Sparsely Interconnected Neural Networks for Associative Memories With Applications to Cellular Neural Networks

Derong Liu *Member, IEEE,* and Anthony N. Michel, *Fellow, IEEE*

***Abstract*—We first present results for the analysis and synthesis of a class of neural networks without any restrictions on the interconnecting structure. The class of neural networks which we consider have the structure of analog Hopfield nets and utilize saturation functions to model the neurons. Our analysis results make it possible to locate in a systematic manner all equilibrium points of the neural network and to determine the stability properties of the equilibrium points. The synthesis procedure makes it possible to design in a systematic manner neural networks (for associative memories) which store all desired memory patterns as reachable memory vectors.**

We generalize the above results to develop a design procedure for neural networks with sparse coefficient matrices. Our results guarantee that the synthesized neural networks have predetermined sparse interconnection structures and store any set of desired memory patterns as reachable memory vectors. We show that a sufficient condition for the existence of a sparse neural network design is self feedback for every neuron in the network. We apply our synthesis procedure to the design of cellular neural networks for associative memories.

Our design procedure for neural networks with sparse interconnecting structure can take into account various problems encountered in VLSI realizations of such networks. For example, our procedure can be used to design neural networks with few or without any line-crossings resulting from the network interconnections. Several specific examples are included to demonstrate the applicability of the methodology advanced herein.

I. Introduction

THE QUALITATIVE ANALYSIS OF single-layered, fully connected neural networks has evoked a great deal of interest in recent years (see, e.g., [1], [3], [11], [12], [15]–[18], [21]–[22], [24]–[26], [28], [33], [35]–[38], [43], [45]). These works are primarily concerned with the existence and the locations (in the state space) of equilibrium points, with the qualitative properties of the equilibria, and with the extent of the basins of attraction of asymptotically stable equilibria (memories). The stability analysis of a class of single layered, sparsely interconnected neural networks-*Cellular Neural Networks*, has also been of recent interest [5], [8], [9], [42], [48].

In [7], [13], [14], [19], [24]–[26], [36]–[41], [46], [47], several synthesis procedures are developed for different types of neural networks models (for an overview of some of these procedures, see [34]). Among these synthesis techniques, the *eigenstructure method* appears to the especially effective. This method has successfully been applied to the synthesis of neural networks defined on hypercubes [26], [38] and the Hopfield model [24]–[26], [44], and it can also be implemented iteratively [47].

Manuscript received November 25, 1992; revised August 27, 1993. This paper was recommended by Associate Editor Gary E. Ford. This work was supported by the National Science Foundation under Grant ECS 91-07728.

D. Liu is with the Electrical and Electronics Research Department, General Motors NAO R&D Center, Warren, MI. A. N. Michel is with the Department of Electrical Engineering, University of Notre Dame, IN.

IEEE Log Number 9400239.

In the present paper, we consider neural networks described by equations of the form

$$\begin{cases} \dot{x} = -Ax + T\ \mathrm{sat}(x) + I \\ y = \mathrm{sat}(x) \end{cases} \tag{1}$$

where $x \in R^n$ is the state vector, $\dot{x}$ denotes the derivative of x with respect to time $t, y \in D^n \triangleq \{x \in R^n : -1 \leq x_i \leq 1, i = 1, \ldots, n\}$ is the output vector, $A = \mathrm{diag}[a_1, \ldots, a_n]$ with $a_i > 0$ for $i = 1, \ldots, n, T = [T_{ij}] \in R^{n\times n}$ is the coefficient (or connection) matrix, $I = [I_1, \ldots, I_n]^T \in R^n$ is a bias vector, and $\mathrm{sat}(x) = [\mathrm{sat}(x_1), \ldots, \mathrm{sat}(x_n)]^T$ represents the activation function, where

$$\mathrm{sat}(x_i) = \begin{cases} 1, & x_i > 1 \\ x_i, & -1 \leq x_i \leq 1 \\ -1, & x_i < -1 \end{cases}.$$

We asume that the initial states of (1) satisfy $|x_i(0)| \leq 1$ for $i = 1, \ldots, n$.

System (1) is a variant of the analog Hopfield model with activation function sat$(\cdot)$. In the analog Hopfield model [20], one requires that T be symmetric. We do not make this assumption for (1). If in (1), the n neurons are arranged in an $M \times N$ array, where $n = M \times N$, and if we consider only local interconnections, then (1) reduces to a two-dimensional *cellular* neural network (see (20) in Section V). In this paper, we consider a more general case. Specifically, we consider system (1) with *arbitrary* interconnections which includes *fully interconnected* nets and cellular neural networks as special cases.

In the present paper, we concern ourselves primarily with the implementation of *associative memories* by means of artificial neural networks (modeled by (1)). One of the major difficulties encountered in VLSI implementations of artificial neural networks is the realization of extremely large numbers of interconnections in the networks. To reduce the number of connections is of great interest from a practical point of view. Most of the existing synthesis procedures for associative memories [13], [14], [19], [24]–[26], [36]–[41], [46], and [47] were developed for fully interconnected neural networks and none of them result in neural networks with *prespecified* partial or sparse interconnection structure. Synthesis procedures for neural networks with arbitrarily (prespecified) sparse interconnection structure, or equivalently, with sparse coefficient matrices constitute a major addition to the development of neural network theory, and such procedures will have potentially many practical applications, especially in the areas of associative memories and pattern recognition (we will define the exact meaning of sparse coefficient matrix later).

Existing work dealing with sparsely interconnected neural networks has been reported in [23]. In this work, a possible solution of transforming a given neural network into a partially connected or cellular network is presented for the discrete-time Hopfield model. However, as pointed out by the author of [23], "the application of the suggested transformation algorithm is severely limited by its quickly growing complexity."

In Section III of the present paper, we will employ the techniques developed in [26] and [38] to establish results concerning the qualitative properties of neural networks described by (1) and to provide a synthesis procedure for such systems. Using the results of Section III, we will develop in Section IV a synthesis procedure for neural networks with sparse coefficient matrices in which the interconnection structure is predetermined. We will develop this synthesis procedure for neural network (1); however, our method is also applicable to other types of neural network models, such as neural networks defined on hypercubes [26], [38] and the Hopfield model [19], [20]. In Section V, we apply the sparse synthesis technique developed in Section IV to the design of a class of (nonsymmetric) cellular neural networks. We will show that under certain restrictions on the interconnection structure, system (1) is equivalent to a class of zero-input, nonsymmetric cellular neural networks. In Section VI, we consider several specific examples to demonstrate the applicability of our analysis and synthesis procedures. Special emphasis is placed on cellular neural networks and networks with different sparse interconnection structures. We conclude with several pertinent remarks in Section VII.

In our examples (Section VI) we will conduct extensive simulations of system (1), using the difference equations

$$\begin{cases} x_i((k+1)h) = [x_i(kh) + h\sum_{j=1}^{n} T_{ij}y_j(kh) \\ \qquad + \frac{I_i}{a_i}(e^{a_ih}-1)]e^{-a_ih} \\ y_i(kh) = \mathrm{sat}(x_i(kh)) \\ \qquad k = 0, 1, \ldots, \end{cases} \tag{2}$$

$i = 1, \ldots, n$, where h is the step size.

II. Notation

Let V and W be arbitrary sets. Then $V \cup W, V \cap W$, and $V - W$ denote the union, intersection, and difference of V and W, respectively. If V is a subset of W, we write $V \subset W$ and if x is an element of V, we write $x \in V$. Let ϕ denote the empty set. Let R denote the set of real numbers and let R^n be real n-space. If $x \in R^n$, then $x^T = [x_1, \ldots, x_n]$ denotes the transpose of x. If $V \subset R^n$, then $\overline{V}, V^0$ and ∂V represent the closure, interior and boundary of V in R^n, respectively. Let $B^n = \{x \in R^n : x_i = 1 \text{ or } -1, i = 1, \ldots, n\}$ and $D^n = \{x \in R^n : -1 \le x_i \le 1, i = 1, \ldots, n\}$. If $A = [A_{ij}]$ is an arbitrary matrix, then A^T denotes the transpose of A. If A is a square matrix, we use $\lambda(A)$ to denote eigenvalues of A. Let $P(n)$ denote the set of all permutations on $\{1, \ldots, n\}$. If $\{x_1, \ldots, x_m\} \subset R^n$, then $\mathrm{Span}(x_1, \ldots, x_m)$ denotes the linear subspace of R^n generated by $x_1, \ldots, x_m$ and Aspan $(x_1, \ldots, x_m)$ dentoes the affine subspace of R^n generated by $x_1, \ldots, x_m$. if $x_0 \in R^n$ and L is a linear subspace of R^n, then $L + x_0$ dentoes the affine subspace of R^n produced by shifting L by x_0, that is, $L + x_0 = \{y \in R^n : y = x + x_0, x \in L\}$. In particular, $\mathrm{Aspan}(x_1, \ldots, x_m) = \mathrm{Span}(x_1 - x_m, \ldots, x_{m-1} - x_m) + x_m$.

III. Analysis and Synthesis of Neural Networks With Arbitrary Interconnections

In this section, we present results which characterize the qualitative behavior of system (1) and present a synthesis procedure for system (1) by utilizing techniques similar to those developed in [26], [38]. We first introduce the following notation.

For each integer $m, 0 \le m \le n$, let

$$\Lambda_m = \{\xi = [\xi_1, \ldots, \xi_n]^T \in \Lambda : \xi_{\sigma(i)} = 0, 1 \le i \le m,$$

$$\text{and } \xi_{\sigma(i)} = \pm 1, m < i \le n, \text{ for some } \sigma \in P(n)\}$$

and $\Lambda = \{\xi = [\xi_1, \ldots, \xi_n]^T : \xi_i = \pm 1 \text{ or } 0, 1 \le i \le n\}$ and $P(n)$ denotes the set of all permutations on $\{1, \ldots, n\}$. (Recall that there are $n!$ elements in $P(n)$). For each $\xi \in \Lambda$, let

$$C(\xi) = \{x = [x_1, \ldots, x_n]^T \in R^n : |x_i| < 1 \text{ if } \xi_i = 0,$$
$$x_i \ge 1 \text{ if } \xi_i = 1, \text{ and } x_i \le -1 \text{ if } \xi_i = -1\}.$$

From the notation given above, we have

Lemma 1: 1) $\Lambda = \cup_{m=0}^{n} \Lambda_m$. 2) $\Lambda_0 = B^n$ *and* $C(\xi) = \{x \in R^n : |x_i| \ge 1, x_i\xi_i > 0, i = 1, \ldots, n\}$ *for any* $\xi \in \Lambda_0$. 3) $\Lambda_n = \{0\}$ *and* $C(0) = (D^n)^0 = \{x \in R^n : -1 < x_i < 1, i = 1, \ldots, n\}$. 4) $R^n = \cup_{m=0}^{n}\{C(\xi), \xi \in \Lambda_m\}$. 5) *For any* $\xi, \eta \in \Lambda, \xi \ne \eta, C(\xi) \cap C(\eta) = \phi$. □

Suppose that $\xi \in \Lambda_m$ and $\sigma \in P(n)$ such that

$$\xi_{\sigma(i)} = 0, 1 \le i \le m \text{ and } \xi_{\sigma(i)} = \pm 1, m < i \le n. \tag{3}$$

We denote

$$\begin{aligned} A_I &= \mathrm{diag}[a_{\sigma(1)}, \ldots, a_{\sigma(m)}], \\ A_{II} &= \mathrm{diag}[a_{\sigma(m+1)}, \ldots, a_{\sigma(n)}], \\ T_{I,I} &= [T_{\sigma(i)\sigma(j)}]_{1 \le i,j \le m}, \\ T_{I,II} &= [T_{\sigma(i)\sigma(j)}]_{1 \le i \le m, m < j \le n}, \\ T_{II,I} &= [T_{\sigma(i)\sigma(j)}]_{m < i \le n, 1 \le j \le m}, \\ T_{II,II} &= [T_{\sigma(i)\sigma(j)}]_{m < i,j \le n}, \\ I_I &= [I_{\sigma(1)}, \ldots, I_{\sigma(m)}]^T, \\ I_{II} &= [I_{\sigma(m+1)}, \ldots, I_{\sigma(n)}]^T, \end{aligned}$$

and

$$\xi_I = [\xi_{\sigma(1)}, \ldots, \xi_{\sigma(m)}]^T, \quad \xi_{II} = [\xi_{\sigma(m+1)}, \ldots, \xi_{\sigma(n)}]^T.$$

Remark 1: For a given $\xi \in \Lambda_m$, there may exist different elements in $P(n)$ for which (3) is true. For these different permutations, the notation given above will be the same up to different orders in the components. Thus, the subsequent analysis and conclusions will be identical for any of the permutations used.

Remark 2: If $m = n$, we have $A_I = A, T_{I,I} = T, I_I = I, \xi_I = \xi$ and the $A_{II}, T_{I,II}, T_{II,I}, T_{II,II}, I_{II}, \xi_{II}$ do not exist. If $m = 0$, we have $A_{II} = A, T_{II,II} = T, I_{II} = I, \xi_{II} = \xi$ and the $A_I, T_{I,I}, T_{I,II}, T_{II,I}, I_I, \xi_I$ do not exist.

A. Analysis

Consider $\xi \in \Lambda_m, 0 < m < n$, with $\sigma \in \mathrm{P}(n)$ such that $\xi_{\sigma(i)} = 0, 1 \le i \le m$, and $\xi_{\sigma(i)} = \pm 1, m < i \le n$. We can rewrite the first equation of system (1) as

$$\begin{cases} \dot{x}_I = -A_I x_I + T_{I,I} x_I + T_{I,II}\xi_{II} + I_I \\ \dot{x}_{II} = -A_{II} x_{II} + T_{II,I} x_I + T_{II,II}\xi_{II} + I_{II} \end{cases} \tag{4}$$

wherer $\xi_{II} = [\xi_{\sigma(m+1)}, \cdots, \xi_{\sigma(n)}]^T$, $x_I = [x_{\sigma(1)}, \cdots, x_{\sigma(m)}]^T$ with $-1 < x_{\sigma(i)} < 1$ for $1 \le i \le m$, and $x_{II} = [x_{\sigma(m+1)}, \cdots, x_{\sigma(n)}]^T$ with $\xi_{\sigma(i)} x_{\sigma(i)} \ge 1$ for $m < i \le n$. Equation (4) is said to be an *equivalent linear representation of* (1) *over the region* $C(\xi)$.

When $m = n, \Lambda_n = \{0\}$. In this case, for $x \in C(0) = (D^n)^0$, system (1) becomes

$$\dot{x} = (T - A)x + I. \tag{5}$$

When $m = 0, \Lambda_0 = B^n$. In this case, for $\xi \in \Lambda_0, x \in C(\xi)$, system (1) can be expressed as

$$\dot{x} = -Ax + T\xi + I. \tag{6}$$

We will have occasion to make use of the follwing hypotheses for system (1).

Assumption (A): For any m, $0 < m \le n$, and for any $\xi \in \Lambda_m$, the $m \times m$ matrix $T_{I,I} - A_I = [T_{\sigma(i)\sigma(j)}]_{1 \le i,j \le m} - \mathrm{diag}[a_{\sigma(i)}, \cdots, a_{\sigma(m)}]$ is non-singular, where $\sigma \in P(n)$ so that $\xi_{\sigma(i)} = 0, 1 \le i \le m$ and $\xi_{\sigma(i)} = \pm 1, m < i \le n$. □

For system (1) satisfying Assumption (A) we will employ the following notation.

1) If $\xi = 0 \in \Lambda_n (m = n)$, let

$$x_\xi = (A - T)^{-1} I. \tag{7}$$

2) For $\xi \in \Lambda_m$, $0 < m < n$, with A_I, A_{II}, $T_{I,I}, \cdots$, $T_{II,II}$, I_I, I_{II} defined above, let

$$x_\xi = [x_{\xi 1}, \cdots, x_{\xi n}]^T \in R^n \tag{8}$$

where

$$\begin{aligned} x_{\xi I} &= [x_{\xi\sigma(1)}, \cdots, x_{\xi\sigma(m)}]^T \\ &= (A_I - T_{I,I})^{-1}(T_{I,II}\xi_{II} + I_I), \end{aligned}$$

and

$$\begin{aligned} x_{\xi II} &= [x_{\xi\sigma(m+1)}, \cdots, x_{\xi\sigma(n)}]^T \\ &= A_{II}^{-1}(T_{II,I} x_{\xi I} + T_{II,II}\xi_{II} + I_{II}). \end{aligned}$$

3) For $\xi \in \Lambda_0 = B^n (m = 0)$, let

$$x_\xi = A^{-1}(T\xi + I). \tag{9}$$

4) For the x_ξ defined above, let $y_\xi = \mathrm{sat}(x_\xi)$.

Assumption (B): With the notation given above, assume that for any $\xi \in \Lambda_m, 0 \le m \le n, x_\xi \notin \partial(C(\xi))$. □

The following result enables us to locate in a systematic manner all equilibria for system (1) and to ascertain the stability properties for these equilibria. Furthermore, this result will serve as the theoretical basis of the synthesis procedures which we will present in the sequel.

Theorem 1: Suppose that (1) satisfies Assumptions (A) and (B). For any $m, 0 \le m \le n$, and for any $\xi \in \Lambda_m$, we have:

Case I $m = n, \xi = 0 \in \Lambda_n$. (Note that in this case $C(\xi) = (D^n)^0$.)

1) *If $x_\xi \notin (D^n)^0$, there is no equilibrium point of system* (1) *in* $(D^n)^0$.
2) *If $x_\xi \in (D^n)^0$, x_ξ is the unique equilibrium point of system* (1) *in* $(D^n)^0$. *In particular,*
   (i) *if $T - A$ has one or more eigenvalues with non-negative real parts, x_ξ is unstable, and*
   (ii) *if all eigenvalues of $T - A$ have negative real parts, x_ξ is asymptotically stable.*

Case II $0 < m < n, \xi \in \Lambda_m$.

1) *If $x_\xi \notin C(\xi)$, there is no equilibrium point of system (1) in $C(\xi)$.*
2) *If $x_\xi \in C(\xi), x_\xi$ is the unique equilibrium point of system (1) in $C(\xi)$. In particular,*
   (i) *if $T_{I,I} - A_I$ has one or more eigenvalues with non-negative real parts, x_ξ is unstable, and*
   (ii) *if all eigenvalues of $T_{I,I} - A_I$ have negative real parts, x_ξ is asymptotically stable.*

Case III $m = 0, \xi \in \Lambda_0 = B^n$

1) *If $x_\xi \notin C(\xi)$, there is no equilibrium point of system (1) in $C(\xi)$.*
2) *If $x_\xi \in C(\xi)$, x_ξ is an asymptotically stable equilibrium point of system* (1).

Proof: For each $\xi \in \Lambda_m, 0 \le m \le n$, consider (4)–(6). Using similar arguments as in [26], the conclusions of this theorem follow directly from the theory of linear differential equations. □

If x_ξ is an asymptotically stable equilibrium point of system (1), y_ξ is said to be a *memory vector* of system (1). A memory vector y_ξ is said to be *reachable* if there exists a neighborhood V of y_ξ such that for any $x(0) \in V \cap D^n \neq \phi$, the output vector $y(t)$ of system (1) tends to y_ξ asymptotically as $t \to \infty$. Using the results given in Theorem 1, it can easily be shown that a memory vector $y_\xi \in (D^n)^0$ or $y_\xi \in B^n$ is always reachable. When a memory vector $y_\xi \in \partial(D^n) - B^n, y_\xi$ is reachable if and only if for *every* neighborhood U of y_ξ, the set $U \cap D^n$ has a non-empty intersection with the domain of attraction of the corresponding asymptotically stable equilibrium point x_ξ. In our synthesis procedures, the objective is to store patterns in B^n. If we can guarantee that a desired set of bipolar patterns is stored as a set of memory vectors, then such vectors will always be reachable. Therefore, in the sequel, we will drop the modifier "reachable" when the context is clear.

Remark 3: With T symmetric, the function $E : D^n \to R$ defined by

$$E(y) = -\frac{1}{2}y^T Ty + \frac{1}{2}y^T Ay - y^T I \tag{10}$$

can be shown to be monotonically decreasing in time t along the solutions of (1). To see this, we follow the same procedure as in [26] by defining a *local solution* for system (1); or, we can follow the procedure in [9], by defining the derivatives dy_i/dx_i at the breaking points $|x_i| = 1$ to be zero. This guarantees that system (1) will neither oscillate nor become chaotic. When T is nonsymmetric, the function E defined in (10) is not necessarily monotonically decreasing, and oscillatory solutions for (1) may exist. □

Remark 4: It is possible to generalize Theorem 1 to a result which does not require Assumption (B). However, for such a case, the conclusions of the theorem will be less straightforward. □

Several important conclusions can be drawn from Theorem 1 which are given in the following.

Corollary 1: Suppose that in system (1) *T is symmetric and that the coefficients of system* (1) *satisfy the conditions that*

$$T_{ii} > a_i \text{ for } i = 1, \ldots, n. \tag{11}$$

Then, every asymptotically stable equilibrium point $x_e = [x_{e1}, \ldots, x_{en}]^T$ *of system* (1) *satisfies the condition that*

$$|x_{ei}| > 1, \text{ for } i = 1, \ldots, n. \tag{12}$$

Proof: If (11) is satisfied, $T_{I,I} - A_I$ and $T - A$ will have one or more non-negative eigenvalues, since $T_{I,I} - A_I$ and $T - A$ are symmetric matrices with positive diagonal elements. Thus, equilibrium conditions in cases I and II of Theorem 1 can never be satisfied. The only equilibrium conditions for (1) which may be satisfied are those that apply to Case III. It is clear that every equilibrium of (1) which satisfies the conditions in Case III of Theorem 1 has the property given in (12). □

We point out that the same result as Corollary 1 has been proved for cellular neural networks (with *symmetric* interconnections) in [9], using a different approach.

Corrolary 2 Every asymptotically stable equilibrium point $x_e = [x_{e1}, \ldots, x_{en}]^T$ *of* (1) *satisfies condition* (12) *if*

$$T_{ii} > a_i + \sum_{j=1, j\neq i}^{n} |T_{ij}|, \text{ for } i = 1, \ldots, n. \tag{13}$$

Proof: If (13) is satisfied, $T_{I,I} - A_I$ and $T - A$ become matrices with positive diagonal elements and satisfy a diagonal dominance condition. By Geršgorin's Theorem [27], all eigenvalues of $T_{I,I} - A_I$ and $T - A$ are contained in the union of the n disks of the complex plane centered at $T_{ii} - a_i$ with radius $\sum_{j=1, j\neq i}^{n} |T_{ij}|$. This in turn implies that $\text{Re}\lambda(T_{I,I} - A_I)$ and $\text{Re}\lambda(T - A)$ are positive. From Theorem 1, we see that (12) is satisfied. □

Corollary 3: Suppose that β is an asymptotically stable equilibrium point and $\alpha = sat(\beta)$ is a memory vector of system (1) *with parameters* $\{A, T, I\}$. *Then, α and β will also be a pair of memory vector and asymptotically stable equilibrium point of system* (1) *with parameters* $\{kA, kT, kI\}$ *for every real number* $k > 0$.

Proof: The proof can easily be established by considering (7)–(9) and Theorem 1. □

Remark 5: The significance of Corollary 3 is that for a given neural network (1), we can increase its speed of evolution by multiplying $A, T,$ and I by a constant $k > 1$, without changing any of its asymptotically stable equilibrium points and any of its memory vectors. Since the speed of evolution of (1) depends on the eigenvalues of $T - A$ and $T_{I,I} - A_I$, it is also clear that the larger the k is, the faster the evolution will be.

B. Synthesis

As pointed out earlier, Theorem 1 will serve as the basis for the synthesis procedures developed in the present paper. In particular, we point to the following important fact, which is a consequence of Case III in Theorem 1.

Corollary 4: Suppose $\alpha \in B^n$. *If* $\beta = A^{-1}(T\alpha + I) \in (C(\alpha))^0$, *then β is an asymptotically stable equilibrium point of* (1). □

We are now in a position to address the follwing synthesis problem.

Synthesis Problem: Given m vectors in B^n (desired memory patterns), say $\alpha^1, \ldots, \alpha^m$, how can we properly choose $\{A, T, I\}$ so that the resulting synthesized system (1) has the properties enumerated below?

1) $\alpha^1, \ldots, \alpha^m$ are memory vectors of system (1).
2) The system has no oscillatory solutions.
3) The total number of spurious memory vectors (i.e., memory vectors of (1) contained in $D^n - \{\alpha^1, \ldots, \alpha^m\}$) is as small as possible.

□

Remark 6: In practice, it is usually required that memory patterns be bipolar, i.e., the memory patterns are in B^n. We will not consider the case where desired memory patterns are not in B^n. □

The preceding results allow us to approach the above synthesis problem in the following manner.

Synthesis strategy: Given m vectors $\alpha^1, \ldots, \alpha^m$ in B^n, find $A, T,$ and I such that

1) $A = \text{diag}[a_1, \ldots, a_n]$ with $a_i > 0$.
2) T is symmetric and has repeated eigenvalues equal to $\mu > 0$ and $-\tau < 0$.
3) $A\beta^i = T\alpha^i + I$ and $A\beta^i = \mu\alpha^i$, where $\beta^i \in (C(\alpha^i))^0$ and μ is the positive eigenvalue of T. □

In the following, we give the rationale for the above strategy.

1) From Remark 3, we see that when T is symmetric, system (1) will have no oscillatory solutions.
2) By Corollary 4, $A^{-1}(T\alpha^i + I) = \beta^i \in (C(\alpha^i))^0$, implies that β^i is an asymptotically stable equilibrium point of the synthesized system, and thus α^i is a memory vector.

3) For $\beta^i \in (C(\alpha^i))^0$ and $A\beta^i = \mu\alpha^i, i = 1, \ldots, m, T$ and I are determined by the relations

$$A\beta^i = \mu\alpha^i = T\alpha^i + I, \quad i = 1, \ldots, m. \tag{14}$$

Solutions of (14) for T and I will always exist. To see this, we let $Y = [\alpha^1 - \alpha^m, \ldots, \alpha^{m-1} - \alpha^m]$. We need to solve T from $TY = \mu Y$, and set $I = \mu\alpha^m - T\alpha^m$. Solutions of $TY = \mu Y$ for T will always exist. We will solve for T by using the singular value decomposition method.

We next present a solution to the Synthesis Problem based on the above observations.

Synthesis Procedure 3.1 Suppose we are given m vectors $\alpha^1, \ldots, \alpha^m$ in β^n, which are to be stored as memory vectors for (1). We proceed as follows:

1) Choose vectors $\beta^i \in (C(\alpha^i))^0$ for $i = 1, \ldots, m$, and a diagonal matrix A with positive diagonal elements, such that $A\beta^i = \mu\alpha^i$, where $\mu > 0$, *i.e.*, choose $\beta^i = [\beta_1^i, \ldots, \beta_n^i]^T$ with $\beta_j^i\alpha_j^i > 1$ and $i = 1, \ldots, m$ and $j = 1, \ldots, n, A = \text{diag}[a_1, \ldots, a_n]$ with $a_j > 0$ for $j = 1, \ldots, n$, and $\mu > \max_{1 \le i \le n}\{a_i\}$ such that $a_j\beta_j^i = \mu\alpha_j^i$.

2) Compute the $n \times (m-1)$ matrix:

$$Y = [y^1, \ldots, y^{m-1}] = [\alpha^1 - \alpha^m, \ldots, \alpha^{m-1} - \alpha^m]. \tag{15}$$

3) Perform a singular value decomposition of $Y = U\Sigma V^T$, where U and V are unitary matrices and Σ is a diagonal matrix with the singular values of Y on its diagonal. (This can be accomplished by standard computer routines.) Let $U = [u^1, \ldots, u^n]$ and $p =$ dimension of $\text{Span}(y^1, \ldots, y^{m-1})$. From the properties of singular value decomposition, we know that $p = \text{rank}(Y), \{u^1, \ldots, u^p\}$ is an orthonormal basis of $\text{Span}(y^1, \ldots, y^{m-1})$, and $\{u^1, \ldots, u^n\}$ is an orthonormal basis of R^n

4) Compute

$$T^+ = [T_{ij}^+] = \sum_{i=1}^{p} u^i(u^i)^T,$$

and

$$T^- = [T_{ij}^-] = \sum_{i=p+1}^{n} u^i(u^i)^T.$$

5) Choose a positive value for the parameter τ and compute

$$T_\tau = \mu T^+ - \tau T^- \text{ and } I_\tau = \mu\alpha^m - T_\tau\alpha^m. \tag{16}$$

Then, $\alpha^1, \ldots, \alpha^m$ will be stored as memory vectors in the following system

$$\begin{cases} \dot{x} = -Ax + T_\tau \text{sat}(x) + I_\tau \\ y = \text{sat}(x) \end{cases}. \tag{17}$$

The states β^i corresponding to $\alpha^i, i = 1, \ldots, m$, will be asymptotically stable equilibrium points of system (17). □

Remark 7: If we wish that in the synthesized system (17), the constant vector $I_\tau = 0$, we can modify Synthesis Procedure 3.1 as follows:

a) In step 2, take

$$y = [\alpha^1, \ldots, \alpha^m]. \tag{18}$$

b) In step 5, take $I_\tau = 0$.

Then all conclusions will remain unchanged. In particular, each $-\beta^i$ and $-\alpha^i, i = 1, \ldots, m$, will also be asymptotically stable equilibrium points and memory vectors, respectively, of the synthesized system (17). □

Remark 8: The synthesis procedure presented above is an adaptation of the *eigenstructure method* developed in [26], [38] to the neural network model described by (1). Using the eigenstructure method, one can guarantee that *any* set of given memory patterns in B^n be stored as memory vectors. However, it is usually required that $p < n$, where $p = \text{rank}(Y), Y$ is defined in (15) or (18), and n is the order of the system. This follows, since if $p = n, T_\tau$ in (16) will become a diagonal matrix with all diagonal elements equal to μ, and I_τ in (16) becomes a zero vector, in which case all vectors in B^n (all corners of the hypercube D^n) will be stored as memory vectors. Simulation results show that when p is very close to $n - 1$, there will be many spurious memory locations in the synthesized system. Experimental studies which compare the eigenstructure method (implemented on various types of artificial neural networks) with other methods have been conducted in several previous works (see, e.g., [24]–[26], [34], [40], [41], [47]). These works indicate that the capacity of neural network paradigms which make use of the eigenstructure method compare rather well with other paradigms (which make use, e.g., of the outer product method, pseudo-inverse techniques, and the like). □

Remark 9: Following the same procedure as in [26], [38], we can prove that all vectors in $L_\alpha \cap B^n$, where $L_\alpha = \text{Aspan}\ (\alpha^1, \ldots, \alpha^m)$, including $\alpha^1, \ldots, \alpha^m$, will be stored as memory vectors in system (17). □

IV. Synthesis Procedure for Sparsely Interconnected Neural Networks

The synthesis techniques developed in the previous section will result in neural networks with *symmetric* and *nonsparse* coefficient matrix T. Implementations of artificial neural networks with *perfectly* symmetric interconnections are not practical. Furthermore, it has been argued by some workers [2], [4], [5], [14], [18], [33], [35], [36], [43], [46] that symmetric interconnections in artificial neural networks are not necessarily always desirable. Moreover, fully interconnected artifical neural networks with even a moderate number of neurons will give rise to large numbers of *line-crossings* resulting from the network interconnections, and thus pose formidable obstacles in VLSI implementations. For these reasons, it is desirable to establish a synthesis procedure which will result in an interconnecting structure which does not require symmetry and which does not demand large numbers of connections.

Using the results of the previous section, we develop in the following a design procedure for artificial neural networks which will result in few line-crossings or no line-crossings at all in the interconnections, and which does not require that the interconnection matrix be symmetric. *Cellular neural networks*, which we address in the next section (with applications to associative memories), are special cases of such *sparsely interconnected artificial neural networks.*

We begin by introducing some necessary terminology.

A matrix $S = [S_{ij}] \in R^{n\times n}$ is said to be an *index matrix* if it satisfies $S_{ij} = 1$ or 0. The restriction of matrix $W = [W_{ij}] \in R^{n\times n}$ to an index matrix S, denoted by $W|S$ is defined by $W|S = [h_{ij}]$, where

$$h_{ij} = \begin{cases} W_{ij}, & \text{if } S_{ij} = 1 \\ 0, & \text{otherwise} \end{cases}.$$

We will say that (1) is a *neural network with a sparse coefficient matrix* if $T = T|S$ for some given index matrix S.

Sparse Design Problem Given an $n \times n$ index matrix $S = [S_{ij}]$ with $S_{ii} \neq 0$ for $i = 1,\ldots,n$, and m vectors $\alpha^1,\ldots,\alpha^m$ in B^n, choose $\{A, T, I\}$ with $T = T|S$ in such a manner that $\alpha^1,\ldots,\alpha^m$ are memory vectors of system (1). □

Remark 10: In the literature, an $n \times n$ matrix is said to be *sparse*, if its number of nonzero elements $\ll n^2$. The definition of *sparse coefficient matrix* given in the present paper is more general and includes the usual definition of sparse matrix as a special case. The sparse design problem considered herein, is in fact, more appropriately called an *indexed design problem.* We will use the term *sparse design* in the present paper since we wish to be able to make comparisons with the fully connected case. □

A solutions for the above sparse design problem is as follows.

Spare Design Procedure 4.1: Suppose we are given an $n\times n$ index matrix $S = [S_{ij}]$ with $S_{ii} \neq 0$ for $i = 1,\ldots,n$ and m vectors $\alpha^1,\ldots,\alpha^m$ in B^n which are to be stored as memory vectors for (1). We proceed as follows:

1) Choose matrix A as the identity matrix.
2) Choose a real number $\mu > 1$ and m vectors $\beta^1,\ldots,\beta^m$, such that $\beta^i = \mu\alpha^i$.
3) Compute the $n\times(m-1)$ matrices $Y = [y^1,\ldots,y^{m-1}] = [\alpha^1 - \alpha^m,\ldots,\alpha^{m-1} - \alpha^m]$, and $Z = [z^1,\ldots,z^{m-1}] = [\beta^1 - \beta^m,\ldots,\beta^{m-1} - \beta^m]$. We let $y^i = [y_1^i,\ldots,y_n^i]^T$ and $z^i = [z_1^i,\ldots,z_n^i]^T$ for $i = 1,\ldots,m-1$.
4) Denote the i^{th} row of the index matrix S by $S_i = [S_{i1},\ldots,S_{in}]$. For each $i = 1,\ldots,n$ construct two sets M_i and N_i, such that $M_i \cup N_i = \{1,\ldots,n\}, M_i \cap N_i = \phi$, and $S_{ij} = 1$ if $j \in M_i, S_{ij} = 0$ if $j \in N_i$. Let $M_i = \{\sigma_i(1),\ldots,\sigma_i(m_i)\}$, where $m_i = \sum_{j=1}^n S_{ij}$ and $\sigma_i(k) < \sigma_i(l)$ if $1 \le k < l \le m_i$. (Note that m_i is the number of nonzero elements in the i^{th} row of matrix S.)
5) For $i = 1,\ldots,n$, and $l = 1,\ldots,m-1$ let $y_{Ii}^l = [y_{\sigma(1)}^l,\ldots,y_{\sigma(m_i)}^l]^T$.
6) For $i = 1,\ldots,n$, compute the $m_i \times (m-1)$ matrices $Y_i = [y_{Ii}^1,\ldots,y_{Ii}^{m-1}]$, and the $1 \times (m-1)$ vectors $Z_i = [z_i^1,\ldots,z_i^{m-1}]$.
7) For $i = 1,\ldots,n$, perform singular values decompositions of Y_i, and obtain
$$Y_i = [U_{i1} \vdots U_{i2}] \begin{bmatrix} D_i & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & 0 \end{bmatrix} \begin{bmatrix} V_{i1}^T \\ \cdots \\ V_{i2}^T \end{bmatrix},$$
where $D_i \in R^{p_i\times p_i}$, is a diagonal matrix with the nonzero singular values of Y_i on its diagonal and $p_i = \text{rank}(Y_i)$.
8) Compute for $i = 1,\ldots,n$, $G_i = [G_{i1},\cdots,G_{im_i}] = Z_iV_{i1}D_i^{-1}U_{i1}^T + W_iU_{i2}^T$ where W_i is an arbitrary $1 \times (m_i - p_i)$ real vector.
9) The matrix $T = [T_{ij}]$ is computed as follows:
$$T_{ij} = \begin{cases} 0, & \text{if } S_{ij} = 0 \\ G_{ik}, & \text{if } S_{ij} \neq 0 \text{ and if } j = \sigma_i(k) \end{cases}. \quad (19)$$
10) The bias vector $I = [I_1,\ldots,I_n]^T$ is computed by $I_i = \beta_i^m - T_i\alpha^m$ for $i = 1,\ldots,n$, where T_i is the i^{th} row of T.

Then $\alpha^1,\ldots,\alpha^m$ will be stored as memory vectors for system (1) with $A, T,$ and I determined as above. The states β^i corresponding to $\alpha^i, i = 1,\ldots,m$, will be asymptotically stable equilibrium points of the synthesized system. □

Remark 11: If in the above synthesis procedure, we choose $A = \text{diag}[a_1,\ldots,a_n]$ with $a_i > 0$, we need to change Z_i in step 6 to $Z_i = [a_iz_i^1,\ldots,a_iz_i^{m-1}]$ and I_i in step 10 to $I_i = a_i\beta_i^m - T_i\alpha^m$. □

Remark 12: If in the index matrix S there are $q > 1$ identical rows, we can design the corresponding q rows of matrix T simultaneously. For such cases, we need to alter slightly steps 5–9 above. We will demonstrate this idea by means of an example in Section VI. We will also demonstrate in Section VI that by special choices of the index matrix S, the Sparse Design Procedure 4.1 can result in a network with few line-crossings, or with no line-crossings at all. □

Our next result addresses the existence of a solution for the sparse design problem and the validity of the above design procedure.

Theorem 2

1) *Solutions for the sparse design problem always exist if* $S_{ii} = 1$ *for* $i = 1,\ldots,n$.
2) *The Sparse Design Procedure* 4.1 *guarantees that* $T = T|S$.
3) *The Sparse Design Procedure* 4.1 *guarantees that all vectors in* $L_\alpha \cap B^n$, *including* $\alpha^1,\ldots,\alpha^m$, *are stored as memory vectors of system* (1), *where* $L_\alpha = Aspan(\alpha^1,\ldots,\alpha^m)$.
4) *The Sparse Design Procedure* 4.1 *can be applied to any set of desired memory patterns* $\alpha^1,\ldots,\alpha^m \in B^n$.

Proof: In order for the synthesized system to be a solution of the sparse design problem, we need $G_iY_i = Z_i$ in steps 7 and 8 of the sparse design procedure. Thus, G_i in step 8 is a solution for the sparse design procedure *if and only if*

$$\text{rank}[Y_i] = \text{rank}\begin{bmatrix} Y_i \\ \cdots \\ Z_i \end{bmatrix}.$$

This condition is satisfied if $S_{ii} = 1, i = 1, \ldots, n$, since under these conditons, Z_i becomes a row vector which is one of the rows in Y_i multiplied by μ. (This argument is also true if we choose $A = \text{diag}[a_1, \ldots, a_n]$ with $a_i > 0$.) This proves part 1 of the theorem.

Part 2 is clear from (19).

To prove part 3, we first check the equilibrium conditions for $\alpha^1, \ldots, \alpha^m$, in which case we require that $T\alpha^l + I = A\beta^l = \beta^l$, for $l = 1, \ldots, m$, where $\beta^l = \mu\alpha^l$ and $\mu > 1$ (therefore $\beta^l \in (C(\alpha^l))^0$). Using the notation given in the design procedure, we write for $l = 1, \ldots, m-1, y_{Ii}^l = Y_i e_l$, where $e_l \in R^{m-1}$ is a column vector with all elements zero except the lth element which is 1 (cf. step 6 of the sparse design procedure). Also, we have for $i = 1, \ldots, n$,

$$U_{i2}^T y_{Ii}^l = 0 \text{ for } l = 1, \ldots, m-1,$$

and

$$Z_i V_{i1} D_i^{-1} U_{i1}^T Y_i = Z_i.$$

The former is clear from the properties of the singular value decomposition. To see the latter, we recall that $Y_i = U_{i1} D_i V_{i1}^T$ and we assume that Z_i is the jth row of Y_i multiplied by μ (without loss of generality). Then, $Z_i = \mu \cdot R_j V_{i1}^T$, where R_j is the jth row of $U_{i1} \times D_i$, and from the properties of the singular value decomposition, we have

$$\begin{aligned} Z_i V_{i1} D_i^{-1} U_{i1}^T Y_i &= \mu \cdot R_j V_{i1}^T \cdot V_{i1} D_i^{-1} U_{i1}^T \cdot U_{i1} D_i V_{i1}^T \\ &= \mu \cdot R_j V_{i1}^T = Z_i, \end{aligned}$$

since $U_{i1}^T U_{i1} = V_{i1}^T V_{i1} = D_i^{-1} D_i = p_i \times p_i$ identity matrix (where p_i is defined in step 7 of the design procedure). According to the design procedure, we compute for $i = 1, \ldots, n$,

$$T_i y^l = G_i y_{Ii}^l,$$

and

$$\begin{aligned} T_i\alpha^l + I_i &= T_i y^l + T_i\alpha^m + I_i = G_i y_{Ii}^l + T_i\alpha^m + \beta_i^m - T_i\alpha^m \\ &= Z_i V_{i1} D_i^{-1} U_{i1}^T Y_i e_l + W_i U_{i2}^T y_{Ii}^l + \beta_i^m = Z_i e_l + \beta_i^m \\ &= z_i^l + \beta_i^m = \beta_i^l - \beta_i^m + \beta_i^m = \beta_i^l. \end{aligned}$$

Hence, $T\alpha^l + I = \beta^l$ for $l = 1, \ldots, m-1$. For $l = m, T\alpha^m + I = \beta^m$ is clear from step 10. Next, we note that the above results imply that

$$T(\alpha^l - \alpha^m) = \beta^l - \beta^m \text{ for } l = 1, \ldots, m-1,$$

and that for vector $\alpha \in L_\alpha \cap B^n$, there is a $\lambda = [\lambda_1, \ldots, \lambda_{m-1}] \in R^{m-1}$ such that

$$\alpha = \lambda_1(\alpha^1 - \alpha^m) + \cdots + \lambda_{m-1}(\alpha^{m-1} - \alpha^m) + \alpha^m.$$

Thus, for every vectory $\alpha \in L_\alpha \cap B^n$, we have

$$\begin{aligned} T\alpha + I &= T[\lambda_1(\alpha^1 - \alpha^m) + \cdots + \lambda_{m-1}(\alpha^{m-1} - \alpha^m)] + T\alpha^m + I \\ &= \lambda_1(\beta^1 - \beta^m) + \cdots + \lambda_{m-1}(\beta^{m-1} - \beta^m) + \beta^m \\ &= \mu[\lambda_1(\alpha^1 - \alpha^m) + \cdots + \lambda_{m-1}(\alpha^{m-1} - \alpha^m) + \alpha^m] \\ &= \mu\alpha \triangleq \beta. \end{aligned}$$

Clearly, $\beta \in (C(\alpha))^0$ since $\mu > 1$. By Theorem 1 (Corollary 4), we see that the states β^i corresponding to $\alpha^i, i = 1, \ldots, m$, as well as the states which correspond to the output vectors in $L_\alpha \cap B^n$ other than α^i, will be asymptotically stable equilibrium points of system (1). Therefore, all vectors in $L_\alpha \cap B^n$, including $\alpha^1, \ldots, \alpha^m$, will be stored as memory vectors for system (1).

Part 4 follows from similar arguments as Remark 8. □

Remark 13: It is emphasized that part 1 of Theorem 2 does not imply or require that $S_{ij}, i \neq j$, be zero. □

Remark 14: If we wish that the above procedure result in a system of form (1) with $I = 0$, we can modify the Sparse Design Procedure 4.1 as follows:

a) In step 3, let $Y = [\alpha^1, \ldots, \alpha^m]$ and $Z = [\beta^1, \ldots, \beta^m]$.
b) In step 10, let $I = 0$.

Then all conclusions will remain unchanged. In particular, all vectors in $\text{Span}(\alpha^1, \ldots, \alpha^m) \cap B^n$ including $\pm\alpha^1, \ldots, \pm\alpha^m$ are stored as memory vectors of system (1). □

V. Applications to Cellular Neural Network Design

Cellular neural networks, introduced in [9], have found several successful applications in image processing and pattern recognition (see, for example, [10], [29]–[32], [49]). On the other hand, applications of (discrete-time) cellular neural networks to associative memories, utilizing the outer product method (the Hebbian rule), seem to have been somewhat less successful [49]. As is well known, the outer product method does not guarantee that every desired memory pattern be stored as an equilibrium point (memory point) of the synthesized system when the desired patterns are not mutually orthogonal. Moreover, the storage capacity of networks designed by the outer product method is known to be exceptionally low.

In the present section, we employ the sparse synthesis techniques developed in the previous section in the design of cellular neural networks with applications to associative memories.

A special class of *two-dimensional cellular neural networks* is described by ordinary differential equations of the form (see [9])

$$\begin{cases} \dot{x}_{ij} = -a_{ij}x_{ij} + \displaystyle\sum_{C(k,l) \in N_r(i,j)} T_{ij,kl} \, \text{sat}(x_{kl}) + I_{ij} \\ y_{ij} = \text{sat}(x_{ij}) \end{cases} \tag{20}$$

where $1 \leq i \leq M, 1 \leq j \leq N, a_{ij} > 0$, and x_{ij} and y_{ij} are the states and the outputs of the network, respectively.

The basic unit in a cellular neural network is called a *cell*. In (20), there are $M \times N$ such cells arranged in an $M \times N$ array. The cell in the ith row and the jth column is denoted by $C(i, j)$, and an *r-neighborhood* $N_r(i, j)$ of the cell $C(i, j)$ for a positive integer r is defined by $N_r(i, j) \triangleq \{C(k, l) : \max\{|k - i|, |l - j|\} \leq r, 1 \leq k \leq M, 1 \leq l \leq N\}$.

Remark 15: System (20) characterizes a special class of continuous-time cellular neural networks with square grids, piecewise linear processors, and memoryless interactions (see, e.g., [6]). □

Remark 16: In the original cellular neural network model introduced by Chua and Yang [9],

$$a_{ij} = \frac{1}{R_x C}$$

(R_x and C are constants), $I_{ij} = I_c$ (I_c is a constant), and $T_{ij,kl} = T_{kl,ij}$. We will not make this assumption in (20). In the applications of cellular neural networks to image processing and pattern recognition, in addition to the bias terms $I_{ij} = I_c$, one frequently requires nonzero input terms

$$\sum_{C(k,l)\in N_r(i,j)} B_{ij,kl}u_{kl}$$

in the first equation of (20), where u_{ij} represents the inputs of the network. In the present application, we consider zero inputs ($u_{ij} \equiv 0$ for all i and j) and a constant *bias vector* $I = [I_{11}, I_{12}, \ldots, I_{MN}]^T$. This renders (20) equivalent to a *nonsymmetric* cellular neural network model by setting the inputs in the original model equal to constants (cf. [5] and [9]). Under these circumstances, we will refer to (20) as a (zero-input) *nonsymmetric cellular neural network*. □

Using the above nomenclature, we choose a matrix $Q = [Q_{ij,kl}] \in R^{MN\times MN}$ as

$$Q_{ij,kl} = \begin{cases} 1, & \text{if } C(k,l) \in N_r(i,j) \\ 0, & \text{otherwise} \end{cases}. \tag{21}$$

We let $S = Q = [S_{ij}] \in R^{n\times n}$, where $n = M \times N$. With this notation, we see that in order for (1) to be equivalent to the nonsymmetric cellular neural network model (20), we require in (1) $T = T|S$, where $S = Q$ and Q is defined in (21). Thus, the cellular neural network model (20) is a special case of the neural network model (1) where the n neurons are arranged in an $M \times N$ array (if $n = M \times N$) and the interconnection structure is confined to local neighborhoods of radius r. Hence, Theorem 1 can be applied to analyze the cellular neural network (20). Clearly $S_{ii} = 1$ ($Q_{ij,ij} = 1$), since $C(i,j) \in N_r(i,j)$ for any positive integer r. Thus, the Sparse Design Procedure 4.1 and its modified version (cf. Remark 14) can be applied to the design of the cellular neural network (20) based on the index matrix $S = Q$, where Q is determined in (21). By Theorem 2, for any given integer $r > 0$ and any set of m vectors $\alpha^1, \ldots, \alpha^m$ in B^{MN}, we can always design a cellular neural network (20) so that (20) will store $\alpha^1, \ldots, \alpha^m$ as memory vectors.

VI. Examples

To demonstrate the applicability of the analysis and synthesis procedures presented in the preceding sections, we consider several specific examples.

Example 1: We use the Sparse Design Procedure 4.1 to synthesize a cellular neural networ (20) with $n = 12$ ($M = 4, N = 3$), and $r = 1$ (neighborhood radius). Given are $m = 4$ vectors specified by

$$\alpha^1 = [1,-1,1,1,-1,1,1,1,1,-1,-1,1]^T,$$
$$\alpha^2 = [1,1,1,1,-1,-1,1,-1,-1,1,1,1]^T,$$
$$\alpha^3 = [1,-1,1,1,-1,1,1,-1,1,1,1,1]^T,$$

and

$$\alpha^4 = [1,-1,-1,1,-1,-1,1,-1,-1,1,1,1]^T.$$

It is desired that these vectors be stored as memory vectors of system (20).

We determine the index matrix $S = Q = [S_{ij}] \in R^{12\times 12}$, where $Q = [Q_{ij,k_\ell}] \in R^{(4\times 3)\times(4\times 3)}$ is defined in (21). Using the Sparse Design Procedure 4.1, we determined A as the 12×12 identity matrix,

$$T = \left[\begin{matrix}
-1.0000e+01 & 0.0000e+00 & 0 & -1.0000e+01 \\
-1.0000e+01 & -2.0000e+00 & 4.0000e+00 & -1.0000e+01 \\
0 & -2.6667e+00 & 4.6667e+00 & 0 \\
-1.0000e+01 & 0.0000e+00 & 0 & -1.0000e+01 \\
-1.0000e+01 & -4.0000e+00 & 4.0000e+00 & -1.0000e+01 \\
0 & -4.4000e+00 & 4.4000e+00 & 0 \\
0 & 0 & 0 & -1.0000e+01 \\
0 & 0 & 0 & -1.0000e+01 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{matrix}\right.$$

$$\begin{matrix}
-1.0000e+01 & 0 & 0 & 0 \\
-1.0000e+01 & -4.0000e+00 & 0 & 0 \\
-1.0000e+01 & -2.6667e+00 & 0 & 0 \\
-1.0000e+01 & 0 & -1.0000e+01 & -0.0000e+00 \\
-1.0000e+01 & -2.0000e+00 & -1.0000e+01 & -1.0888e-14 \\
-1.0000e+01 & -1.2000e+00 & 0 & -2.1826e-15 \\
-1.0000e+01 & 0 & -1.0000e+01 & -1.3333e+01 \\
-1.0000e+01 & -4.4409e-16 & -1.0000e+01 & -1.2667e+01 \\
-1.0000e+01 & -1.0000e+00 & 0 & -1.0000e+01 \\
0 & 0 & -1.0000e+01 & -1.4000e+01 \\
0 & 0 & -1.0000e+01 & -1.4000e+01 \\
0 & 0 & 0 & -1.0000e+14
\end{matrix}$$

$$\left.\begin{matrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
-2.0000e+00 & 0 & 0 & 0 \\
-1.2000e+00 & 0 & 0 & 0 \\
0 & -6.6667e+00 & -6.6667e+00 & 0 \\
-1.3323e-15 & -7.3333e+00 & -7.3333e+00 & -1.0000e+01 \\
1.0000e+00 & 0 & -1.0000e+01 & -1.0000e+01 \\
0 & -6.0000e+00 & -6.0000e+00 & 0 \\
-2.2538e-15 & -6.0000e+00 & -6.0000e+00 & -1.0000e+01 \\
-7.8505e-16 & 0 & -1.0000e+01 & -1.0000e+01
\end{matrix}\right],$$

and

$$\begin{aligned} I = [&1.2e+01, 6.0e+00, -1.2667e+01, 2.2e+01, \\ &1.4e+01, -1.44e+01, 1.2e+01, 2.0e+01, \\ &-8.8818e-15, 1.0e+01, 2.0e+01, 1.2e+01]^T. \end{aligned}$$

(In the above computations, we chose $\mu = 2$ in step 2 and $W_i = -10 \times O_{m_i} \times U_{i2}$ in step 8 of the Sparse Design Procedure 4.1, where $O_{m_i} = [1, \ldots, 1] \in R^{1\times m_i}$ and $m_i = \sum_{j=1}^{n} S_{ij}$).

Using Theorem 1, we verified that $\alpha^1, \ldots, \alpha^4$ are memory vectors of the synthesized system (20) with $\{A, T, I\}$

TABLE I

	The cellular neural network with $r = 1$	The fully connected neural network
average of total number of memory vectors in B^n	12.2	4.5
average of total number of undesired memory vectors in B^n	8.2	0.5
average of total number of memory vectors in $D^n - B^n$	3.1	0.4
average of total number of unstable equilibrium points in R^n	64.65	14.25
total number of desired patterns which were not stored as memory vectors	0	0

given above. Simulation results, using (2), also verified that $\alpha^1, \ldots, \alpha^4$ are reachable memory vectors of (20).

By Theorem 1, we determined that system (20) has 2 additional memory vectors in B^n given by

$$\alpha^5 = [1, -1, -1, 1, -1, -1, 1, 1, -1, -1, -1, 1]^T,$$

and

$$\alpha^6 = [1, 1, 1, 1, -1, -1, 1, 1, -1, -1, -1, 1]^T.$$

Their corresponding asymptotically stable states are given by $\beta^5 = 2\alpha^5$ and $\beta^6 = 2\alpha^6$, respectively. System (20) has 1 additional memory vector in $D^n - B^n$, given by

$$\alpha^7 = [-0.7273, -1, -1, 1, 1, -1, -0.7273, \\ -1, -1, 1, 1, 1]^T.$$

Its corresponding asymptotically stable state is given by

$$\beta^7 = [-0.7273, -4.7273, -22, 16.5455, 12.5455, -22, \\ -0.7273, -4.7273, -22, 19.2727, 19.2727, 2]^T.$$

System (20) also has five unstable equilibrium points.

Example 2: In order to ascertain how typical the results of Example 1 are, we repeated the example twenty times using different sets of desired vectors to be stored as memory vectors. Each set contained $m = 4$ vectors in B^n which were generated randomly. For each given set of vectors, we synthesized system (20) using the Sparse Design Procedure 4.1. Table I summarizes our findings. Also shown in Table I are the results for system (1) synthesized by Synthesis Procedure 3.1 *for the same sets of desired vectors* to be stored as memories.

From Table I we see that in the present example, the cellular neural network implementations for associative memories have more spurious states than the fully connected networks. Also, there are more unstable equilibrium points in the synthesized cellular neural networks than in the fully connected networks in the present example.

Example 3: We now present several problems which to the best of our knowledge cannot be addressed by other synthesis procedures for associative memories. In all cases, we consider a neural network with 16 neurons ($n = 16$) and in all cases our objective is to store the four patterns shown in Figure 1 as memories. As indicated in this figure, sixteen boxes are used to represent each pattern (in R^{16}), with each box corresponding

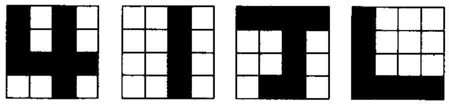


Fig. 1. The four desired memory patterns used in Example 3.

Fig. 2. Grey levels.

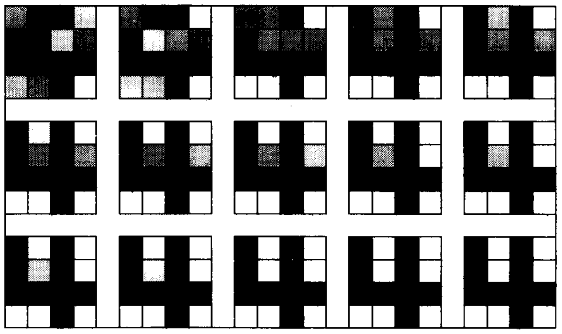


Fig. 3. A typical evolution of pattern number 1 of Figure 1.

to a vector component which is allowed to assume values between −1 and 1. For purpose of visualization, −1 will represent white, 1 will represent black, and the intermediate values will correspond to appropriate grey levels, as shown in Figure 2.

The four cases which we consider below, were synthesized by the Sparse Design Procedure 4.1. These cases involve different prespecified constraints on the interconnecting structure of each network.

Case I Cellular Neural Network. We designed a cellular neural network with $r = 1, M = 4$, and $N = 4$. (Due to space limitations, we will not display the interconnecting matrix T for the present case, as well as for the three subsequent cases.) The performance of this network is illustrated by means of a typical simulation run of (1) (or (20)), shown in Figure 3. In this figure, the desired memory pattern is depicted in the lower right corner. The initial state, shown in the upper left corner, is generated by adding to the desired pattern zero-mean Gaussian noise with a standard deviation SD = 1. The iteration of the simulation evolves from left to right in each row and from the top row to the bottom row. The desired pattern is recovered in 14 steps with a step size $h = 0.2$ in the simulation of (1). We do not identify a unit for the step size h. In view of Remark 5, the unit could be seconds, milliseconds, or any other small time intervals. All simulations for the present paper were performed on a Sun SPARC Station using MATLAB.

Case II Reduction of Line-Crossings. We arranged the 16 neurons in a 4×4 array and we considered only horizontal and vertical interconnections. For this case, the index matrix $S = Q = [S_{ij}] \in R^{16\times16}$, and $Q = [Q_{ij,kl}] \in R^{(4\times4)\times(4\times4)}$ assumes the form

$$Q_{ij,kl}\begin{cases}1, & \text{if } i = k \text{ or } j = l \\ 0, & \text{otherwise}\end{cases}. \tag{22}$$

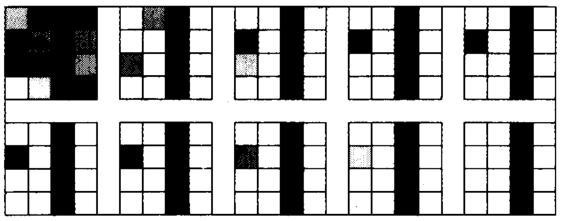


Fig. 4. A typical evolution of pattern number 2 of Figure 1.

A typical simulation run for the present case is depicted in Figure 4. In this figure, the noisy pattern is generated by adding to the desired pattern uniformly distributed noise defined on $[-0.7, 0.7]$. Convergence occurred in 9 steps with $h = 0.2$.

We emphasize that by choosing the index matrix as in (22), we were able to reduce significantly the number of line-crossings, which is of great concern in VLSI implementations of artificial neural networks.

Case III Two Rows of S Identical (see Remark 12). In this case, we chose an index matrix $S = [S_{ij}] \in R^{16\times16}$ of the form

$$S_{ij} = \begin{cases} 1, & \text{if } i = 1 \text{ or } i = 16 \text{ or } j = 1 \text{ or } \\ & j = 16 \text{ or } |i-j| \le 1. \\ 0, & \text{otherwise} \end{cases}$$

This requires that the T matrix has zero elements everywhere except in its first and last rows, its first and last columns, and in its tridiagonal elements.

Rows 2 to 15 are designed by using the Sparse Design Procedure 4.1, step by step. Since the first row S_1 and the last row S_{16} of S are identical, we can design the rows T_1 and T_{16} of T simultaneously. To see this, we take in step 5 of the design procedure $y_{I1}^l = [y_{\sigma(1)}^l, \dots, y_{\sigma(m_1)}^l]^T$ and $y_{I16}^l = [y_{\sigma(1)}^l, \dots, y_{\sigma(m_{16})}^l]^T$ for $l = 1, \dots, m-1$. Clearly, $m_1 = m_{16} = n = 16$ and $y_{I1}^l = y_{I16}^l$ for $l = 1, \dots, m-1$, since $S_1 = S_{16} = [1, \dots, 1] \in R^{1\times16}$. In step 6, we take $Y_1 = [y_{I1}^1, \dots, y_{I1}^{m-1}]$ and the $2 \times (m-1)$ vector

$$Z_1 = \begin{bmatrix} z_1^1 & & z_1^{m-1} \\ & \cdots & \\ z_{16}^1 & & z_{16}^{m-1} \end{bmatrix}.$$

In step 7, we perform a singular value decomposition of Y_1 and obtain U_{11}, U_{12}, D_1, and V_{11}. In step 8, we compute $G_1 = Z_1 V_{11} D_1^{-1} U_{11}^T + W_1 U_{12}^T$, where W_1 is an arbitrary $2 \times (m_1 - p_1)$ real matrix and $p_1 = \text{rank}(Y_1)$. In step 9, we determine T_1 from the first row of G_1 and T_{16} from the second row of G_1 using (19).

A typical simulation run for this network is shown in Figure 5. In this case, the noisy pattern was generated by adding Gaussian noise $N(0, 0.5)$ to the desired pattern. Convergence occured in 24 steps with $h = 0.2$.

We can generalize the above case to design problems for which the index matrix S has several identical rows. In particular, if $S = [S_{ij}], S_{ij} = 1$ for all i and j, the Sparse Design Procedure 4.1 reduces to a procedure for a *fully connected* neural network (1), and the reduced design procedure (where all rows of T are determined simultaneously) will be more general than Synthesis Procedure 3.1. To see this, note that the reduced design procedure will generally result in a nonsymmetric T and by special choice of matrix W in step 8, the reduced design procedure will become Synthesis Procedure 3.1.

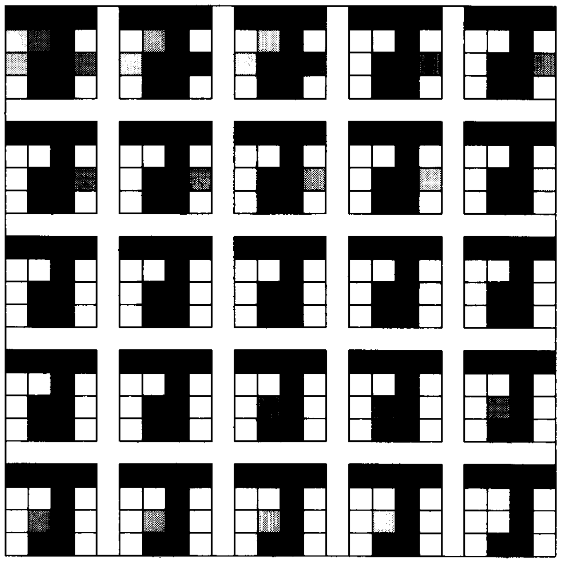


Fig. 5. A typical evolution of pattern number 3 of Figure 1.

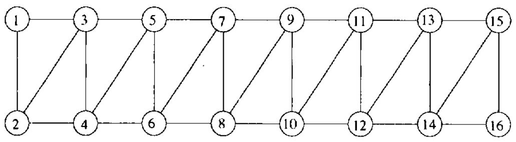


Fig. 6. A possible structure for a neural network without line-crossings in the interconnecting structure.

Case IV Quinquediagonal Matrix S Resulting in an Interconnecting Structure Without Line-Crossings. We chose $S = [S_{ij}] \in R^{16\times16}$ as

$$S_{ij} = \begin{cases} 1, & \text{if } |i-j| \le 2 \\ 0, & \text{otherwise} \end{cases}. \tag{23}$$

This will result in a quinquediagonal matrix S, enabling us to arrange the $n = 16$ neurons in the configuration shown in Figure 6. Note that in this figure there are no line-crossings. Furthermore, note that this configuration can be generalized to arbitrary n.

A typical simulation run for the present case is depicted in Figure 7. In this figure, the noisy pattern was generated by adding Gaussian noise $N(0.1, 0.7)$ to the desired pattern. Convergence occurred in 13 steps with $h = 0.2$.

We note that for the above example, many other interesting design cases can be addressed in a *systematic* manner, including a neural network (1) with lower or upper triangular matrix T, combinations of Cases I, II, III, and IV given above, and so forth.

Example 4: We consider the 25 desired memory patterns $\alpha^1, \dots, \alpha^{25}$ shown in Figure 8. (The upper left pattern is denoted by α^1 and the lower right pattern is denoted by α^{25}).

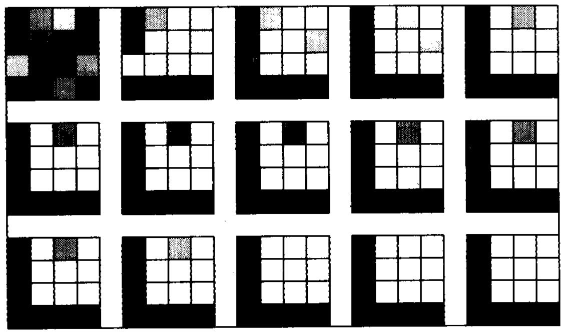


Fig. 7. A typical evolution of pattern number 4 of Figure 1.

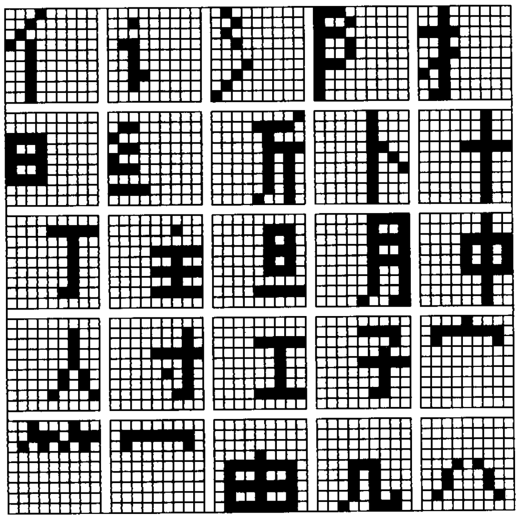


Fig. 8. The twenty-five desired memory patterns for Example 4.

These patterns constitute modules which represent individually or in combination, Chinese characters. They are coded in R^{81} in a similar manner as was done in Example 3 (including usage of grey levels, as described in Example 3).

We wish to synthesize a cellular neural network (20) with $n = 81(M = N = 9)$ and $r = 3$, which will "remember" these modules. As mentioned above, some of the Chinese characters can be represented by two modules. In particular, the patterns given in Figure 8 can be used to generate at least 50 commonly used Chinese characters. To demonstrate this, we add one more vector, α^{26}, with every entry equal to 1 (black), to the set of desired memory patterns. In doing so, we can generate desired combinations for Chinese characters which are made up of some of the basic modules given in Figure 8. For instance, the character corresponding to α^6 means "sun" and the character corresponding to α^{14} means "moon". A new Chinese character can be generated as $\alpha^{27} = \alpha^6 + \alpha^{14} + \alpha^{26} \in \text{Span}(\alpha^1, \ldots, \alpha^{26}) \cap B^{81}$, which means "bright" (see Figure 9). Using the modified Sparse Design Procedure 4.1 as discussed in Remark 14, we only need to synthesize a system (1), in which the 81 neurons are arranged in a 9×9 array and the interconnections are restricted to local neighborhoods of radius $r = 3$ by employing these basic

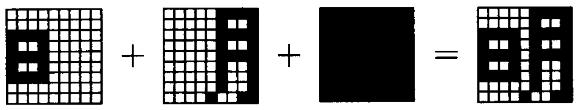


Fig. 9. The Chinese character composed of patterns number 6 and number 14 in Figure 8.

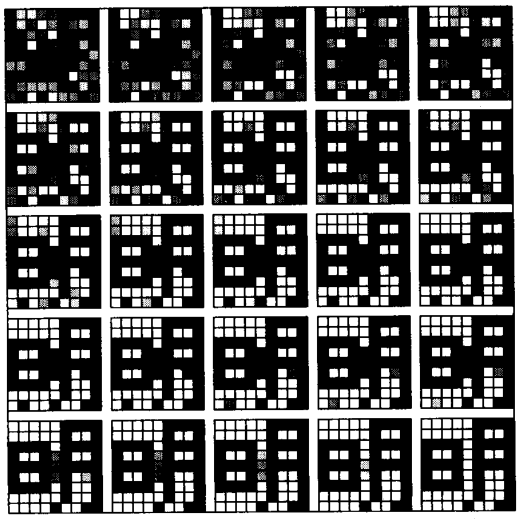


Fig. 10. A typical evolution of the Chinese character composed of patterns number 6 and number 14 in Figure 8.

patterns. The resulting system will automatically "remember" all possible combinations of these basic components, which include the 50 commonly used Chinese characters mentioned above (by Theorem 2 and Remark 14).

In the modified Sparse Design Procedure 4.1, by taking $\mu = 2$ in step 2 and $W_i = -10 \times O_{m_i} \times U_{i2}$ in step 8, where $O_{m_i} = [1, \ldots, 1] \in R^{1 \times m_i}$ and $m_i = \sum_{j=1}^{n} S_{ij}$, we designed a neural network of the form (1) with the above specifications (i.e., a cellular neural network of the form (20) with $M = N = 9$ and $r = 3$) which stores $\alpha^1, \ldots, \alpha^{26}$ as memory vectors. This system has 2601 total interconnections, while a fully connected neural network with $n = 81$ will have total of 6561 interconnections. By using the cellular neural network of the present example, we were able to reduce the total number of required interconnections to less than 40%.

A typical simulation run, involving the pattern $\alpha^{27} = \alpha^6 + \alpha^{14} + \alpha^{26}$ is depicted in Figure 10. The noisy initial pattern in Figure 10 (upper left hand corner) is generated by adding to α^{27} zero-mean Gaussian noise with a standard deviation SD = 1. The desired pattern α^{27} is recovered in 24 steps with a step size $h = 0.227$ (lower right hand corner in Figure 10).

Simulation results showed that all the other vectors corresponding to the aforementioned 50 commonly used Chinese characters are (reachable) memory vectors of the synthesized cellular neural network.

For the same initial noisy pattern shown in Figure 10, the desired pattern is recovered in 8 steps, with the same step size, when using a fully connected neural network (1) designed by the modified Synthesis Procedure 3.1 (as discussed in Remark 7) for the same desired set of memory patterns $\alpha^1, \ldots, \alpha^{26}$. One of the reasons for the lower convergence speed of cellular neural networks is that we only use local interconnections in such systems.

The most commonly used Chinese characters are roughly 6700 in number. More than half of these consist of approximately 500 basic characters (modules) or combinations of these characters, as described in Example 4. This example can be expanded by designing a cellular neural network which will store these 500 basic Chinese characters as well as combinations of these characters. In doing so, we will have stored over one half of the 6700 commonly used Chinese characters. The remaining commonly used characters (numbering about 3000), will have to be stored separately, using the design procedure described above.

VII. Conclusion

In the present paper we considered a class of artificial neural networks which have the basic structure of analog Hopfield neural networks [20] and which use the (piecewise linear) saturation funciton to model the neurons (see Section I and system (1)). This model is closely related to the cellular neural networks considered in [9] and to the neural networks defined on hypercubes in [26].

For system (1), we first conducted a qualitative analysis which enables us to locate all of the equilibria and determine their stability properties (see Section III-A and Theorem 1). Next, we developed for system (1) a synthesis procedure for associative memories which *guarantees* to store desired patterns in B^n as memories (see Section III-B). The rationale for this procedure is based on Theorem 1. It yields neural networks with symmetric interconnecting structure with no other constraints on the structure (such as sparsity). This procedure constitutes an adaptation of the *eigenstructure method* [26] to system (1).

By utilizing the result described above, we developed in Section IV a synthesis procedure (for associative memories) for sparsely interconnected neural networks (Sparse Design Procedure 4.1). This procedure results in neural networks which satisfy a *prespecified* interconnecting structure. In Section V, we applied this design procedure in the synthesis of a class of cellular neural networks for associative memories. Finally, in Section VI, we demonstrated the applicability and the versatility of the results established herein by means of four specific examples.

The significance of the results presented in this paper is that we can synthesize by the present methods neural networks which have a *prespecified interconnecting structure* and which will *guarantee to store any desired set of memory patterns in* B^n as memories provided that the interconnecting structure includes self feedback for all neurons. Design procedures which result in neural networks with prespecified interconnecting structure without self feedback are currently under investigation.

References

[1] S. V. B. Aiyer *et al.*, "A theoretical investigation into the performance of the Hopfield Model," *IEEE Trans. on Neural Networks*, vol. 1, pp. 204–215, June 1990.

[2] G. Avitabile *et al.*, "On a class of nonsymmetrical neural networks with application to ADC," *IEEE Trans. on Circuit Syst.*, vol. 38, pp. 202–209, Feb. 1991.

[3] J. Bruck, "On the convergence properties of the Hopfield Model," *Proc. of the IEEE*, vol. 78, Oct. 1990, pp. 1579–1585.

[4] G. A. Carpenter *et al.*, "Computing with neural networks," *Science*, vol. 235, pp. 1226–1227, 1987.

[5] L. O. Chua and T. Roska, "Stability of a class of nonreciprocal cellular neural networks," *IEEE Trans. on Circuit Syst.*, vol. 37, pp. 1520–1527, Dec. 1990.

[6] L. O. Chua and T. Roska, "The CNN paradigm," *IEEE Trans. on Circuit Syst.-I*, vol. 40, pp. 147–156, Mar. 1993.

[7] L. O. Chua and P. Thiran, "An analytic method for designing simple cellular neural networks," *IEEE Trans. on Circuit Syst.*, vol. 38, pp. 1332–1341, Nov. 1991.

[8] L. O. Chua and C. W. Wu, "On the universe of stable cellular neural networks," *Int. J. Circuit Theory and App.*, vol. 20, pp. 497–572, 1992.

[9] L. O. Chua and L. Yang, "Cellular neural networks: theory," *IEEE Trans. on Circuit Syst.*, vol. 35, pp. 1257–1272, Oct. 1988.

[10] L. O. Chua, L. Yang, "Cellular neural networks: applications," *IEEE Trans. on Circuit Syst.*, vol. 35, pp. 1273–1290, Oct. 1988.

[11] M. A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Trans. on Systems, Man, and Cybernet.*, vol. 13, pp. 815–826, Sep./Oct. 1983.

[12] M. Cottrell, "Stability and attractivity in associative memory networks," *Biol. Cybern.*, vol. 58, pp. 129–139, 1988.

[13] S. R. Das, "On the synthesis of nonlinear continuous neural networks," *IEEE Trans. on Systems, Man, and Cybernet.*, vol. 21, pp. 413–418, March/Apr. 1991.

[14] J. A. Farrell, A. N. Michel, "A synthesis procedure for Hopfield's continuous-time associative memory," *IEEE Trans. on Circuit Syst.*, vol. 37, pp. 877–884, July 1990.

[15] R. M. Golden, "The 'brain-state-in-a-box' neural model is a gradient descent algorithm," *J. of Math. Psych.*, vol. 30, pp. 73–80, March 1986.

[16] H. J. Greenberg, "Equilibria of the brain-state-in-a-box (BSB) neural model," *Neural Net.*, vol. 1, pp. 323–324, 1988.

[17] L. T. Grujić and A. N. Michel, "Exponential stability and trajectory bounds of neural networks under structure variations," *IEEE Trans. on Circuit Syst.*, vol. 38, pp. 1182–1192, Oct. 1991.

[18] A. Guez *et al.*, "On the stability, storage capacity, and design of nonlinear continuous neural networks," *IEEE Trans. on Systems, Man, and Cybernet.*, vol. 18, pp. 80–87, Jan./Feb. 1988.

[19] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, pp. 2554–2558, Apr. 1982.

[20] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci. USA*, vol. 81, pp. 3088–3092, May 1984.

[21] S. Hui and S. H. Żak, "Dynamical analysis of the brain-state-in-a-box (BSB) neural models," *IEEE Trans. on Neural Net.*, vol. 3, pp. 86–94, Jan. 1992.

[22] J. D. Keeler, "Basins of attraction of neural network models," *AIP Conf. Proc. 151*, Snowbird, UT, pp. 259–264, 1986.

[23] J. Levendovszky, "A possible transformation of fully connected neural nets into partially connected networks," *Proc. 1990 IEEE Int. Workshop on Cellular Neural Networks and App.*, Budapest, Hungary, Dec. 1990, pp. 55–64.

[24] J.-H. Li *et al.*, "Qualitative analysis and synthesis of a class of neural networks," *IEEE Trans. on Circuit Syst.*, vol. 35, pp. 976–986, Aug. 1988.

[25] J.-H. Li *et al.*, "Analysis and synthesis of a class of neural networks: variable structure systems with infinite gain," *IEEE Trans. on Circuit Syst.*, vol. 36, pp. 713–731, May 1989.

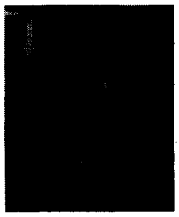
[26] J.-H. Li *et al.*, "Analysis and synthesis of a class of a neural networks: linear systems operating on closed hypercube," *IEEE Trans. on Circuit Syst.*, vol. 36, pp. 1405–1422, Nov. 1989.

[27] M. Marcus and H. Minc, *A Survey of Matrix Theory and Matrix Ineq.*, Boston, MA: Allyn and Bacon, 1964.

[28] C. M. Marcus and R. M. Westervelt, "Dynamics of Iterated-Map Neural Networks," *Phy. Rev. A* vol. 40, pp. 501–504, July 1989.

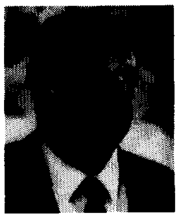
[29] T. Matsumoto, L. O. Chua and R. Furukawa, "CNN Cloning Template: Hole-Filler," *IEEE Trans. on Circuit Syst.*, vol. 37, pp. 635–638, May 1990.

[30] T. Matsumoto *et al.*, "CNN cloning template: connected component detector," *IEEE Trans on Circuit Syst.*, vol. 37, pp. 633–635, May 1990.
[31] T. Matsumoto *et al.*, "CNN Cloning Template: Shadow Detector," *IEEE Trans. on Circuit Syst.*, vol. 37, pp. 1070–1073, Aug. 1990.
[32] T. Matsumoto *et al.*, "Image thinning with a cellular neural network," *IEEE Trans. on Circuit Syst.*, vol. 37, pp. 638–640, May 1990.
[33] K. Matsuoka, "Stability conditions for nonlinear continuous neural networks with asymmetric connection weights," *Neural Net.*, vol. 5, pp. 495–500, 1992.
[34] A. N. Michel and J. A. Farrell, "Associative memories via artificial neural networks," *IEEE Control Syst.* , vol. 10, pp. 6–17, Apr. 1990.
[35] A. N. Michel *et al.*, "Qualitative analysis of neural networks," *IEEE Trans. on Circuit Syst.*, vol. 36, pp. 229–243, Feb. 1989.
[36] A. N. Michel *et al.*, "Analysis and synthesis techniques for Hopfield type synchronous discrete time neural networks with application to associative memory," *IEEE Trans. on Circuit Syst.*, vol. 37, pp. 1356–1366, Nov. 1990.
[37] A. N. Michel and D. L. Gray, "Analysis and synthesis of neural networks with lower block triangular interconnecting structure," *IEEE Trans. on Circuit Syst.*, vol. 37, pp. 1267–1283, Oct. 1990.
[38] A. N. Michel *et al.*, "Analysis and synthesis of a class of discrete-time neural networks described on hypercubes," *IEEE Trans. on Neural Net.*, vol. 2, pp. 32–46, Jan. 1991.
[39] R. Perfetti, "A neural network to design neural networks," *IEEE Trans. on Circuit Syst.*, vol. 38, pp. 1099–1103, Sept. 1991.
[40] L. Personnaz *et al.*, "Information storage and retrieval in spin-glass like neural networks," *J. Physique Lett.*, vol. 46, pp. L359–365, Apr. 1985.
[41] L. Personnaz *et al.*, "Collective computational properties of neural networks: new learning mechanisms," *Phys. Rev. A*, vol. 34, pp. 4217–4228, Nov. 1986.
[42] T. Roska and L. O. Chua, "Cellular neural networks with nonliner and delay-type template elements," *Int. J. Circuit Theory and App.*, vol. 20, pp. 469–481, 1992.
[43] F. M. A. Salam *et al.*, "On the analysis of dynamic feedback neural nets," *IEEE Trans. on Circuit Syst.*, vol. 38, pp. 196–201, Feb. 1991.
[44] M. E. Savran and Ö. Morgül, "On the associative memory design for the Hopfield neural network," *Proc. of 1991 IEEE Int. Joint Conf. on Neural Net.*, Singapore, pp. 1166–1171, Nov. 1991.
[45] F. R. Waugh *et al.*, "Fixed-point attractors in analog neural computation," *Phys. Rev. Lett.*, vol. 64, pp. 1986–1989, Apr. 1990.
[46] G. Yen and A. N. Michel, "A learning and forgetting algorithm in associative memories: results involving pseudo-inverses," *IEEE Trans. on Circuit Syst.*, vol. 38, pp. 1193–1205, Oct. 1991.
[47] G. Yen and A. N. Michel, "A learning and forgetting algorithm in associative memories: the eigenstructure method," *IEEE Trans. on Circuit Syst.–II*, vol. 39, pp. 212–225, Apr. 1992.
[48] F. Zou and J. A. Nossek, "Stability of cellular neural networks with opposite-sign templates," *IEEE Trans. on Circuit Syst.*, vol. 38, pp. 675–677, June 1991.
[49] *Proc. 1990 IEEE Int. Workshop on Cellular Neural Net. and App.*, Budapest, Hungary, Dec. 1990.



Derong Liu (S'91-M'94) received the B.S. degree in mechanical engineering from the East China Institute of Technology in 1982, and the M.S. Degree in electrical engineering from the Institute of Automation, Chinese Academy of Sciences in 1987. Currently, he is working toward the Ph.D. degree in electrical engineering at the University of Notre Dame, Notre Dame, IN. During his first year of graduate study at Notre Dame (1990–1991), he received the Michael J. Birck Fellowship.

From 1982 to 1984, he worked as an electromechanical engineer at China North Industries Corp., Jilin, China. From 1987 to 1990, he was a faculty member in the Department of Electrical Engineering at the Graduate School of Chinese Academy of Sciences, Beijing, China. Since October 1993, he has been with the Electrical and Electronics Research Department, General Motors NAO R&D Center. His research interests include systems and control theory, signal processing, and neural networks. He is a member of Eta Kappa Nu.



Anthony N. Michel (SM'79-F'82) received the Ph.D. degree in electrical engineering from Marquette University and the D.Sc. degree in applied mathematics from the Technical University of Graz, Austria.

He has seven years of industrial experience. From 1968 to 1984, he was at Iowa State University, Ames, IA. From 1984 to 1988, he was Frank M. Freimann Professor and Chairman of the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN. Currently, he is Frank M. Freimann Professor and Matthew H. McCloskey Dean of the College of Engineering at the University of Notre Dame. He is author and coauthor of three texts and several other publications.

Dr. Michel received the 1978 Best Transactions Paper Award of the IEEE Control Systems Society, the 1984 Guillemin-Cauer Award from the IEEE Circuits and Systems Society, and an IEEE Centennial Medal. He is a former Associate Editor and a former Editor of the IEEE Transactions on Circuits and Systems, a former associate editor of the IEEE Transactions on Automatic Control, and a former Associate Editor of IEEE Transactions on Neural Networks. He was the Program Chairman of the 1985 IEEE Conference on Decision and Control. He was President of the IEEE Circuits and Systems Society in 1989. He was Co-Chairman of the 1990 IEEE International Symposium on Circuits and Systems. He was a Fulbright Scholar in 1992 (at the Technical Unviersity of Vienna, Austria). He is an Associate Editor at Large for the IEEE Transactions on Automatic Control, and he is an elected member of the Board of Governors and the Vice President of the IEEE Control Systems Society. He is a Foreign Member of the Academy of Engineering of the Russian Federation.