Analysis and Synthesis Techniques for Hopfield Type Synchronous Discrete Time Neural Networks with Application to Associative Memory

ANTHONY N. MICHEL, FELLOW, IEEE, JAY A. FARRELL, MEMBER, IEEE, AND HUI-FANG SUN

Abstract —In the present paper we establish a qualitative theory for synchronous discrete time Hopfield-type neural networks. Our objectives are accomplished in two phases. First, we address the analysis of the class of neural networks considered. Next, making use of these results, we develop a synthesis procedure for the class of neural networks considered herein.

In the analysis section, we utilize techniques from the theory of large-scale interconnected dynamical systems to derive tests for the asymptotic stability of an equilibrium of the neural network. We also present estimates for the rate at which the trajectories of the network will converge from an initial condition to a final state. In the synthesis section we utilize the stability tests from the analysis section as constraints to develop a design algorithm for associative memories. The present algorithm guarantees that each desired memory will be stored as an equilibrium and that each desired memory will be asymptotically stable.

The applicability of the present results is demonstrated by means of two specific examples.

I. INTRODUCTION

In [1] and [2], Hopfield presents models for neural networks that are fully interconnected. In addition, Hopfield was able to show that the trajectories of these networks would not oscillate, but would seek local minima of a certain energy function. The main assumption of this analysis is that the interconnection matrix must be symmetric with zero diagonal elements. Using the energy function as the basis for a design procedure (see [19]), neural networks have been utilized to convert analog signals to digital signals, to decompose an additive signal, and to implement associative memories. Several other applications are discussed in [8].

The particular application discussed in this paper is the associative memory. The goal of the associative memory is to store a set of patterns (i.e., x^i, for $i = 1, \cdots, m$) in such a manner that the network will recognize an input pattern (i.e., $x = x^i + \Delta x$) that is near one of the stored vectors (i.e., x^i) as the stored vector. Hopfield showed that a neural network could simulate an associative memory if each x^i could be stored as a local minimum of his energy function.

The recent article [17] (also see [7]) presents both a thorough introduction to Hopfield discrete time neural networks and a detailed stochastic analysis of their memory capacity. The analysis in [17] is applicable to both synchronous and asynchronous systems with symmetric interconnection matrices. Section V of that paper mentions the need for methods to test the stability and attractivity of patterns that we desire for the network to store. Such tests are presented in the analysis section of the present paper. The tests are applicable to networks having either the symmetric interconnection matrices of [17] or nonsymmetric interconnection matrices of the present paper. In addition, the present paper presents methods for estimating the rate of convergence to an equilibrium. Results similar to those presented herein for discrete time systems are presented in [9] and [18] for the continuous time Hopfield model.

The application of neural networks to coding theory has been discussed in [16]. This paper considers a two-layered network, in which the first layer is an Adaline network designed to transform a set of vectors from a "feature space" to a set of vectors in the "code space." The new vectors in the "code space" are chosen to be pseudo-orthogonal. The second layer is also an Adaline network, which has been designed to classify the codes that are output from the first layer. Notice that the network has been designed and is not adaptive. The examples of the present paper are similar to the examples in [16] and may be interpreted in the same manner. However, our examples discuss single layer, feedback, and autoassociative networks. The addition of a preprocessing layer would undoubtedly improve the performance in our examples, because all the patterns to be stored (being outputs of the preprocessor) could be chosen as pseudo-orthogonal vectors. We did not investigate this approach

Manuscript received March [illegible], 19[illegible]. The work of A. N. Michel and H.-F. Sun was supported by the National Science Foundation under Grant ECS [illegible]. The work of J. A. Farrell was supported by a fellowship from the University of Notre Dame Applied Mathematics Center and by the NSF under Grant ECS [illegible]. This paper was recommended by Associate Editor M. Ilic.

A. N. Michel and H.-F. Sun are with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN [illegible].

J. A. Farrell is with the Charles Stark Draper Laboratory, Cambridge, MA.

IEEE Log Number [illegible].

because our main emphasis is on the presentation of general analysis and synthesis procedures that allow networks to be designed with a memory capacity of N, where N is the number of neurons in the network.

The present paper does not follow Hopfield's global energy function approach. Instead, we develop in Section IV a local theory for fully interconnected discrete time neural networks. In our analysis we view the network as an interconnected system of N simple processors. The local theory includes sufficient conditions for the asymptotic stability of an equilibrium, and bounds on the rate of convergence of system trajectories.

In order to correspond closely to biological neural networks, Hopfield's discrete time model switches asynchronously. We consider a model that switches synchronously. By making this assumption, we are able to present deterministic proofs of stability. In addition, the model is suitable for a straightforward implementation via parallel processing computers. Both of these characteristics are important, because we are interested in an engineering application (pattern recognition) for which we desire guaranteed storage and fast responses.

In Section V, using as constraints the stability tests that were determined in Section IV, we develop a method to design associative memories. The procedure guarantees to store each of the n vectors ($n \le N$) the designer wishes to store. The only condition on the vectors is that they must be linearly independent. The design procedure chooses the interconnections and thresholds in a manner to store each desired pattern as an equilibrium of the network, and to satisfy an asymptotic stability constraint. This is in contrast to methods that choose the interconnections and thresholds to store the patterns as energy minima. In Section VI we present two design examples.

The designs by the method presented herein will not result in a symmetric interconnection matrix in most cases. The reader must remember that the symmetry condition is a mathematical assumption that is necessary for the Hopfield energy function approach. Biological networks are not expected to have symmetric interconnections, and networks designed to be symmetric should be expected to lose their symmetry property during the implementation process due to parameter inaccuracies. The present synthesis procedure is based on the properties of M-matrices. These matrices are robust to parameter inaccuracies.

The present design method does not have a proof of global stability (i.e., oscillatory solutions are not proven to be nonexistent) due to the lack of a symmetric interconnection matrix. Instead, we have guaranteed asymptotic stability for the stored patterns. As our second example illustrates, the number of nonconvergent solutions is very small.

II. Neural Network Models

In [1] and [2] Hopfield presents, respectively, models for continuous and discontinuous neural networks. Associated with each of these models is an energy function which the trajectories of the system seek to minimize. The network is considered to have N neurons. The inputs and outputs of the set of N neurons are described in vector notation by $u = [u_1, \cdots, u_N]^T$ and $V = [V_1, \cdots, V_N]^T$, respectively. Here, $u_i: R^+ = [0, \infty) \to R = (-\infty, \infty)$ and $V_i: R^+ \to R$ denote the input and output of the ith neuron, respectively.

The dynamics of the discontinuous model are described by the following algorithm. Each neuron computes its output according to the formula

$$u_i = \sum_{j=1}^{N} T_{ij} V_j + I_i$$

$$V_i = \operatorname{sgn}(u_i) \qquad \text{(DM)}$$

where T_{ij}, the strength of connection between neuron i and j, belongs to R, and I_i, an external input to the ith neuron, also belongs to R. The order in which neurons change states is completely random and asynchronous, but the mean rate of change of the state, W, is fixed. In the above, the function sgn is the signum function defined as

$$\operatorname{sgn}(x) = \begin{cases} 1, & x > 0 \\ -1, & x < 0. \end{cases}$$

In the case where $u = 0$, V_i retains its previous value.

The equation that describes the dynamics of Hopfield's continuous model can be simplified to

$$\dot{u}_i = \sum_{j=1}^{N} T_{ij} V_j - b_i u_i + I_i$$

$$V_i = g_i(u_i) \qquad \text{(CM)}$$

where $\dot{u}_i = du_i/dt$, $T_{ij} \in R$, $b_i \in R$, and $I_i \in R$.

Remark 1: The function $g_i: R \to R$ is a monotone nondecreasing, continuously differentiable function with $g_i(0) = 0$. Thus $g_i(x)$ satisfies a sector condition: i.e., there are two positive real numbers c_1 and c_2 such that $c_1 \le g_i(x)/x \le c_2$ for all $x \in B(r_i) = \{y \in R: |y| < r_i\}$ for some $r_i > 0$. In the above, $|\cdot|$ denotes absolute value.

The design of associative memories (AM's) for the discontinuous model has been considered in several papers (see [3]–[6]). The method of [5] is also discussed in [1], where the outer product method is shown to be capable of storing about $0.15N$ patterns. A difficulty in designing these networks is due to the discontinuous input–output relationships. Often, it is possible to store the desired patterns as stable states of (DM); however, the discrete nature of the model's dynamical equation makes it difficult to guarantee that a pattern (stored as an equilibrium point) will attract neighboring states. This model is difficult to implement in hardware due to the requirement of a large number of interconnections and the inherent capacitances and delays. This model is often implemented on a digital computer where the requirement of random asynchronous switching times is approximated by a pseuor random switching sequence. Synchronous models have recently been discussed (e.g., in [7] and [19]).

ANM's can also be implemented by Hopfield's continuous time model, as discussed in [2], [9]–[11], [22], and [23]. The design method of [1] is applicable in the high gain limit where (CM) approaches (DM). This technique does not guarantee to store each of the desired patterns (as asymptotically stable equilibrium points). The methods discussed in [9]–[11], [22], and [23] guarantee to store each desired pattern. The methods are compared in [10]. Again, a problem arises with implementing this circuit in hardware because of inherent parasitic dynamics, the parameter inaccuracies, and the amount of space required to implement a large number of resistors. For a study of implementation questions see [12]. For some interesting implementations that attempt to avoid parasitics, refer to [24], [25].

Most applications to date have been implemented by special purpose software. Either (CM) was simulated by numerical routines on a digital computer or (DM) was implemented in a synchronous or pseudosynchronous manner. The present paper seeks to avoid the simulation problems of both models, the design problems of the discontinuous problem, and the implementation problem of the continuous model by considering a discrete time synchronous model that approximates (CM). We analyze this model directly by the method of [13]. In addition to these analysis results, we present in Section V a synthesis technique for the synchronous discrete time model.

By Euler's method, (CM) can be approximated in discrete time by replacing

$$\dot{u}(t) \approx \frac{u((k+1)T) - u(kT)}{T} \qquad \text{for } kT \le t < (k+1)T. \tag{2.1}$$

In the above equation, T is the sampling period, and as $T \to 0$ the accuracy of the approximation increases. Substitution of (2.1) into (CM) yields

$$\frac{u_i((k+1)T) - u_i(kT)}{T} = \sum_j T_{ij}V_j(kT) - b_i u_i(kT) + I_i.$$

Hence,

$$\begin{aligned} u_i(k+1) &= \sum_j T_{ij}V_j(k) + (1 - Tb_i)u_i(k) + I_i \\ &= \sum_j T_{ij}V_j(k) + a_i u_i(k) + I_i \end{aligned} \tag{DT}$$

where $a_i = 1 - Tb_i$, $u_i(k) = (1/T)u_i(kT)$, and $V_i(k) = V_i(kT)$. The equations describing (DT) and (DM) are very similar. For both systems, the input to each neuron is a weighted sum of the outputs of all the neurons. The features that distinguish model (DT) include its operating in a synchronous mode and its use of a sigmoidal input–output function. These features give model (DT) two advantages over (DM): first, (DT) allows direct implementation in software or special purpose hardware, and second, it allows design techniques that guarantee both stability and attractivity to stored patterns. The dynamics of the discrete time, synchronous model described by (DT) are the subject of the remainder of this paper.

III. Large-Scale Systems Approach

To circumvent the difficulties inherent in the direct analysis of the system (DT) as a whole, we find it convenient to incorporate the method of analysis proposed in [13]. By this approach we will view the system (DT) as the interconnection of free or isolated subsystems. This section will be divided into two parts. The first part transforms (DT) into the form necessary for the second part. The second part explicitly defines the system in a manner consistent with the large-scale systems approach.

3.1. The Translation of the Equilibrium

Defining $T = [T_{ij}] \in R^{N \times N}$, $A = \text{diag}(a_i) \triangleq \text{diag}(a_1, \cdots) \in R^{N \times N}$, $I = [I_i] \in R^N$, and $g = [g_1, \cdots, g_N]^T: R^N \to R^N$, then the equation (DT) in matrix form becomes

$$\begin{aligned} u(k+1) &= TV(k) + Au(k) + I \\ V(k) &= g(u(k)). \end{aligned} \tag{DT}$$

In general, the network will have equilibria that may not be located at the origin ($u_i = 0$, $i = 1, 2, \cdots, N$). The equilibria of (DT) may be desired memories that we have stored, or may be extraneous equilibria. Each equilibrium must, by definition, satisfy the equation

$$u(k) = TV(k) + Au(k) + I.$$

This equation is equivalent to

$$0 = TV(k) + Bu(k) + I \tag{3.1}$$

where the matrix $B = A - E$ and E is an N dimensional identity matrix. We are able to translate any equilibrium of (DT) into the origin in the following manner. Let

$$\left.\begin{aligned} p(k) &= u(k) - u^* \\ G(p(k)) &= g(u(k)) - g(u^*) \end{aligned}\right\} \tag{3.2}$$

where u^* satisfies (3.1) and $G = [G_1, \cdots, G_N]^T$. Then

$$\begin{aligned} p(k+1) + u^* &= Tg(p(k) + u^*) + A(p(k) + u^*) + I \\ &= T[G(p(k)) + g(u^*)] + Ap(k) + Au^* + I \\ &= TG(p(k)) + Ap(k) + Tg(u^*) + Au^* + I. \end{aligned}$$

Thus

$$\begin{aligned} p(k+1) &= TG(p(k)) + Ap(k) + Tg(u^*) + Bu^* + I \\ &= TG(p(k)) + Ap(k) \end{aligned} \tag{Σ}$$

which has an equilibrium at $p(k) = 0$. Henceforth, we will use (Σ) to represent the model of the neural network under study. We will assume that all equilibria of (Σ) are isolated. In the next section, we will address the stability properties of (arbitrary) given equilibrium points. In our approach *we will assume without loss of generality that a given equilibrium under investigation is located at the origin.*

Remark 2: Due to the relationship between $G_i(x)$ and $g_i(\cdot)$, $G_i(\cdot)$ is also a monotonically nondecreasing, continuously differentiable function with $G_i(0) = 0$. Again, $G_i(0)$ will satisfy a sector condition $c_{i1} \le G_i(x)/x \le c_{i2}$ for all $x \in B(r_i)$ with some $r_i > 0$ for $i = 1, \cdots, N$. Here, c_{i1} and c_{i2} are two positive real numbers that may not be the same as those in Remark 1.

3.2 Interconnected Systems

We rewrite (Σ) into the form

$$p_i(k+1) = \sum_{j=1}^{N} T_{ij} G_j(p_j(k)) + A_{ii} p_i(k), \qquad \text{for } i = 1, \cdots, N \qquad (\Sigma)$$

System (Σ) may be viewed as an interconnection of N free (isolated) subsystems represented by the equations

$$x_i(k+1) = T_{ii} G_i(x_i(k)) + A_{ii} x_i(k), \qquad \text{for } i = 1, \cdots, N \qquad (S_i)$$

with the interconnecting structure defined by

$$h_i(x_1, \cdots, x_N) \triangleq \sum_{\substack{j=1 \\ j \neq i}}^{N} T_{ij} G_j(x_j(k)), \qquad \text{for } i = 1, \cdots, N \qquad (3.2)$$

Following the method of analysis in [13], we will establish stability results for system (Σ) that are phrased in terms of the qualitative properties of the free subsystems (S_i) and in terms of the properties of the interconnecting structure given in (3.2).

IV. Stability Analysis

The stability results of the entire neural network (Σ) are determined by the stability results of the individual free neurons (S_i) and the properties of the interconnection structure (3.2) of the neural network. In this section, we first analyze the stability of the free subsystems. Next we establish stability results for the entire neural network (Σ) that are phrased in terms of the stability properties of the individual free subsystems (S_i) and in the terms of the properties of the interconnecting structure given by (3.2).

4.1 Stability Results of the Free Subsystem

To analyze the stability of a free subsystem (S_i), we need the following assumption.

Assumption (A-1): For (S_i),

$$0 < \sigma_i \triangleq (|A_{ii}| + |T_{ii}| c_{i2}) < 1$$

where c_{i1} and c_{i2} are defined in Remark 2.

Proposition 1: If (A-1) is true, then the equilibrium $x_i = 0$ of (S_i) is asymptotically stable.

Proof: Choose $v_i(x_i(k)) = |x_i(k)|$ as a Lyapunov function for (S_i). Then the first forward difference of v_i along the solution of (S_i) is given by

$$\begin{aligned}
\nabla v_{i(S_i)}(x_i(k)) &= v_i(x_i(k+1)) - v_i(x_i(k)) \\
&= |A_{ii} x_i(k) + T_{ii} G_i(x_i(k))| - |x_i(k)| \\
&\leq (|A_{ii}| - 1)|x_i(k)| + |T_{ii} G_i(x_i(k))| \\
&\leq [|A_{ii}| - 1 + |T_{ii}| c_{i2}] |x_i(k)|.
\end{aligned}$$

Clearly, v_i is positive definite and ∇v_i is negative definite if (A-1) is true. Therefore, the equilibrium $x_i = 0$ is asymptotically stable (cf. [14]). ■

4.2 Stability Results of the Neural Network

Assumption (A-2): Given σ_i of (A-1), the successive principal minors of matrix $D = [D_{ij}]$ are all positive, with

$$D_{ij} = \begin{cases} -(\sigma_i - 1), & i = j \\ -\sigma_{ij}, & i \neq j \end{cases}$$

where $\sigma_{ij} = |T_{ij}| c_{j2}$ (c_{j2} is defined in (A-1)).

Theorem 1: If (A-1) and (A-2) are true, then the equilibrium $p = 0$ of (Σ) is asymptotically stable.

Proof: We choose a Lyapunov function for (Σ):

$$v(p(k)) = \sum_{i=1}^{N} \lambda_i |p_i(k)|$$

where $\lambda_i > 0$ for $i = 1, \cdots, N$. Then the first forward difference of v along the solutions of (Σ) is given by

$$\begin{aligned}
&\nabla v_{(\Sigma)}(p(k)) \\
&= v(k+1) - v(k) \\
&= \sum_{i=1}^{N} \lambda_i \{|p_i(k+1)| - |p_i(k)|\} \\
&= \sum_{i=1}^{N} \lambda_i \left\{ \left| A_{ii} p_i(k) + \sum_{j=1}^{N} T_{ij} G_j(p_j(k)) \right| - |p_i(k)| \right\} \\
&\leq \sum_{i=1}^{N} \lambda_i \left\{ |A_{ii}| |p_i(k)| + \sum_{j=1}^{N} |T_{ij} G_j(p_j(k))| - |p_i(k)| \right\} \\
&\leq \sum_{i=1}^{N} \lambda_i \left\{ [|A_{ii}| - 1] |p_i(k)| + \sum_{j=1}^{N} |T_{ij}| |G_j(p_j(k))| \right\} \\
&\leq \sum_{i=1}^{N} \lambda_i \left\{ [|A_{ii}| - 1] |p_i(k)| + \sum_{j=1}^{N} |T_{ij}| c_{j2} |p_j(k)| \right\} \\
&= \sum_{i=1}^{N} \lambda_i [|A_{ii}| - 1 + |T_{ii}| c_{i2}] |p_i(k)| \\
&\quad + \sum_{i=1}^{N} \lambda_i \sum_{\substack{j=1 \\ j \neq i}}^{N} |T_{ij}| c_{j2} |p_j(k)| \\
&= \sum_{i=1}^{N} \lambda_i (\sigma_i - 1) |p_i(k)| + \sum_{i=1}^{N} \lambda_i \sum_{\substack{j=1 \\ j \neq i}}^{N} \sigma_{ij} |p_j(k)| \\
&= -\lambda^T D w
\end{aligned}$$

where $\lambda = (\lambda_1, \cdots, \lambda_N)^T$ and $w = (|p_1|, \cdots, |p_N|)^T$. Since by (A-2) $D_{ij} \leq 0$ when $i \neq j$, and the successive principal minors of matrix D are all positive, D is an M-matrix. By the properties of M-matrices, D^{-1} exists and each element of D^{-1} is nonnegative. Hence there exists a vector $y = (y_1, \cdots, y_N)^T$, with $y_i > 0$ for $i = 1, \cdots, N$, such that (see [13])

$$y^T w < 0, \qquad \text{where } y^T = -\lambda^T D$$

$$\lambda = (D^{-1})^T y > 0.$$

Therefore, ∇v is negative definite, v is positive definite, and the equilibrium $p = 0$ of (Σ) is asymptotically stable (cf. [14]). ■

The Lyapunov function in the proof of Theorem 1 is a generalized Hamming distance. Thus if for an equilibrium x^* the translation (T) is applied and the assumptions of

Theorem 1 are found to be true, then Theorem 1 shows that for initial conditions sufficiently near x^*, the generalized Hamming distance from x^* to solutions of (DT) will asymptotically approach zero. The weighting vector λ is included to increase the applicability and to decrease the conservatism of Theorem 1. One method for choosing λ is shown in the proof of Theorem 1. Since the value of λ is not unique, [20] and [21] consider methods for choosing λ in an optimal fashion.

Assumption (A-3): Given ϵ_1 and ϵ_2 in Remark 2, we define δ_i:

$$\delta_i = \begin{cases} \dfrac{1}{\epsilon_1}, & \text{if } |A_{ii}| \ge 1 \\[2mm] \dfrac{1}{\epsilon_2}, & \text{if } |A_{ii}| < 1. \end{cases}$$

Assumption (A-4): The successive principal minors of the $N \times N$ test matrix D are all positive where

$$D_{ij} = \begin{cases} \delta_i(|A_{ii}| - 1) - T_{ii}, & i = j \\ -|T_{ij}|, & i \ne j. \end{cases}$$

Similarly as in Theorem 1, we now obtain the following result.

Corollary 1: The equilibrium $p = 0$ of (Σ) is asymptotically stable if (A-1), (A-3), and (A-4) are satisfied.

Proof: We choose a Lyapunov function for (Σ) as

$$v(p(k)) = \sum_{i=1}^{N} \lambda_i |p_i(k)|, \qquad \text{with } \lambda_i > 0.$$

Then

$$\begin{aligned}
\Delta v_{(\Sigma)}(p(k)) &= v(k+1) - v(k) \\
&= \sum_{i=1}^{N} \lambda_i \big(|p_i(k+1)| - |p_i(k)|\big) \\
&\le \sum_{i=1}^{N} \lambda_i \Big\{ (|A_{ii}| - 1)|p_i(k)| + \sum_{j=1}^{N} |T_{ij}||G_j(p_j(k))| \Big\} \\
&\le \sum_{i=1}^{N} \lambda_i \Big\{ (|A_{ii}| - 1)\frac{|p_i(k)|}{|G_i(p_i(k))|}|G_i(p_i(k))| \\
&\qquad + \sum_{j=1}^{N} |T_{ij}||G_j(p_j(k))| \Big\} \\
&\le \sum_{i=1}^{N} \lambda_i \Big\{ \big[\delta_i(|A_{ii}| - 1) + T_{ii}\big]|G_i(p_i(k))| \\
&\qquad + \sum_{\substack{j=1 \\ j \ne i}}^{N} |T_{ij}||G_j(p_j(k))| \Big\} \\
&= -\lambda^T D w \\
&< 0, \text{ by proper choice of } \lambda
\end{aligned}$$

where $\lambda = (\lambda_1, \cdots, \lambda_N)^T$ and $w = (|G_1(p_1)|, \cdots, |G_N(p_N)|)^T$. Therefore, the equilibrium is asymptotically stable. ∎

4.3. Estimates of Trajectory Bounds

The previous section presented methods for determining the stability properties of the various equilibria of (Σ). Usually we also desire information about network performance. One critical performance issue concerns the network's rate of convergence from an initial condition to the final state. The present section develops trajectory bound estimates that allow the designer to predict the rate of convergence near the equilibria of the network.

In the case where (A-2) is satisfied, D is an M-matrix. Properties of M-matrices are discussed in [13]. The particular property of M-matrices that will be used in the sequel is given in (A-5).

Assumption (A-5): For the matrix $D = [D_{ij}]$ (as defined in (A-2)), there exist constants $\lambda_i > 0$, $i = 1, \cdots, N$, such

$$D_{jj} + \sum_{\substack{i=1 \\ i \ne j}}^{N} \frac{\lambda_i}{\lambda_j} D_{ij} \ge \epsilon > 0, \qquad j = 1, \cdots, N.$$

The condition for D to be an M-matrix (to satisfy (A-2)) only requires (A-5) to be satisfied with $\epsilon = 0$. Thus (A-5) is slightly more stringent than (A-2). Using (A-5), we can prove the following.

Theorem 2: If (A-5) is satisfied, then

$$\|p(k)\| \le (1-\epsilon)^k \|p(0)\|$$

for all $\|p\| < r = \min(\epsilon_1, \epsilon_2)$, where $\|p\| = \sum_{i=1}^{N} \lambda_i |p_i|$, ϵ_i is defined in Remark 2, and λ_i are given by (A-5).

Proof: Choose $v(p(k)) = \sum_{i=1}^{N} \lambda_i |p_i|$. Then

$$\begin{aligned}
\Delta v_{(\Sigma)}(p(k)) &= v(p(k+1)) - v(p(k)) \\
&= \sum_{i=1}^{N} \lambda_i \big(|p_i(k+1)| - |p_i(k)|\big) \\
&= \sum_{i=1}^{N} \lambda_i \Big[\Big| A_{ii} p_i(k) - \sum_{j=1}^{N} T_{ij} G_j(p_j(k)) \Big| - |p_i(k)| \Big] \\
&\le \sum_{i=1}^{N} \lambda_i \Big[|A_{ii}||p_i(k)| - |p_i(k)| + \sum_{j=1}^{N} |T_{ij} k_j||p_j(k)| \Big] \\
&= -\sum_{i=1}^{N} \lambda_i \Big[D_{ii}|p_i(k)| + \sum_{\substack{j=1 \\ j \ne i}}^{N} D_{ij}|p_j(k)| \Big] \\
&= -\sum_{j=1}^{N} \sum_{i=1}^{N} \lambda_i D_{ij} |p_j(k)| \\
&= -\sum_{j=1}^{N} \lambda_j \Big[\sum_{i=1}^{N} \frac{\lambda_i}{\lambda_j} D_{ij} \Big] |p_j(k)| \\
&\le -\epsilon \sum_{j=1}^{N} \lambda_j |p_j(k)| = -\epsilon v(p(k))
\end{aligned}$$

of u^*,

$$\Delta_{(4)} v(p(k)) \le -\varepsilon v(p(k)) \qquad (4.1)$$

At this point we invoke the comparison principle (see [14]). Inequality (4.1) gives rise to the comparison equation

$$r_{k+1} - r_k = -\varepsilon r_k. \qquad (4.2)$$

The solution of (4.2) is

$$r_k = (1-\varepsilon)^k r_0.$$

It now follows from the comparison principle that

$$v(p(k)) \le \mu^k v(p(0))$$

where $\mu = (1-\varepsilon)$. From (A-5), $0 < \varepsilon < 1$, which implies $1 > \mu > 0$. ■

By use of Theorem 2, we are able to calculate an exponential bound for the rate of convergence from an initial state within the domain of attraction of u^*.

In the translation (T) and the subsequent analysis, we have assumed that the input I is constant. The model can be generalized by the addition of a time variant input $\tilde{I}(k)$. This model is described by

$$u(k+1) = TV(k) + Au(k) + I + \tilde{I}(k). \qquad (DT')$$

With the following assumption, Theorem 2 can be extended to apply to (DT') as is presented in Corollary 2.

Assumption (A-6): Assume that for (DT')

$$\sum_{i=1}^{N} \lambda_i |\tilde{I}_i(k)| \le M, \qquad \text{for all } k \ge 0$$

for some $M > 0$.

Corollary 2: If (A-5) and (A-6) are true, then

$$v(p(k)) \le \left[\alpha - \frac{M}{\varepsilon}\right]\mu^k + \frac{M}{\varepsilon}$$

provided that $\alpha > M/\varepsilon$ and $v(p(0)) \le \alpha$.

The proof of this corollary follows the same approach as the proof of Theorem 2.

V. Synthesis Results

As mentioned in the introduction, neural networks have been shown to effectively implement AM's. This section presents a technique that invokes the stability conditions of the previous sections to synthesize an AM using (DT).

5.1 Equilibrium Constraints

For the purpose of designing AM's, we will assume that a set of *library vectors* is given:

$$\Lambda = \{V^1, V^2, \ldots, V^r\}, \qquad V^i \in R^N.$$

Each element of Λ is a pattern that is to be stored in the AM. In terms of (DT), we will proceed by designing the network to have each element of Λ as an asymptotically stable equilibrium. In this approach, each stored pattern V^i will have an associated domain of attraction D^i:

$$D^i = \{V_0 \in R^N \mid V(k, 0, V_0) \to V^i \text{ as } k \to \infty\}.$$

In order for each $V^i \in \Lambda$ to be an equilibrium of (Σ), (CT) must be satisfied

$$u^i = TV^i + Au^i + I, \qquad i = 1, \ldots, r \qquad (CT)$$

where $V^i = (V_1^i, \ldots, V_N^i)^T$, $u^i = (u_1^i, \ldots, u_N^i)^T$, and $V_j^i = G_j(u_j^i)$. Equation (CT) is equivalent to (2.1).

To compactly present the design algorithm, we define two matrices L and H. Let

$$L = [V^1 : V^2 : \cdots : V^r] \qquad (5.1)$$

$$H = [u^1 : u^2 : \cdots : u^r], \qquad (5.2)$$

where V^i and u^i are the corresponding ith output and input library vectors, respectively. In terms of the matrices defined in (5.1) and (5.2), (CT) can be expressed as

$$H = TL + AH + J$$

where $J = [I : I : \cdots : I]$ is the $N \times r$ matrix with vector I as each of its columns. Letting E represent the $N \times N$ identity matrix, the above equation is equivalent to

$$0 = TL + (A - E)H + J.$$

Our goal is to specify T, A, and I so that the linear constraint (CT) is satisfied, where L and H are given. With this goal in mind, we let

$$R_j = [L^T, H_j^T, Q]$$

$$W_j = [T_{j1}, T_{j2}, \ldots, T_{jN}, A_j, I_j]$$

where H_j is the jth row of H, and $Q \in R^r$ is a column vector containing all ones. Solving (CT) is equivalent to solving

$$H_j^T = R_j W_j^T, \qquad \text{for } j = 1, \ldots, N. \qquad (C1)$$

However, (C1) has a well known solution (making use of pseudo inverse) given by

$$W_j^T = R_j^+ H_j^T \qquad (5.3)$$

where $R_j^+ = R_j^T(R_j R_j^T)^{-1}$. This solution is not, in general, unique. A full discussion of solutions of (C1) can be found in [15].

5.2 Asymptotic Stability Constraints

Constraint (C1) allows us to design a network with each element of Λ as an equilibrium. However, that constraint alone does not guarantee that each element of Λ will be asymptotically stable.

Theorem 1 of the previous section presents a criterion that may be used to guarantee asymptotic stability. Assumption (A-7) below presents a condition equivalent to (A-2). The form of (A-7) is more easily testable than the form of (A-2).

Assumption (A-7): For the matrix $D = [D_{ij}]$ defined in (A-2), assume that there exist constants $\lambda_j > 0$, $j = 1, \ldots, N$, such that

$$\sum_{j=1}^{N} \lambda_j D_{ji} > 0, \qquad \text{for } i = 1, \ldots, N$$

Assumption (A-7) is called a row-dominance condition (see [15]). Explicitly, (A-7) requires

$$\lambda_i(1-|A_{ii}|) - \sum_{j=1}^{N} \lambda_j |T_{ij}| c_{j2} > 0 \qquad \text{(C2')}$$

for $i = 1, \cdots, N$.

For the purpose of this design, we will arbitrarily set $\lambda_i = 1$, for $i = 1, \cdots, N$. If no solutions exist for this choice of λ_i, then other values may be assumed. The optimal choice of λ_i has not been considered, because good results have been attained by the present choice.

Due to the sigmoidal shape of the function g_i and the fact that the desired patterns are usually composed of binary elements, the upper sector bound is usually quite small. The following assumption states this explicitly.

Assumption (A-8): For $j = 1, \cdots, N$, assume that $0 < c_{j2} < 1$.

In the unlikely case when (A-8) is not found to be true, the following analysis can be repeated using the appropriate bound for c_{i2} in place of the value 1.

With the choice of $\lambda_i = 1$ for all $i = 1, \cdots, N$ and (A-8), inequality (C2') can be simplified as follows

$$\begin{aligned}\lambda_i(1-|A_{ii}|) - \sum_{j=1}^{N} \lambda_j |T_{ij}| c_{j2} &= 1-|A_{ii}| - \sum_{j=1}^{N} |T_{ij}| c_{j2}\\ &\geq 1-|A_{ii}| - \sum_{j=1}^{N} |T_{ij}|.\end{aligned}$$

This analysis shows that if inequality (C2),

$$F_i = 1 - |A_{ii}| - \sum_{j=1}^{N} |T_{ij}| > 0, \qquad i = 1, \cdots, N \qquad \text{(C2)}$$

is satisfied, then (C2') will be satisfied. Since (C2') is equivalent to (A-7), satisfaction of constraint (C2) will guarantee asymptotic stability.

5.3. The Synthesis Technique

This section presents a method for designing the desired network. We will consider two cases. In the first case, the gain λ of the function g will be considered as variable. The second case considers λ to be constant. In either case the technique will seek to satisfy both constraints (C1) and (C2) for $i = 1, \cdots, N$.

Remark 3: Subsequently we will assume that the nonlinear function g_i of the individual neurons are all the same. Thus we drop the index from g_i and λ_i.

Due to the form of constraints (C1) and (C2) (both constraints are a function of W_i, which contains one row of T, A, and I), the synthesis procedure enables us to design the network one row at a time resulting in a great simplification. For example, the matrix inversion required in (C1) involves an $(r \times r)$ matrix. To solve the entire set of constraints at once would require the inversion of an $(rN \times rN)$ matrix. The former inversion is much simpler than the latter.

Case I: In this case, the gain of the function g is a variable parameter. To show this explicitly, we will write

$$V_i = g(\lambda u_i).$$

As usual, we will consider the set V to be given. Due to the properties of g discussed in Remark 1, g is invertible. Hence, a simple calculation

$$u_i = \frac{1}{\lambda} g^{-1}(V_i)$$

will yield the U matrix for any L matrix. To further simplify, we set $A_{ii} = 0$, $i = 1, \cdots, N$, in the following procedure. If solutions exist in this special case, then solutions will also exist without the constraint, since the latter case is a generalization of the former.

For the present case, the designer chooses an initial gain λ_0. Using this gain the designer can, in turn, find U^0, R_i^0, and P_i^0. The initial solution to (5.3) for this choice of λ_0 will be denoted W_i^0, for $i = 1, \cdots, N$. In each case, $F_i(W_i^0)$ is easily calculated. Define

$$\epsilon = \min_{1 \le i \le N} (F_i(W_i^0)).$$

If $\epsilon > 0$, the constraints (C1) and (C2) are both satisfied. The desired values of T, A, and I are given by the W_i^0, $i = 1, \cdots, N$.

If $-\epsilon > 0$, choose a constant $\delta > 1 - \epsilon > 1$, and define

$$\begin{aligned}\lambda &= \delta\lambda_0\\ L &= L_0\\ U &= U_0/\delta.\end{aligned}$$

Due to the assumption that $A = 0$, the U_i^1 column may be removed from R_i. In this case, neither P_i nor R_i are affected by the change in λ. Hence, (5.3) shows that decreasing U by the factor δ will decrease each W_i by a factor of δ. Specifically,

$$\begin{aligned}W_i &= [T_{i1}, \cdots, T_{iN}, I_i]\\ &= [T_{i1}^0/\delta, \cdots, T_{iN}^0/\delta, I_i^0/\delta] = W_i^0/\delta.\end{aligned}$$

The following shows that (C2) is now satisfied: for $i = 1, \cdots, N$,

$$F_i^0 = 1 - |A_{ii}^0| - \sum_{j=1}^{N} |T_{ij}^0| = 1 - \sum_{j=1}^{N} |T_{ij}^0| \geq \epsilon$$

$$\sum_{j=1}^{N} |T_{ij}^0| \leq 1 - \epsilon, \quad \epsilon < 0$$

$$\sum_{j=1}^{N} |T_{ij}| = \sum_{j=1}^{N} |T_{ij}^0|/\delta \leq \frac{1-\epsilon}{\delta} < 1$$

$$F_i = 1 - \sum_{j=1}^{N} |T_{ij}| \geq 1 - \left(\frac{1-\epsilon}{\delta}\right) > 0.$$

The above steps are easily translated into a computer algorithm. This algorithm will present the desired T, A, and I matrices under the conditions that λ is variable and that the assumptions required to solve (C1) are satisfied.

In the several specific examples that we considered using this method, the results have been very successful. Two of these examples are presented in the final section of the present paper.

Case 2: In this case, we consider both matrices L and H to be given; thus λ is fixed. As was mentioned in Section II, (DS) is meant to be implemented as a computer algorithm. As such, the designer should be able to choose the gain λ, and Case 1 should usually apply. However, for the sake of completeness we will present the design algorithm for Case 2. The design approach for Case 1 is very straightforward, while the algorithm for Case 2 may require iteration.

Our goal is to present an algorithm that tests for solutions W_i satisfying (C1) and (C2). The algorithm presented below solves for T, A, and I, one row at a time. To complete a design the algorithm must be repeated N times (once for each row).

An initial solution to (C1) is given by (5.5) as

$$W_{i0} = P_i H_i^T$$

where $P_i = B_i^T(B_iB_i^T)^{-1}$ and $W_{i0} = (T_{i1}, T_{i2}, \cdots, T_{iN}, A_{ii}, I_i)$. For this initial solution, $F_i(W_{i0})$ is calculated.

1) If $F_i(W_{i0})$ is positive, then a solution has been found for the current row. In this case, the solution can either be further optimized by use of the remaining degrees of freedom or it can be accepted. In the latter case the algorithm will proceed to solve for the next row, W_{i+1}, until all N rows have been determined.

2) If $F_i(W_{i0})$ is negative, then we apply the following algorithm. The gradient of $F_i(W_i)$ is

$$\nabla F_i(W_i) = (-\mathrm{sgn}(T_{i1}), \cdots, -\mathrm{sgn}(T_{iN}), -\mathrm{sgn}(A_{ii}), 0).$$

The direction of the gradient for a given value of W_i will be denoted by d'. The vector d' is the direction of maximum increase of $F_i(W_i)$.

Since W_{i0} fails to satisfy (C2), but does satisfy (C1), the algorithm will produce a new vector W_{i1} by moving from W_{i0} in a direction that both increases F_i and forces W_{i1} to satisfy (C1). The required direction is given by

$$d = P^{\perp} d'$$

where $P^{\perp} = (E - B_i^T(B_iB_i^T)^{-1}B_i)$ and E is the $N+2$ dimensional identity matrix. The matrix $P^{\perp}$ projects any vector, $x \in R^{N+2}$, into the null space of B_i (the null space will be denoted by N_B). Since $d \in N_B$,

$$B_i(W_{i0} + \alpha d) = B_iW_{i0} + \alpha B_i d = B_iW_{i0} = H_i.$$

Thus by defining $W_i(\alpha) = W_{i0} + \alpha d$, W_i will satisfy (C1).

Due to the fact that F is linear when restricted to any one of the 2^{N+2} orthants of R^{N+2}, if the angle between d and d' is less than 90°, then d will be a direction from W_{i0} that increases F_i. The angle is less than 90° by the projection theorem. Since the gradient is constant until one of the components of $W_i(\alpha)$ becomes zero, $F(W_i(\alpha))$ is maximum at α^*, where α^* is chosen as the smallest positive number that sets one of the components of $W_i(\alpha)$ to zero. When the jth component of W_i (denoted $W_i^{(j)}$) is set equal to zero, the corresponding column of B_i should be removed. In this manner, $W_i^{(j)}$ will remain zero for all subsequent iterations, and at each iteration the dimension of N_B is reduced by one.

The above algorithm can be used iteratively. At each step (C1) is satisfied and F_i is increased. The algorithm is guaranteed to stop after no more than $(N+2-r)$ iterations since N_B begins with dimension $(N+2-r)$ and the dimension decreases by one at each iteration.

At the algorithm's termination, either (C2) will be satisfied and the network will have stored each element of L as an asymptotically stable equilibrium, or the algorithm will have failed to design the required network. In the latter case, the choice of α_i, $i = 1, \cdots, N$ (from (C3)) may be altered or new patterns may be chosen.

VI. Simulations

In the current section, we present two simulation examples using the synthesis method which is given in Section V. The purpose of the first example is to examine the average performance of (DS) based on a total of 130 13-neuron networks that were designed. The second example demonstrates an application of (DS) to pattern recognition. A neural network was designed to recognize 27 key patterns (the 26 upper case letters and a blank), each pattern being represented by a 9×9 pixel array.

Remark 4: In both of the following examples, the designation "binary" will mean that the only possible values that a variable may assume is −1 or 1.

Example 1: A 13-neuron system has 8192 (i.e., 2^{13}) possible binary inputs. For each design each of the possible binary inputs will, in turn, be applied as an initial condition for the system. The network will be allowed to evolve from its initial condition to a final state. We will interpret this final state as the network's binary response to the given initial condition. The Hamming distance from the initial condition to the final binary response will be used as a basis for estimating basins of attraction of the stored patterns. Specifically, we would like to determine how the network's performance will be affected by the number of patterns (r) to be stored. For each value of r, between 1 and 13 ($= N$), 10 trials were made. Each trial consisted of choosing a set of r linearly independent, but otherwise random, binary patterns of length $N = 13$. For each set of r patterns, a network was designed and simulated. The results of the ten trials for each value of r were then averaged. The results are presented graphically in Figs. 3–9.

In Figs. 3–8 the number of patterns to be stored is the independent variable. The dependent variable is the average number of binary inputs converging to a desired pattern from the Hamming distance (HD) noted in the caption of each figure.

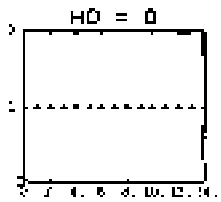

Fig. 1. As a function of the number of patterns stored (horizontal), the average number of binary input patterns converging from Hamming distance, HD (vertical), to a desired output.

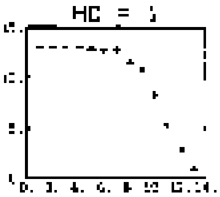

Fig. 2. As a function of the number of patterns stored (horizontal), the average number of binary input patterns converging from Hamming distance, HD (vertical), to a desired output.

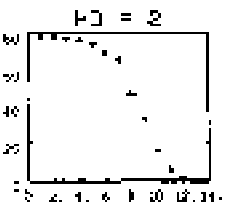

Fig. 3. As a function of the number of patterns stored (horizontal), the average number of binary input patterns converging from Hamming distance, HD (vertical), to a desired output.

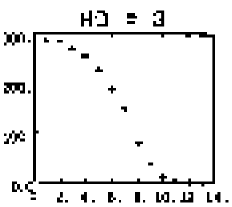

Fig. 4. As a function of the number of patterns stored (horizontal), the average number of binary input patterns converging from Hamming distance, HD (vertical), to a desired output.

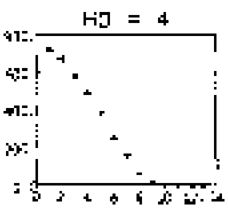

Fig. 5. As a function of the number of patterns stored (horizontal), the average number of binary input patterns converging from Hamming distance, HD (vertical), to a desired output.

Fig. 1 shows that each stored pattern attracts the binary input that is at a Hamming distance of zero. In addition to verifying that each desired pattern is in fact stored, this verifies that each stored pattern does have a nonempty basin of attraction.

Fig. 2 shows that for $s = 1$ to 4 (i.e., $\le 5\%$), in each of the ten trials, each of the stored patterns attracted all of

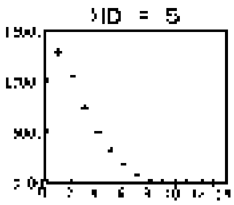

Fig. 6. As a function of the number of patterns stored (horizontal), the average number of binary input patterns converging from Hamming distance, HD (vertical), to a desired output.

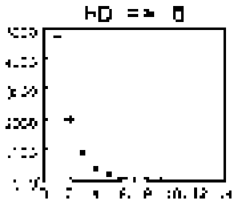

Fig. 7. As a function of the number of patterns stored (horizontal), the average number of binary input patterns converging from Hamming distance, HD (vertical), to a desired output.

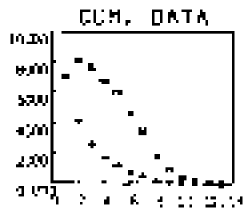

Fig. 8. As a function of the number of patterns stored (horizontal), the average number of binary input patterns converging from Hamming distance, HD (vertical), to a desired output.

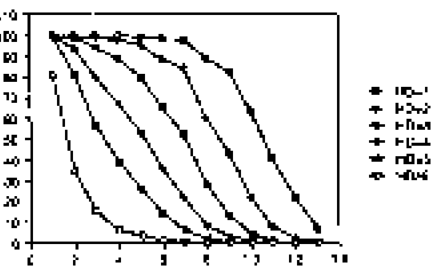

Fig. 9. The cumulative representation of the data from Figs. 1–7. Each plot displays, as a function of s, the average percentage of all the inputs at Hamming distance HD from a desired pattern that converge to the desired pattern.

the input patterns that only differed by a single bit. As s increased further, the attraction of patterns from a Hamming distance of one remained near the maximum of 13 until $s = 6$.

Each of the plots is expected to decrease monotonically as s approaches 8. This is because the number of possible binary inputs (8192) is constant. Hence, on the average, each stored pattern should attract $8192/s$ of the possible inputs. Fig. 8 plots the average total number of inputs attracted per output pattern (indicated by "+") and the average total number of the input patterns (indicated by "□") attracted to all of the s stored vectors. Interestingly, the lower plot of Fig. 8 does appear to decrease approximately as $1/s$. The upper plot remains

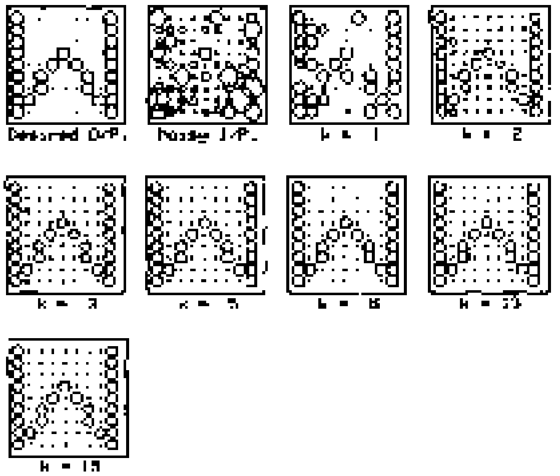

Fig. 10. Time evolution of a neural network storing 27 patterns. The network recognizes the letter "W" from an input condition that has added Gaussian noise with zero mean and 1.40 standard deviation to the "W" pattern.

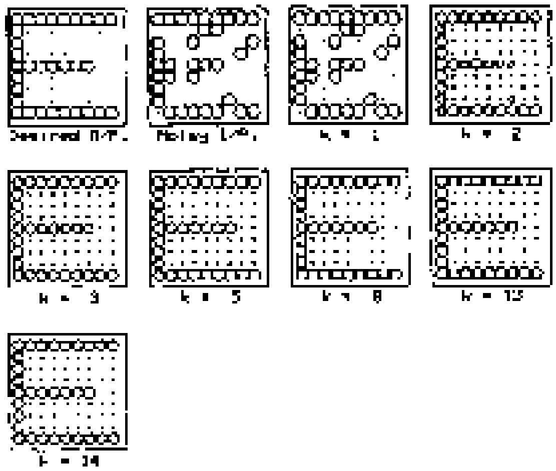

Fig. 11. Time evolution of a neural network recognizing the letter "E" from a noisy input that is at a Hamming distance of 15 from the desired output pattern "E".

above 8192/2 until $r = 6$. At this point, the extraneous equilibria are attracting almost half of the input patterns. The usefulness of networks with r near N is application-dependent. For example, in a secure voice recognition application, it is desirable for the majority of inputs to be attracted by extraneous equilibria. It should be noted that for each of the 10 trials with $r = 2$, all 8192 inputs were attracted to the two stored patterns.

Finally, Fig. 9 shows the percentage of input patterns that are at the Hamming distance of HD from a stored pattern, which are attracted to the stored pattern for $HD = 1, \cdots, 5$ and $HD = 6$. As is expected, the percentage of patterns converging from large Hamming distances drops off faster than the percentage from a smaller Hamming distance. The form of this graph is similar to the "waterfall" graphs common in coding theory and signal processing. Waterfall graphs are used to display the degradation of the system performance as the input noise increases. Using this type of interpretation, Fig. 9 displays that the ability of the network to handle small signal to noise ratios (large Hamming distances) decreases as the number of patterns stored (r) increases.

Example 2: In this example, we design an 81-neuron network. The given set of binary vectors corresponds to the 26 upper case letters and a blank pattern, i.e., $N = 81$ and $r = 27$. The purpose of the network is to recognize the key pattern from a noisy input pattern. The two different types of noise that will concern us in the simulations are the Gaussian noise and the binary noise, a kind of noise that will randomly flip binary bits. For the former, the noisy input pattern is generated by adding zero mean Gaussian noise with a given specific standard deviation to the key pattern (Fig. 10). For the latter, the noisy input pattern is generated by randomly flipping a certain number of the binary bits of the key pattern to produce an input pattern at the desired Hamming distance from the desired output vector (Fig. 11).

TABLE I

Rate in Percentage \ Noise Type	Zero-mean Gaussian Noise			Binary Noise		
	Standard Deviation(%)			Hamming Distance(Bits)		
	25[†]	75[††]	100	6	12	15
Converge to Key Pattern	95.5	89.1	65.0	94.4	85.2	[illegible]
Converge to Spurious State	1.7	10.2	17.5	5.5	14.6	[illegible]
Not converged*	0.0	2.0	2.5	0.0	0.2	0.1

* means the system did not converge within 100 time units. † ‡ correspond to SNR of 2 dB and 1.6 dB (= 10 $\log_{10}$ ...%) respectively.

The typical time evolution of the neural network is shown in Figs. 10 and 11 for the Gaussian noise and the binary noise, respectively. For each figure, the plot in the left upper corner is the desired output pattern and the next pattern to the right is the noisy input pattern. The rest of the plots are the time evolution results (the caption on the bottom of each plot indicates the current value of the time for each of the successive plots). The final result (i.e., the pattern to which the network converged) is shown in the plot of the last row, which is furthest to the right.

Remark 5: Figs. 10 and 11 use a small dot and a large circle to represent the output of neurons with values of -1 and 1, respectively. Output values between these two extremes are represented by circles whose size is scaled between the large circle and the small dot. This is the continuous nature of $G(\cdot)$ (discussed in Remark 1); we see that the final output state can never assume the desired values of -1 and 1. Instead, we recognize values sufficiently near the binary values as the binary values.

We have made six different sets of simulations, three for Gaussian noise with different standard deviations and three for binary noise with different Hamming distances.

TABLE II

Time Units \ Noise Type	Zero-mean Gaussian noise			Binary Noise		
	Standard Deviation			Hamming Distance		
	0.71	0.45	1.00	5	10	15
Average time to converge	15.4	20.2	24.8	15.7	15.5	21.7

*Not including the nonconvergent cases.

For each case, we test all of the key patterns twice (i.e., 2×27 = 54 trials). The results are summarized in Tables I and II.

From Table I we find that the performance of the system decreases, as expected, as the noise level increases. However, about 85% of the noisy input patterns with a 3-dB SNR (or a Hamming distance of 5 away from the key patterns) can be successfully recognized. Table II shows that the convergence time of the network increases as the input noise increases. This is expected because the increased noise moves the input pattern farther away from the equilibrium point (memory).

VII. Conclusions

In this paper we have presented a detailed stability analysis of and design algorithm for a class of neural networks. We utilize the stability analysis to derive the techniques for synthesizing a discrete time neural network. Compared to the already existing techniques for designing neural networks, the algorithm of the present paper is very easy to implement and yields guaranteed stability properties. To demonstrate the applicability of our results, we have presented two specific examples. Both examples are simulated on a digital computer using a Fortran program requiring less than 100 lines of code.

References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in Proc. Nat. Acad. Sci. U.S.A., vol. 79, pp. 2554–2558, 1982.
[2] —, "Neurons with graded response have collective computational properties like those of two-state neurons," in Proc. Nat. Acad. Sci. U.S.A., vol. 81, pp. 3088–3092, 1984.
[3] L. Personnaz, I. Guyon, and G. Dreyfus, "Information storage and retrieval in spin-glass like neural networks," J. Physique Lett., vol. 46, pp. L359–L365, Apr. 1985.
[4] —, "Collective computational properties of neural networks: New learning mechanisms," Physical Review A, vol. 34, pp. 4217–4228, Nov. 1986.
[5] L. N. Cooper, F. Liberman, and E. Oja, "A theory for the acquisition and loss of neuron specificity in visual cortex," Biological Cybernetics, vol. 33, pp. 9–28, 1979.
[6] J. Li, A. N. Michel, and W. Porod, "Analysis and synthesis of a class of neural networks: Variable structure systems with infinite gains," IEEE Trans. Circuits Syst., vol. 36, pp. 713–731, May 1989.
[7] E. Goles and G. Vichniac, "Lyapunov functions for parallel neural networks," in [illegible], pp. 165–181, 1986.
[8] Neural Networks for Computing, J. S. Denker, Ed., Snowbird, UT: American Institute of Physics Conference 151, 1986.
[9] A. N. Michel, J. A. Farrell, and W. Porod, "Qualitative analysis of neural networks," IEEE Trans. Circuits Syst., vol. 36, pp. 229–243, Feb. 1989.
[10] J. A. Farrell and A. N. Michel, "A synthesis procedure for Hopfield's continuous-time associative memory," IEEE Trans. Circuits Syst., vol. 37, pp. 877–884, July 1990.
[11] J. Li, A. N. Michel, and W. Porod, "Qualitative analysis and synthesis of a class of neural networks," IEEE Trans. Circuits Syst., vol. 35, pp. 976–986, Aug. 1988.
[12] D. B. Schwartz et al., "Dynamics of microfabricated electronic neural networks," Appl. Phys. Lett., vol. 50, pp. 1110–1112, Apr. 20, 1987.
[13] A. N. Michel and R. K. Miller, Qualitative Analysis of Large Scale Dynamical Systems. New York: Academic, 1977.
[14] R. K. Miller and A. N. Michel, Ordinary Differential Equations. New York: Academic, 1982.
[15] A. Albert, Regression and the Moore–Penrose Pseudoinverse. New York: Academic, 1972.
[16] Tzi-Dar Chiueh and R. Goodman, "A neural network classifier based on coding theory," American Institute of Physics, pp. 174–183, 1988.
[17] R. J. McEliece et al., "The capacity of the Hopfield associative memory," IEEE Trans. Inform. Theory, vol. IT-33, pp. 461–482, July 1987.
[18] J. A. Farrell, "Analysis and synthesis techniques for two classes of nonlinear dynamical systems: Digital controllers and neural networks," Ph.D. dissertation, University of Notre Dame, Notre Dame, IN, May 1989.
[19] D. W. Tank and J. J. Hopfield, "Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," IEEE Trans. Circuits Syst., vol. CAS-33, pp. 533–541, May 1986.
[20] A. N. Michel, "On the status of stability of interconnected systems," IEEE Trans. Automat. Contr., vol. AC-28, pp. 639–653, June 1983.
[21] M. A. Pai, Power System Stability. Amsterdam, The Netherlands: North Holland, 1981.
[22] J. Li, A. N. Michel, and W. Porod, "Analysis and synthesis of a class of neural networks: Linear systems operating on a closed hypercube," IEEE Trans. Circuits Syst., vol. 36, pp. 1405–1422, Nov. 1989.
[23] A. N. Michel and J. A. Farrell, "Associative memories via artificial neural networks," IEEE Control Syst. Mag., vol. 10, pp. 6–17, Apr. 1990.
[24] F. M. A. Salam, "A model of neural circuits for programmable VLSI implementation," in Proc. 1989 IEEE Int. Symp. Circuits and Systems, Portland, OR, pp. 849–851, May 1989.
[25] —, "New artificial neural net models: Basic theory and characteristics," in Proc. 1990 IEEE Int. Symp. Circuits and Systems, New Orleans, LA, pp. 200–203, May 1990.

Anthony N. Michel (S'55–M'59–SM'79–F'82), for a biography and photo please see page 884 of the July 1990 issue of this TRANSACTIONS.

Jay A. Farrell (S'86–M'89), for a biography and photo, please see page 884 of the July 1990 issue of this TRANSACTIONS.

Hung-Fa Sun received the B.S. degree in electrical engineering from National Taiwan University in 1981 and the M.S. degree in electrical engineering from Marquette University in 1987. He is currently working on the Ph.D. degree in electrical engineering at the University of Notre Dame, Notre Dame, IN.

His professional interests are in digital signal processing and control systems.