Preventing Computational Chaos in Asynchronous Neural Networks

Jacob Barhen Vladimir Protopopescu

Center for Engineering Science Advanced Research
Computing and Computational Sciences Directorate
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6355
barhenj@ornl.gov

Abstract. One of the primary advantages of artificial neural networks is their inherent ability to perform massively parallel, nonlinear signal processing. However, the asynchronous dynamics underlying the evolution of such networks may often lead to the emergence of computational chaos, which impedes the efficient retrieval of information usually stored in the system's attractors. In this paper, we discuss the implications of chaos in concurrent asynchronous computation, and provide a methodology that prevents its emergence. Our results are illustrated on a widely used neural network model.

1. Introduction

Artificial neural networks are massively parallel, adaptive dynamical systems [1]. Their models are inspired by the general features of biological networks. In such networks, asynchronous behavior is prevalent. It arises from delays in nerve signal propagation, refractory periods, and adaptive thresholding [2]. Recently, there has been considerable interest in better understanding and exploiting the computational properties of asynchronous neurocomputing models (see [3-6] and references therein).

From an algorithmic perspective, two regimes have traditionally been considered for updating quantities of interest [7-9]. In the *synchronous* regime, all nodes simultaneously update their state variables. This implies that each node must receive, within the interval Δ characterizing the resolution of the discrete-event network dynamics, all the necessary information it needs for its computation from all the nodes to which it is connected. In the *asynchronous* regime, only one node (usually determined randomly) is allowed to update its state on the basis of its inputs, and only after state information has been received from all required nodes. Clearly, *this* type of asynchronicity limits the ability of a network to perform massively parallel, distributed information processing. Hereafter, we will refer to this regime as *sequentially asynchronous*. To date, both paradigms still provide the algorithmic foundation of available computational models [3-6].

The true (*concurrent*) computational *asynchronicity*, however, implies an uncoordinated, system–wide activity. In that context, there is a strong motivation to develop algorithms that can fully exploit such a behavior. It should be noted, however, that asynchronous relaxation algorithms have long been known to give rise to *computational chaos* [10].

In the sequel, we first discuss some implications of asynchronous computing. Then we provide a methodology that prevents the emergence of computational chaos to enable efficient retrieval of information stored in attractors of the network. Finally, we illustrate our results in terms of the well established Grossberg–Hopfield model [8].

2. Limitations of Previous Approaches.

Several important limitations can be associated with the lack of concurrent capabilities for asynchronous computing in present models of artificial neural networks. These include problems encountered during: (1) VLSI, optical, or opto-electronic computations; (2) discrete time simulations on large–scale high–performance computers; and (3) emulation of biological systems. For instance, complex global synchronization circuitry is typically needed in VLSI circuits and optoelectronic devices to neutralize the clock skew effects arising from variation in the physical [11] and/or optical [12] path lengths of the actual synaptic interconnections. Not only does this circuitry lack biological basis, it also limits the

overall network performance to operate at the rate of the slowest neuron, and enforces rigid firing sequences that would be difficult to sustain due to signal leakages and component instability. It is well known [13] that in large–scale networks such self–induced pathological activation could destabilize the entire neuromorphic system.

From a purely algorithmic perspective, let us consider a synchronous algorithm implemented on an MIMD parallel computer. The processors associated to sets of neurons must communicate their partial results to each other, at every instance of time specified by the precedence–constrained task graph obtained from the problem decomposition. Such almost sequential algorithms produce overheads in the form of load imbalance due to processor inactivity. The potentially lower processor utilization then enhances resource contention due to communication and coordination requirements, and can lead to severe performance degradation in a real-time network environment.

Finally, from a biological emulation perspective, the current synchronous paradigm implies that neurons are not allowed to evaluate a firing threshold without having to wait to receive excitatory or inhibitory input signals from *all* other neurons to which they are connected. Failure to receive input from some inoperating node in the sequentially asynchronous case could lead to blocking of the entire network. Both alternatives are quite unacceptable.

3. Basic Concepts

We begin by defining more precisely what we mean by concurrent asynchronous computation. Let N denote the total number of nodes (neurons) in the network. A quantity of interest, $u_n(t)$, is being estimated at each node n, where t indexes a discrete temporal sequence. Let $\boldsymbol{\varphi}$ be a nonlinear operator from $\mathcal{R}^N$ to $\mathcal{R}^N$, whose network components are expressed as $\varphi_n(u_1, u_2, \ldots u_N)$.

Definition. A *concurrently asynchronous system iteration*, denoted by the tuple $\{\boldsymbol{\varphi}, \mathbf{u}(0), \xi, \psi\}$, corresponding to the operator $\boldsymbol{\varphi}$, and starting from a given vector of initial estimates $\mathbf{u}(0)$, is a sequence of state iterates $\mathbf{u}(t)$ of vectors in $\mathcal{R}^N$, obtained by the following recursion:

$$u_n(t) = \begin{cases} u_n(t-1) & \text{if } n \notin S_t \\ \varphi_n(u_1(x_1(t),\ldots,u_N(x_N(t)) & \text{if } n \in S_t \end{cases}. \quad (1)$$

Here, the sets $x_n(t)$ index the availability of the n-th node's most recent updated state. The update configurations of the network are given by $\psi = \{ x_1(t), \ldots, x_N(t) \mid t = 1,2, \ldots\}$. S_t denotes the set of nodes that carry out an update at the t-th time grid point. The set $\xi = \{ S_t \mid t = 1,2, \ldots\}$ is the sequence of nonempty subsets of nodes that performed an update at each t.

Assumptions. Three operational assumptions are made. Two refer to the set ψ, and one constrains the set ξ. Specifically, we require:

- Each consecutive update uses only state information previously available at the node under consideration, i.e., $x_n(t) \leq t - 1$.
- More and more recent state information must be used in evolving the set of nodes; in other words, $x_n(t)$ considered as a function of t tends to infinity as t tends to infinity.
- Node n is not starved in ξ, i.e., there exists a finite natural number $s \in \mathcal{N}$, such that each node updates its estimate at least once in every s successive time intervals.

The above definition provides a formal framework for algorithms that implement concurrent, asynchronous network dynamics. Such a dynamics is capable of updating the nodes in an uncoordinated manner, where the neurons are seen as a collection of functionally cooperating processes, with no explicit dependencies to enforce waiting at synchronization points for the purpose of swapping partially computed results. Since the ensuing dynamics may become chaotic, additional tools are needed to guarantee that correct results are ultimately obtained. These tools are presented next.

4. Contraction Theorems

The concept of contraction [14] plays a fundamental role in the iterative solution of nonlinear equations. It is most useful to express contraction in terms of vector norms, which induces a partial ordering on $\mathcal{R}^N$.

Definition. An operator $\boldsymbol{\varphi}$: $\mathcal{D} \subset \mathcal{R}^N \rightarrow \mathcal{R}^N$ is called a $\boldsymbol{\Phi}$-contraction on a set $\mathcal{D}_0 \subset \mathcal{D}$, if there exists a linear operator $\boldsymbol{\Phi} \in L(\mathcal{R}^N)$ with the following properties:

- $|\varphi(\mathbf{u})-\varphi(\mathbf{v})| \le \mathbf{\Phi}\,|\mathbf{u}-\mathbf{v}|$ $(2-a)$
- $\mathbf{\Phi} \ge 0$ $(2-b)$
- $\rho(\mathbf{\Phi}) < 1.$ $(2-c)$

The first property implies Lipschitz continuity. Indeed, $\mathbf{\Phi}$ is often referred to as the Lipschitz matrix of φ**.** The latter requirements, namely non-negativity and spectral radius of $\mathbf{\Phi}$, generalize the typical specification of the contractive constant used in conjunction with the usual norm on $\mathcal{R}^N$. The existence of a fixed point is then given by the following theorem.

Contraction Mapping Theorem. Suppose that φ: $\mathcal{D} \subset \mathcal{R}^N \to \mathcal{R}^N$ is a $\mathbf{\Phi}$-contraction on the closed set $\mathcal{D}_0 \subset \mathcal{D}$, such that $\varphi(\mathcal{D}_0) \subset \mathcal{D}_0$. Then, for any $\mathbf{u}(0) \in \mathcal{D}_0$, the sequence

$$\mathbf{u}(t+1) = \varphi(\mathbf{u}(t)) \qquad t = 0,\ 1,\ 2,\ \ldots \tag{3}$$

converges to the *unique* fixed point (denoted $\mathbf{u}(\infty)$) of φ in $\mathcal{D}_0$, and the following error estimate holds for $t = 0,1,2, \ldots$:

$$|\mathbf{u}(t+1)-\mathbf{u}(\infty)| \le (\mathbf{I}-\mathbf{\Phi})^{-1}\ \mathbf{\Phi}\,|\mathbf{u}(t)-\mathbf{u}(t-1)|. \tag{4}$$

These concepts can be applied to study the convergence of concurrently asynchronous time-evolving processes in general, and neural networks in particular. In that context, a theorem by Baudet [15] is of particular relevance.

Baudet's Theorem. If φ: $\mathcal{R}^N \to \mathcal{R}^N$ is a $\mathbf{\Phi}$-contraction on the closed subset $\mathcal{D} \subset \mathcal{R}^n$, and if $\varphi(\mathcal{D}) \subset \mathcal{D}$, then any concurrent asynchronous iteration corresponding to φ and starting with a vector $\mathbf{u}(0) \in \mathcal{D}$, converges to a unique fixed point of φ on $\mathcal{D}$.

We will now apply these concepts to neurocomputing. For illustrative purposes, we will focus on the Grossberg–Hopfield (GH) network.

5. Asynchronous Neurocomputing

Consider the temporal evolution of a fully connected, GH–type neurodynamical system [8]. Such a system is modeled by the following system of coupled, nonlinear differential equations:

$$\frac{du_n}{dt} + a_n u_n = \sum_m T_{nm}\, g_m(\gamma_m u_m) + I_n. \tag{5}$$

Here u_n represents the internal state of the nth neuron. The strength of the synaptic coupling from neuron m to neuron n is denoted by T_{nm}, and the external bias is denoted by I_n. The sigmoidal function g_n modulates the neural response, γ_n denotes the gain of the transfer function of the nth neuron, and a_n represents the inverse of a characteristic time constant or a decay scaling term. To create a discrete event formalism, we replace the time derivative of u_n with a first order finite difference representation. We select the explicit (forward) Euler scheme,

$$\frac{du_n}{dt} \cong (\,u_n(t+1)-u_n(t)\,) \,/\, \Delta \quad , \tag{6}$$

despite its simplicity and occasional inadequacy, in order to illustrate an intriguing property of asynchronous computation. Indeed, it is well known that the forward differencing Euler scheme does usually not yield a correct integration of differential equations, except for very small discretization (time) steps Δ. Hereafter, we will show that under concurrent asynchronous processing, even for arbitrarily large but finite time delays (e.g., of the order of hundreds of Δ's), correct convergence to the attractors of the dynamical system (5) is achieved under appropriate constraints on the model parameters.

Let $\varphi_n(\mathbf{u})$ denote the nth component of the GH operator obtained from Eq. (5) using the discretization scheme (6). We have

$$\varphi_n(\mathbf{u}) = u_n + \Delta(-a_n u_n + \sum_m T_{nm} g_m(\gamma_m u_m) + I_n). \tag{7}$$

We seek convergence to fixed-point attractors of (5). For any two phase-space points $\mathbf{u}$ and $\mathbf{v}$ in a domain of attraction, we can write

$$\varphi_n(\mathbf{u}) - \varphi_n(\mathbf{v}) = (1-\Delta a_n)(u_n - v_n) + \Delta \sum_m T_{nm}[g_m(\gamma_m u_m) - g_m(\gamma_m v_m)]. \tag{8}$$

Taking the vector norm, we obtain

$$|\varphi_n(\mathbf{u}) - \varphi_n(\mathbf{v})| \le |1-\Delta a_n| \cdot |u_n - v_n| + \Delta \sum_m |T_{nm}| \cdot |g_m(\gamma_m u_m) - g_m(\gamma_m v_m)|. \tag{9}$$

We assume that, for each neuron, the transfer function g_n: $\mathcal{R} \to [-1, +1]$ is of class $\mathcal{C}^1$, and that $|g_n'| \le 1$. This is indeed the case for the neural response functions usually considered, i.e.,

$$g(\gamma u) = \tanh(\gamma u) \qquad \text{or} \tag{10-a}$$

$$g(\gamma u) = (1+e^{-\gamma u})^{-1}. \tag{10-b}$$

Then, the Mean Value Theorem implies that there exists a number $z \in \mathcal{R}$ such that

$$g_n(\gamma_n u_n) - g_n(\gamma_n v_n) = g_n'(z)\gamma_n(u_n - v_n). \qquad (11)$$

Thus

$$|\, g_n(\gamma_n u_n) - g_n(\gamma_n v_n)\,| \;\le\; |\,\gamma_n\,| \cdot |\, u_n - v_n\,|. \qquad (12)$$

Regrouping all terms, we obtain

$$|\,\varphi_n(\mathbf{u}) - \varphi_n(\mathbf{v})\,| \;\le\; |\,1 - \Delta a_n\,| \cdot |\, u_n - v_n\,| + \Delta \sum_m |\, T_{nm}| \cdot |\,\gamma_m\,| \cdot |u_m - v_m\,|. \qquad (13)$$

Let us now define a matrix $\boldsymbol{\Phi}$ in the following manner:

$$\Phi_{nm} = |\,1 - \Delta a_n\,|\,\delta_{nm} + \Delta\,|\, T_{nm}| \cdot |\,\gamma_m\,|. \qquad (14)$$

We see that, by definition, $\boldsymbol{\Phi}$ is non-negative. Considering Eqs. (13–14), we observe that

$$|\,\varphi_n(\mathbf{u}) - \varphi_n(\mathbf{v})\,| \;\le\; \sum_m \Phi_{nm}\,|u_m - v_m\,|, \qquad (15)$$

or, equivalently

$$|\,\varphi(\mathbf{u}) - \varphi(\mathbf{v})\,| \;\le\; \boldsymbol{\Phi} \cdot |\,\mathbf{u} - \mathbf{v}\,|. \qquad (16)$$

Thus, the GH operator φ is Lipschitzian with Lipschitz matrix $\boldsymbol{\Phi}$. From Baudet's theorem, for φ to converge to a fixed point in an appropriate basin of attraction, the spectral radius of $\boldsymbol{\Phi}$ must be less than one. Our basic idea, here, is to use this requirement to establish constraints on the model parameters. In order to produce an operational statement, we invoke the Beckenbach – Bellman theorem [14]. For any vector $\mathbf{y}$ with positive components, we can write:

$$\min_{1\le n\le N} \frac{\sum_{m=1}^{m=N} \Phi_{nm}\, y_m}{y_n} \;<\; \rho(\Phi) \;<\; \max_{1\le n\le N} \frac{\sum_{m=1}^{m=N} \Phi_{nm}\, y_m}{y_n}. \qquad (17)$$

In particular, we can choose all vector components y_n to be equal. The contraction requirement, $\rho(\boldsymbol{\Phi}) \le 1$, then translates into

$$\max_{1\le n\le N} \sum_{m=1}^{m=N} \Phi_{nm} < 1. \qquad (18)$$

Recalling equations (2-b) and (14), we see that the above inequality induces constrained inter-relationships between the values of the model parameters a_n, Δ, γ_n, and T_{nm} (with n, m = 1...N). Explicitly, one obtains

$$\max_n \{\, |\,1 - \Delta a_n\,| + \Delta \sum_m (|\, T_{nm}| \cdot |\,\gamma_m\,|)\,\} \;<\; 1 \qquad (19)$$

and

$$|\,1 - \Delta a_n\,|\,\delta_{nm} + \Delta\,|\, T_{nm}| \cdot |\,\gamma_m\,| \;>\; 0. \qquad (20)$$

It is important, at this stage, to emphasize that the reader should not confuse arbitrarily large (but finite) delays with large values of Δ. If that were the case, Eq. (19-20) would require arbitrarily small values for γ_n, the system would become almost linear, and chaos would be excluded by definition. In our paradigm, delays are handled in the framework of a set-theoretic formalism (they appear implicitly in the set ξ). Thus, we allow for arbitrarily large delays, without affecting the structure of the dynamical system.

To fix the ideas, and without loss of generality, let us consider the simplest situation where all gain parameters γ_n are positive, equal and set to γ. Then, convergence of the concurrently asynchronous dynamics of the GH model will be guaranteed if the parameters satisfy the following relationships.

$$0 < a_n < \frac{1}{\Delta}\;;\quad 0 < \gamma \;< \min_n \frac{a_n}{\sum_m |\, T_{nm}\,|}\;; \qquad (21-\text{a})$$

$$\frac{2}{\Delta} > a_n > \frac{1}{\Delta}\;;\quad 0 < \gamma \;< \min_n \frac{2 - a_n\Delta}{\Delta \sum_m |\, T_{nm}\,|}. \qquad (21-\text{b})$$

We note that these conditions, once ensured, do not have to be changed during the calculations, even if a node temporarily fails. Indeed, for such an occurrence, the corresponding parameters a_n, T_{nm}, m = 1, ..N, are set to zero, and the inequalities (21-a, 21-b) continue to be satisfied.

6. Application

We now illustrate the dynamic behavior of the GH model for an associative memory problem. Fully connected networks of sizes varying by several orders of magnitude were studied. We find that the sequential asynchronous updating algorithm leads to the slowest convergence to a stored memory. The synchronous update leads to the fastest *rate* of convergence. Note, however, that for large–scale systems that have to be implemented on high–performance parallel computers, or for

device–level implementations, strict clock synchronization mechanisms are required to implement this paradigm. This, in turn, leads to significant degradation of the real-time performance of the system. For concurrently asynchronous updating, we observe that, in absence of proper conditioning (as specified by Eq. (21) above), networks exhibit sustained oscillations and convergence to spurious memories. However, when the requirements of Eq.(21) are satisfied, convergence to the correct stored attractors is achieved despite large communication delays (for instance, up to 2000Δ) and globally inconsistent state information, i.e., neurons operate using the latest information locally available to each of them, which differs from the actual states on the network.

The illustrations shown below refer to a network of 8 neurons. They display the network behavior under different time delays and parameter settings for an associative memory problem. Patterns are 8-dimensional vectors. Thus, a typical stored memory (pattern) would look like (+1, +1, −1, −1, −1, −1, +1, −1). Note that component values were set to ± 0.8 rather than ± 1 to avoid the asymptotic region of the sigmoid transfer function. A typical input probe would look like (+0.7, +0.9, −0.5, −0.7, −0.4. −0.9, +0.3, −0.8).

Each figure comprises four quadrants. The upper left region indicates the frequency of sign changes (zero crossings) in the state of a neuron over an interval of 100Δ. The upper right region shows the absolute magnitude of neuronal activities as they evolve in time. The lower left region plots all neuronal activities against the activity of neuron V2 (the second of the 8 neurons).

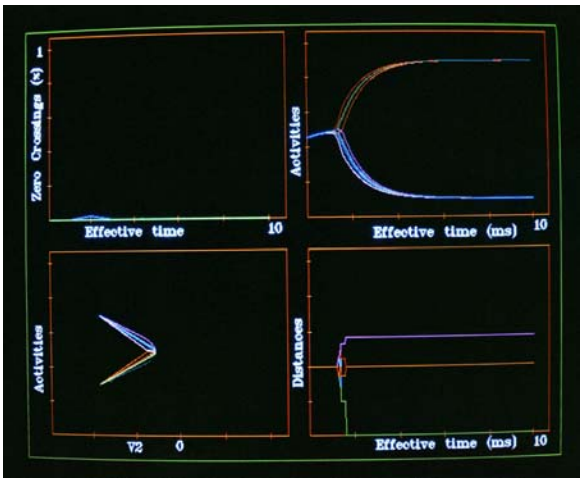

Figure 1. System evolution under concurrently ***synchronous*** updating.

Finally, the lower right region displays the temporal evolution of the Euclidean distance between the current state components and the stored memory to which they are expected to converge.

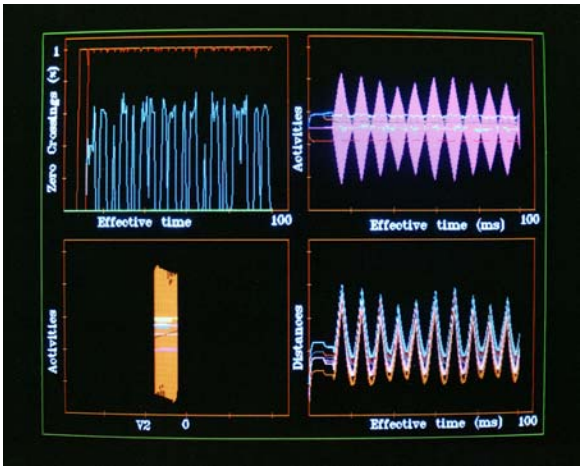

Figure 2. System evolution under concurrently ***asynchronous*** updating with two *ill-conditioned* neurons.

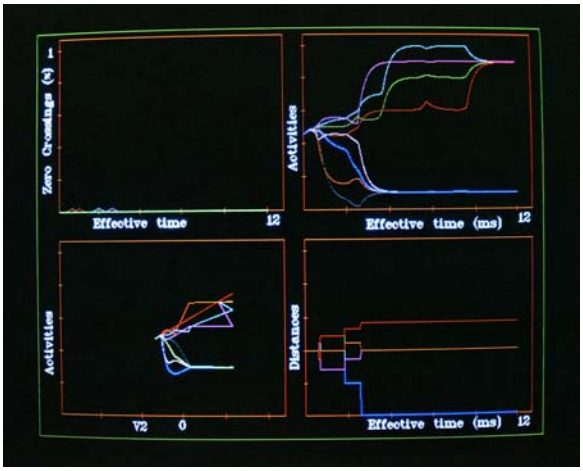

Figure 3. System evolution under concurrently ***asynchronous*** updating with *large* time delays.

Here, Fig. 1 illustrates the concurrently synchronous dynamics, i.e., an idealized situation in which no delays occur in information propagation between neurons. Figure 2 illustrates the aperiodic oscillations that arise when the stable dynamical bounds, specified by Eq. (21), are violated by 2 neurons (V1 and V4). Finally, Fig. 3 illustrates convergence under concurrently asynchronous conditions, with propagation delays up to 100Δ.

In addition to the prevalent notions on instability in neural networks that attribute oscillatory behavior mainly to the topology of the synaptic inter-

connection matrix [7], we were able to demonstrate that it can be a manifestation of emergent *computational chaos*. An analysis of the Lyapunov spectrum, computed from the time series that follows the evolution of the average component difference from a stored memory, gave the following indications. For an ill-conditioned model (e.g., $\Delta = 0.002$, $a = 1000.$, $\gamma = 10{,}000.$, and $\sum |T_{nm}| = 84.$) the largest Lyapunov exponent was found to be $+ 1.49$. The same calculations, performed for a concurrently asynchronous (contracting) system (e.g., $\Delta = 0.002$, $a = 100.$, $\gamma = 1.$) produces a value of $- 1.09$ for the largest exponent, thereby preventing the emergence of computational chaos.

7. Conclusions

The biggest promise of artificial neural networks as computational tools lies in the hope that they will be able to emulate the information processing capabilities of biological systems. Their paradigmatic advantages (i.e., their inherent ability to perform distributed, massively parallel, asynchronous information processing) cannot, however, be fully realized under the existing neurodynamics relaxation schemes. In particular, full (concurrent) asynchronicity has not been used to date, since it often engenders computational chaos. In this paper, we have derived conditions that ensure that computational chaos does not occur. These conditions are invariant with respect to a change of the number of nodes (neurons) during the computation process.

The new methodology was illustrated on an associative memory application modeled via the Grossberg–Hopfield formalism. The Lyapunov exponents were calculated to characterize the network dynamics. For concurrently asynchronous algorithms, we observed the emergence of computational chaos in the absence of proper conditioning. When network design obeyed the constraints derived in this paper, robust operation was demonstrated. It resulted in convergence to the correct memory patterns even in the presence of considerable inter-node communication delays.

ACKNOWLEDGEMENTS

This research was performed at the Center for Engineering Science Advanced Research, Computer Science and Mathematics Division, Oak Ridge National Laboratory. Funding was provided by the Material Science and Engineering Division of the DOE Office of Science under contract DE-AC05-00OR22725 with UT - Battelle, LLC.

References

1. M. Hassoun, *Fundamentals of Artificial Neural Networks*, MIT Press (1995).
2. C. Marcus and M. Westervelt, "Stability of analog neural networks with delay", *Phys. Rev.* **A 39**, 347-359 (1989).
3. K. Yamanaka, M. Agu, and T. Miyajima, "A continuous-time asynchronous Boltzmann machine", *Neural Networks*, **10**, 1103-1107 (1997).
4. R. VanRullen and S. Thorpe, "Spatial attention in asynchronous neural networks", *Neurocomputing*, **26-27**, 911-918 (1999).
5. M. Benson and J. Hu, "Asynchronous self-organizing maps", *IEEE Trans. Neural Networks*, **11**(6), 1315-1322 (2000).
6. J. Matsuoka, Y. Sekine, K. Saeki, and K. Aihara, "Analog hardware implementation of a mathematical model of an asynchronous chaotic neuron", *IEICE Trans Fundamentals*, **E 85-A**, 389-394 (2002).
7. K. Cheung, L. Atlas, and R. Marks, "Synchronous versus asynchronous behavior of Hopfield's CAM", *Applied Optics*, **26**, 4808-4813 (1987).
8. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons", *Proc. Nat. Acad. Sci.*, **91**, 3088-3092 (1984).
9. N. Toomarian and J. Barhen, "Learning a trajectory using adjoint functions and teacher forcing", *Neural Networks*, **5**, 473-484 (1992); *ibid*, "Fast temporal neural learning using teacher forcing", *U.S. Patent No. 5,428,710* (June 1995); *ibid*, "Neural Networks Training by Integration of an Adjoint System of Equations Forward in Time", *U.S. patent No. 5,930,781* (July 1999).
10. D. Chasan and W. Miranker, "Chaotic relaxations", *Linear Algebra & Applic.*, **2**, 199-222 (1959).
11. C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley (1989).
12. J. Shamir, "Fundamental speed limitations on parallel processing", *Applied Optics*, **26**, 1567-1568 (1987).
13. B. Macukow and H. Arsenault, "Modification of the threshold condition for a content addressable memory based on the Hopfield model", *Applied Optics*, **26**, 34-36 (1987).
14. J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press (1970).
15. G. M. Baudet, "Asynchronous Iterative Methods for Multiprocessors", *Jour. ACM*, **25**, 226-244 (1983).