Example 9 for pattern 4 ($N = 25$).

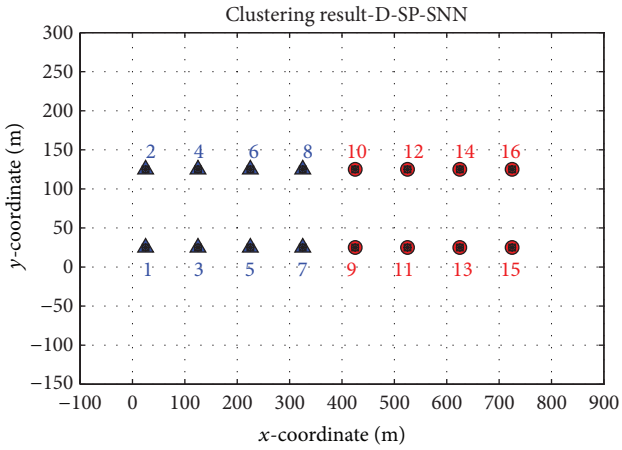

Figure 17: Result of clustering by D-SP-SNN, $N = 16$ in Example 10.

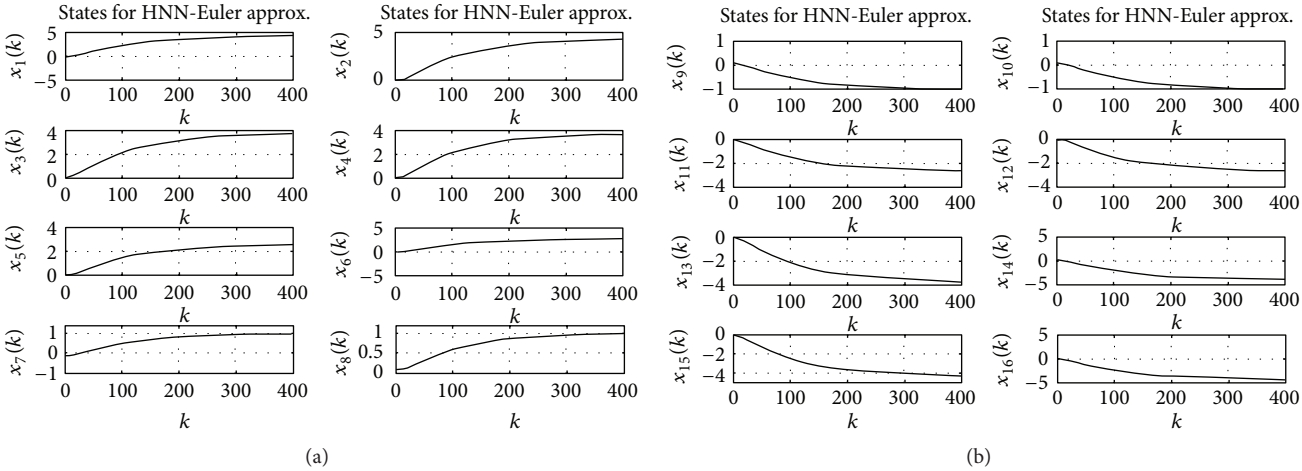

Figure 18: Evolutions of states (a) 1 to 8 and (b) 9 to 16 in the clustering by HNN-Euler in Example 10 ($N = 16$).

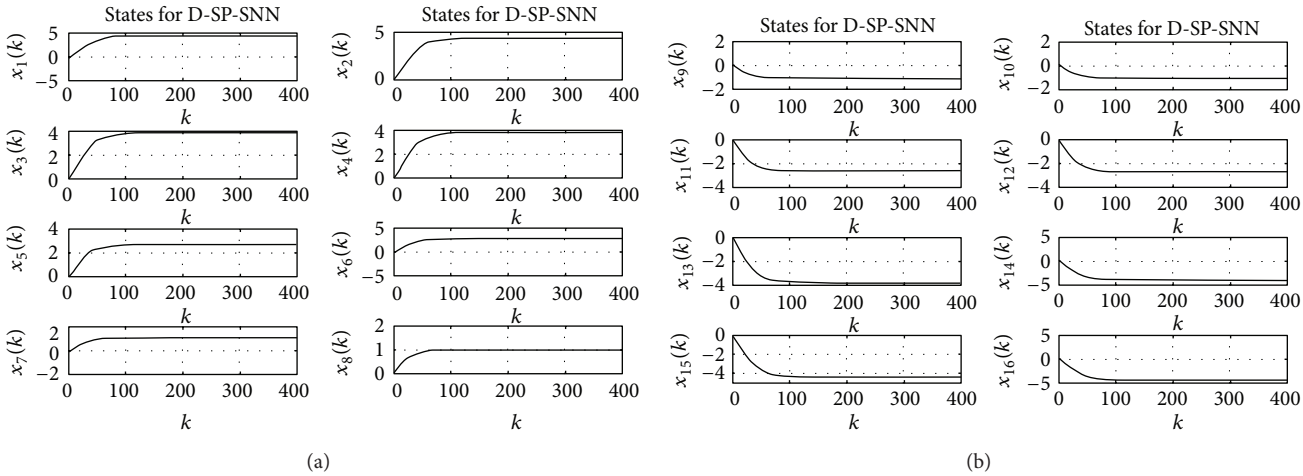

Figure 19: Evolutions of states (a) 1 to 8 and (b) 9 to 16 in the clustering by D-SP-SNN in Example 10 ($N = 16$).

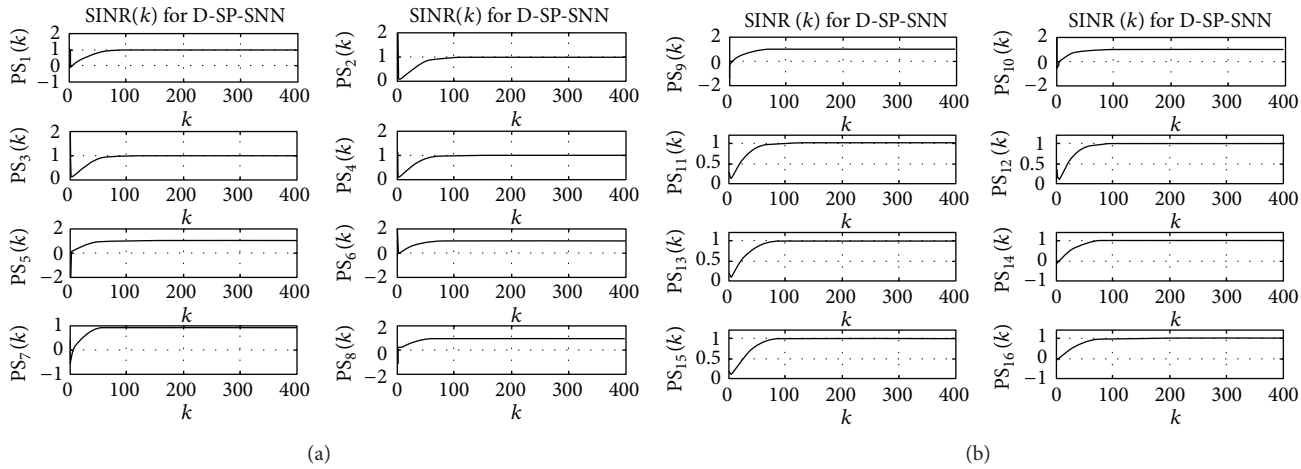

FIGURE 20: Evolutions of psuedo-SINRs of states (a) 1 to 8 and (b) 9 to 16 in the clustering by D-SP-SNN in Example 10 ($N = 16$).

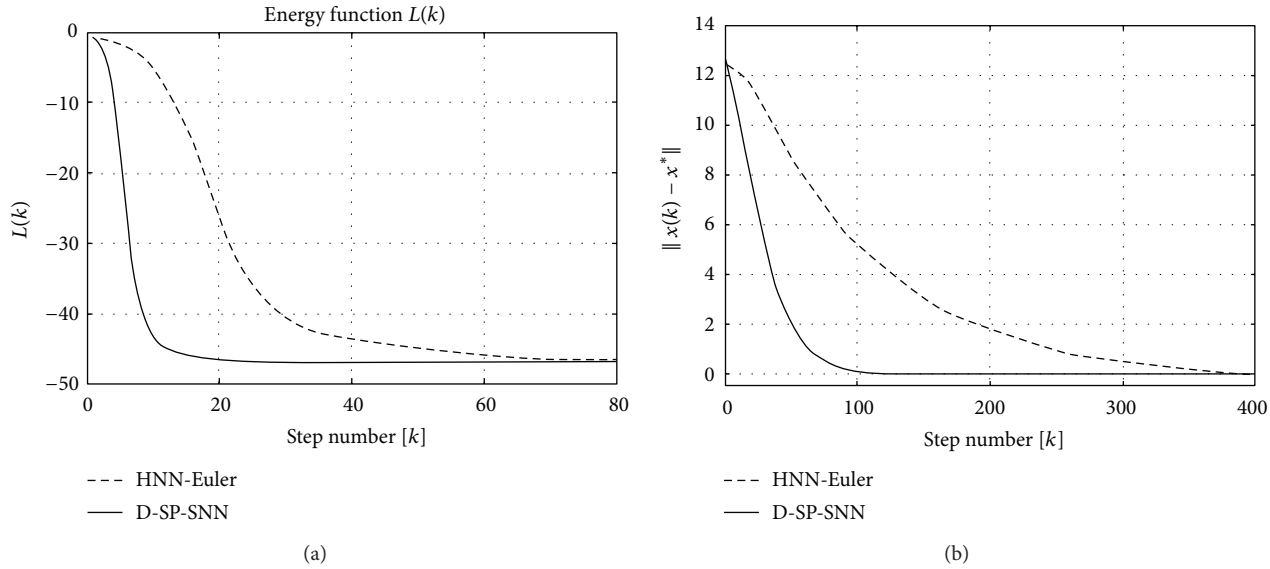

FIGURE 21: Evolution of (a) Lyapunov function and (b) norm of the difference between the state vector and equilibrium point in Example 10 ($N = 16$).

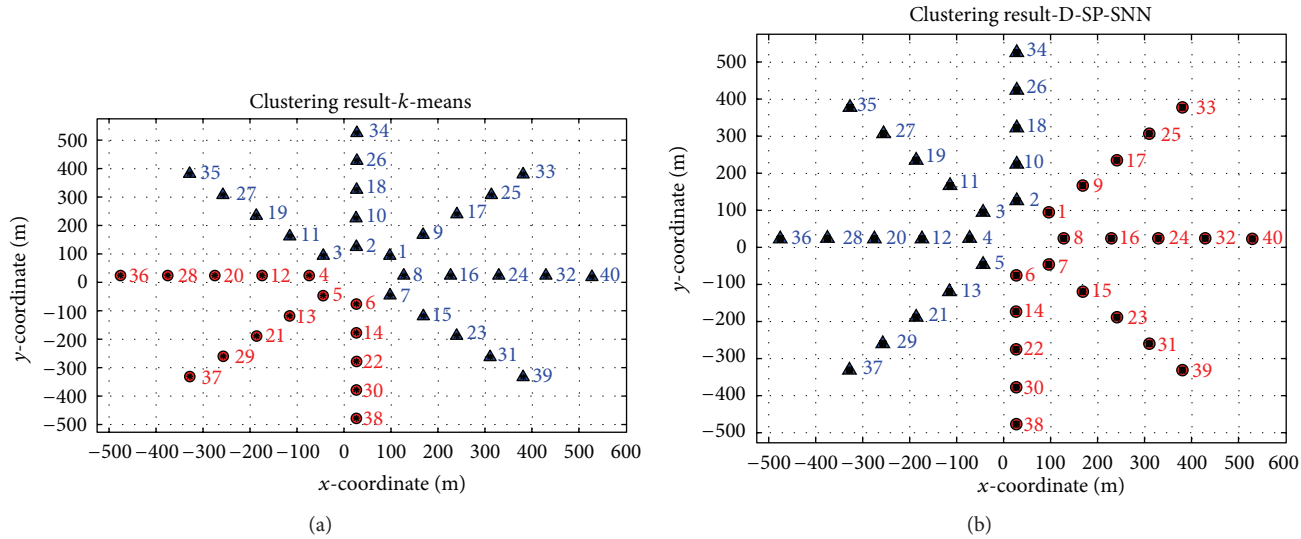

FIGURE 22: Bisecting clustering results by (a) k-means and (b) D-SP-SNN in Example 11 ($N = 40$).

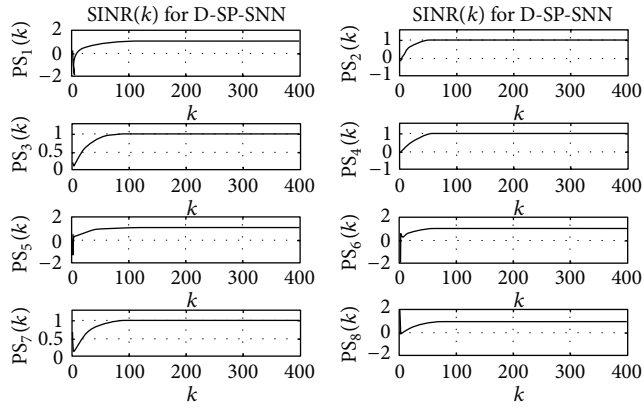

Figure 23: Evolution of pseudo-SINRs of states 1 to 8 in Example 11 by the D-SP-SNN, ($N = 40$).

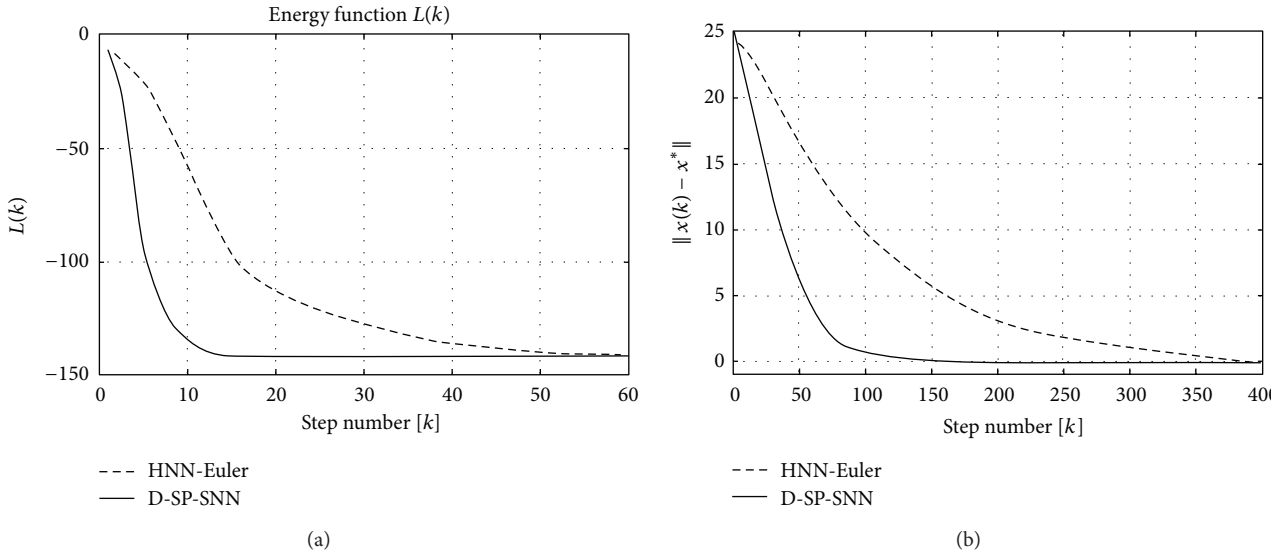

Figure 24: Evolution of (a) Lyapunov function and (b) norm of the difference between the state vector and equilibrium point in Example 11 ($N = 40$).

As seen from the figure, while k-means fail to find the optimum clustering solution for this example (for a randomly given initial values), the proposed D-SP-SNN succeeds in finding the optimum solution (for the same initial values).

Figure 23 shows the evolution of pseudo-SINRs of states by the D-SP-SNN. The figure shows that the pseudo-SINRs approach to constant value 1 as states converge to the equilibrium point, as before.

Evolutions of the Lyapunov function and the norm of the difference between the state vector and equilibrium point are shown in Figure 24. The figure confirms the superior convergence speed of the D-SP-SNN as compared to its HNN-Euler counterpart.

4. Conclusions

In this paper, we present and analyze a discrete recurrent nonlinear system which includes the Hopfield neural networks [1, 2] and the nonlinear sigmoid power control algorithm for cellular radio systems in [13], as special cases by properly choosing the functions. This paper extends the results in [11], which are for autonomous linear systems, to nonlinear case. The proposed system can be viewed as a discrete-time realization of a recently proposed continuous-time network in [12]. In this paper, we focus on discrete-time analysis and present various novel key results concerning the discrete-time dynamics of the proposed system, some of which are as follows: (i) the proposed network is shown to be stable in synchronous and asynchronous work mode in discrete time; (ii) a novel concept called Pseudo-SINR (pseudo-signal-to-interference-noise ratio) is introduced for discrete-time nonlinear systems; (iii) it is shown that when the network approaches one of its equilibrium points, the instantaneous Pseudo-SINRs become equal to a constant target value.

The simulation results confirm the novel results (e.g., Pseudo-SINR convergence, etc.) presented and show a superior performance of the proposed network as compared to its Hopfield network counterpart in various associative memory systems and clustering examples. Moreover, the results show that the proposed network minimizes the Lyapunov function

of the Hopfield neural networks. The disadvantage of the D-SP-SNN is that it increases the computational burden.

Appendices

A. Lipschitz Constant of the Sigmoid Function

In what follows, we will show the sigmoid function ($f(a) = 1 - (2/(1 + \exp(-\sigma a)))$, $\sigma > 0$) that has the global Lipschitz constant $k = 0.5\sigma$. Since $f(\cdot)$ is a differentiable function, we can apply the mean value theorem:

$$f(a) - f(b) = (a-b)\, f'\left(\mu a + (1-\mu)(b-a)\right) \quad \text{with } \mu \in [0,1]. \tag{A.1}$$

The derivative of $f(\cdot)$ is $f'(a) = -2\sigma/e^{\sigma a}(1+e^{-\sigma a})^2$ whose maximum is at the point $a = 0$; that is, $|f'(a)| \leq 0.5\sigma$. So we obtain the following inequality:

$$\left|f(a) - f(b)\right| \leq k\,|a-b|, \tag{A.2}$$

where $k = 0.5\sigma$ is the global Lipschitz constant of the sigmoid function.

B. Outer Product-Based Network Design

Let us assume that L desired prototype vectors are orthogonal and each element of a prototype vector is either −1 or +1.

Step 1. Calculate the sum of outer products of the prototype vectors (Hebb Rule, [19])

$$\mathbf{Q} = \sum_{s=1}^{L} \mathbf{d}_s \mathbf{d}_s^T. \tag{B.1}$$

Step 2. Determine the diagonal matrix $\mathbf{A}$ and $\mathbf{W}$ as follows:

$$a_{ij} = \begin{cases} q_{ii} + \rho & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases} \quad i, j = 1, \ldots, N, \tag{B.2}$$

where ρ is a real number and

$$w_{ij} = \begin{cases} 0 & \text{if } i = j, \\ q_{ij} & \text{if } i \neq j, \end{cases} \quad i, j = 1, \ldots, N, \tag{B.3}$$

where q_{ij} shows the entries of matrix $\mathbf{Q}$, N is the dimension of the vector $\mathbf{x}$, and L is the number of the prototype vectors ($N > L > 0$). In (B.2), $q_{ii} = L$ from (B.1) since $\{\mathbf{d}_s\}$ is from $(-1,+1)^N$. It is observed that $\rho = 0$ gives relatively good performance; however, by examining the nonlinear state equations in Section 2, it can be seen that the proposed networks D-SP-SNN and FSP-SNN contain the prototype vectors at their equilibrium points for a relatively large interval of ρ.

Another choice of ρ in (B.2) is $\rho = N - 2L$ which yields $a_{ii} = N - L$. In what follows we show that this choice also assures that $\{\mathbf{d}_j\}_{j=1}^{L}$ are the equilibrium points of the networks.

From (B.1)–(B.3)

$$[-\mathbf{A} + \mathbf{W}] = -(N-L)\,\mathbf{I} + \sum_{s=1}^{L} \mathbf{d}_s \mathbf{d}_s^T - L\mathbf{I}, \tag{B.4}$$

where $\mathbf{I}$ represents the identity matrix. Since $\mathbf{d}_s \in (-1,+1)^N$, then $||\mathbf{d}_s||_2^2 = N$. Using (B.4) and the orthogonality properties of the set $\{\mathbf{d}_s\}_{s=1}^{L}$ gives the following:

$$[-\mathbf{A} + \mathbf{W}]\,\mathbf{d}_s = -(N-L)\,\mathbf{d}_s + (N-L)\,\mathbf{d}_s = \mathbf{0}. \tag{B.5}$$

So, the prototype vectors $\{\mathbf{d}_j\}_{j=1}^{L}$ correspond to equilibrium points.

References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554–2558, 1982.

[2] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisons in optimization problems," *Biological Cybernetics*, vol. 52, no. 3, pp. 141–152, 1985.

[3] S. Matsuda, "Optimal Hopfield network for combinatorial optimization with linear cost function," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1319–1330, 1998.

[4] K. Smith, M. Palaniswami, and M. Krishnamoorthy, "Neural techniques for combinatorial optimization with applications," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1301–1318, 1998.

[5] J. K. Paik and A. K. Katsaggelos, "Image restoration using a modified Hopfield network," *IEEE Transactions of Image Processing*, vol. 1, no. 1, pp. 49–63, 1992.

[6] G. G. Lendaris, K. Mathia, and R. Saeks, "Linear Hopfield networks and constrained optimization," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 29, no. 1, pp. 114–118, 1999.

[7] J. A. Farrell and A. N. Michel, "A synthesis procedure for Hopfield's continuous-time associative memory," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 7, pp. 877–884, 1990.

[8] M. K. Müezzinoglu, C. Güzeliş, and J. M. Zurada, "An energy function-based design method for discrete Hopfield associative memory with attractive fixed points," *IEEE Transactions on Neural Networks*, vol. 16, no. 2, pp. 370–378, 2005.

[9] J. M. Zurada, *Introduction to Artificial Neural Systems*, West Publishing Company, 1992.

[10] M. Vidyasagar, "Location and stability of the high-gain equilibria of nonlinear neural networks," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 660–672, 1993.

[11] Z. Uykan, "On the SIRs ("Signal" -to- "Interference" -Ratio) in discrete-time autonomous linear networks," in *Proceedings of the 1st International Conference on Advanced Cognitive Technologies and Applications (COGNITIVE '09)*, Athens, Greece, November, 2009.

[12] Z. Uykan, "Fast convergent double-sigmoid hopfield neural network as applied to optimization problems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 990–996, 2013.

[13] Z. Uykan and H. N. Koivo, "Sigmoid-basis nonlinear power-control algorithm for mobile radio systems," *IEEE Transactions on Vehicular Technology*, vol. 53, no. 1, pp. 265–271, 2004.

[14] J. Zander, "Performance of optimum transmitter power control in cellular radio systems," *IEEE Transactions on Vehicular Technology*, vol. 41, pp. 57–62, 1992.

[15] J. Zander, "Distributed cochannel interference control in cellular radio systems," *IEEE Transactions on Vehicular Technology*, vol. 41, pp. 305–311, 1992.

[16] S. Haykin, *Neural Networks*, Macmillan, 1999.

[17] J. van den Berg, "The most general framework of continuous Hopfield neural networks," in *Proceedings of the 1st International Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing (NICROSP '96)*, pp. 92–100, August 1996.

[18] H. Harrer, J. A. Nossek, and F. Zou, "A learning algorithm for time-discrete cellular neural networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN '91)*, pp. 717–722, November 1991.

[19] D. O. Hebb, *The Organization of Behaviour*, John Wiley & Sons, New York, NY, USA, 1949.

[20] M. K. Müezzinoğlu and C. Güzeliş, "A Boolean Hebb rule for binary associative memory design," *IEEE Transactions on Neural Networks*, vol. 15, no. 1, pp. 195–202, 2004.

[21] J. D. Herdtner and E. K. P. Chong, "Analysis of a class of distributed asynchronous power control algorithms for cellular wireless systems," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 436–446, 2000.

[22] D. Kim, "On the convergence of fixed-step power control algorithms with binary feedback for mobile communication systems," *IEEE Transactions on Communications*, vol. 49, no. 2, pp. 249–252, 2001.