Fuzzy Chaos Computing: A New Paradigm for Uncertain Knowledge Processing

Toshinori Munakata, Computer and Information Science Dept.
Cleveland State University, Cleveland, OH 44115, U.S.A.
t.munakata@csuohio.edu

Abstract

Fuzzy systems have extensively been applied to process uncertain information and knowledge, and have mostly been implemented on traditional silicon based chips or electronic circuits. This paper proposes a new, non-silicon based computing paradigm called *fuzzy chaos computing*. In this scheme, fuzzy systems, such as fuzzy sets, logic (e.g., inference), knowledge-bases, and databases, are implemented by means of chaotic elements. Both digital and analog forms are considered for this new scheme. In the logic-gate based digital technique, a new *evolvable dynamic fuzzy architecture* is proposed, where the hardware system can evolve during the computation to the best fit for the required information processing. Fuzzy chaos computing inherits advantages of both fuzzy systems and chaos computing, and allows efficient processing of uncertain knowledge.

Key Words: New computing paradigm; chaos computing; evolvable dynamic fuzzy architecture

1. Introduction

New computing paradigms. For the past 40 years computer hardware has been dominated by silicon-based technology. Recently, computer architecture concepts based on totally new principles other than traditional silicon chips have been given much attention. While practical, everyday-use of these proposals as computing devices is yet to be seen, these ideas have stimulated the scientific community due to their fundamental nature, novelty, and potentials for new forms of information processing and applications. These concepts include quantum, atomic (e.g., carbon nanotube transistors), molecular (e.g., organic), DNA, optical, and chaos computing. Some of these new technologies are hoped to complement or replace the current computer architecture based on silicon chips. This paper considers implementing fuzzy systems on the platform of chaos computing.

Chaos. The study of chaotic systems, or chaos in short, has attracted much attention for the past 30 years or so. Chaos represents a deterministic dynamical system that is nonlinear, sensitive to initial conditions (the so-called butterfly effect), and exhibits sustained irregularity. We exploit these unique characteristics of chaotic systems to implement chaos computing.

Chaos is all around us and found in many disciplines, such as physics, chemistry, biology, medicine and engineering. Any chaotic system can be a candidate for chaos computing. For example, chaotic phenomena are found in lasers, electronic circuits, chemical systems, brains and hearts. Many researchers have worked in the field because of the theoretical and practical importance of the subject, and the challenging and fascinating features of extreme nonlinearity [5, 13]. In particular, a recurring theme of research into chaotic systems over the last 15 years or so has been that chaos provides flexibility in the performance of natural systems and provides such systems with a rich variety of behaviors that can be utilized for improved performance. Successful implementations of this concept have included the exploitation of chaotic behavior for control [17], synchronization [4, 16], encoding information [7], and communications [22].

Fuzzy systems. Born in the United States around 1965, fuzzy set theory has grown to a major scientific domain collectively called "fuzzy systems," which include fuzzy sets, logic, algorithms, control, and others. The most fundamental characteristic of a fuzzy system is

that it allows a gradual and continuous transition, say, from 0 to 1, rather than a crisp and abrupt change between binary values of 0 and 1. Fuzzy systems are suitable for uncertain or approximate reasoning, especially for the system whose mathematical model is hard to derive. For example, the input and parameter values of a system may involve fuzziness, may be inaccurate, or incomplete. Similarly, the formulas or inference rules to derive conclusions may be incomplete or inaccurate. Fuzzy logic allows decision making with estimated values under incomplete information. Fuzzy system approaches also allow us to represent descriptive or qualitative expressions such as "slow" or "moderately fast", and are easily incorporated with symbolic statements. These expressions and representations are more natural than mathematical equations for many human judgmental rules and statements [12, 13].

The hardware implementations of fuzzy systems have been extensive, both in digital and analog forms. Because of the prevalence of digital computing in general, digital fuzzy system implementations are dominated by silicon-based chips. Since many fuzzy systems deal with continuous values, analog implementations are often the most reasonable choice and they are typically realized by electronic circuits.

Fuzzy chaos computing. Fuzzy chaos computing is a new computing paradigm that implements fuzzy system functions, such as inference and control, on the platform of chaos computing. When it is realized, it will have advantages inherited from both systems. They include robustness and the ability to deal with hard problems inherited from fuzzy systems. Potential advantages from chaos computing are discussed in the next section.

2. Chaos Engineering and Computing

Applications of chaotic systems come in many forms. For simplicity, in this paper we divide the many forms into two categories: "chaos engineering" and "chaos computing." Chaos engineering exploits the unique characteristics of chaotic systems to manipulate, control or communicate a variety of systems. It has been applied to various engineering problems such as control, synchronization, encryption, and communications, as well as to other fields such as biomedicine [5, 2]. Most of the applications employ the analog form, while some the digital form, and many of the practical applications started after 1990. Some authors use the term "chaos computing" for some of these techniques [6]. Often chaos computing in this sense and fuzzy computing are discussed together under a bigger umbrella of soft computing.

Chaos computing in this paper is different form these engineering approaches and discussed in the next subsection. Unlike the engineering schemes, chaos computing builds a device that has the basic ingredients of a silicon based computer, such as logic gates, adders and memory. When we need to distinguish our chaos computing from chaos engineering clearly, we can call the former "logic-gate based digital chaos computing." Chaos engineering is overviewed in the subsequent subsection 2.2. Both "engineering" and "computing" techniques can be exploited for fuzzy chaos computing.

2.1 Logic-gate based digital chaos computing

In this paper we call this technique "chaos computing" for simplicity. For the past seven years a new computing scheme based on chaotic systems has been proposed [18, 14, 15, 19, 20]. This technique implements logical gates such as NOR, AND, OR, NOT, XOR, and NAND, and constructs arithmetic units and memory, i.e., sufficient ingredients to build a computer. Although an actual computer has not been built except construction of a few logic gates, its implementation appears feasible. Such a device can perform parallel computing utilizing high dimensionality of chaotic elements, and has unique dynamic architecture (like field programmable gate array) by system parameter manipulation. These features can be realized by utilizing a chaotic or certain nonlinear element. Chaos computing has many potential advantages including the following:

• High speed, low power – high performance of computing, e.g., the computing speed can be very fast.
• Reconfigurable or dynamic logical architecture, where logical gates in a network can change during computation. This is a somewhat similar idea to FPGA (Field Programmable Gate Array) or evolvable hardware.
• A universal form of computing, i.e., a general-purpose computer for a wide spectrum of applications.
• Robustness against noise.
• Simple yet cost effective.
• Abundant candidates in nature and engineering.
• Parallel and distributed computing
• Chaos computing can overlap and interact with other important technical areas such as nanotechnology.

In the following, we describe the basic principle of chaos computing by a 1-D (one-dimensional) chaotic element with one and two input values. The scheme can be extended to an n-D chaotic element, leading to parallel computing. In these schemes, we exploit threshold mechanisms of chaotic or very nonlinear systems.

Here we describe cases for two input values. Let I_1 and I_2 be the two "interpreted" inputs and O be the "interpreted" output. Each of these variables can assume either 0 or 1. Let X_1 and X_2 be the two "actual" inputs and Z be the "actual" output. Each of these variables can assume either 0 or δ, where δ is a positive constant. The actual values are similar to physical values; their values of 0 and δ correspond to 0 and 1 of interpreted values, respectively. The threshold mechanism of a chaotic system works as follows:

Step 1. Initialization of the state x of the system to x_0 and addition of external inputs: $x = x_0 + X_1 + X_2$.
Step 2. Chaotic update: $x \rightarrow f(x)$, where f(x) is a chaotic function.
Step 3. Threshold mechanism to obtain output, i.e. the output Z is:
$Z = 0$ if $f(x) \le x^*$; $Z = \delta = f(x) - x^*$ if $f(x) > x^*$.

x^* is called the threshold value of the system. The core of chaos computing is to realize various logical gates by carefully selecting the parameter values of δ, x_0 and x^*. Furthermore, by combining these logical gates, we can build adders and memory.

The next step is to design our system in such a way that it yields the desired input-to-output mapping. More specifically, given a chaotic function f(x), we want to have the three parameters, δ, x_0, x^*, to be consistent with the procedure and also to achieve the required mapping. Take, for example, the case of Z = AND (X_1, X_2). There are four possible input value combinations: (X_1, X_2) = (0, 0), (0, δ), (δ, 0), (δ, δ), which should yield outputs of: 0, 0, 0, δ. For our current purpose, however, only the following three cases are sufficient, combining the second and third input combinations, (0, δ) and (δ, 0), into Case 2.

Case 1: Both X_1, X_2 are 0. In this case, x in Step 1 is $x_0 + 0 + 0 = x_0$, and f(x) in Step 2 is $f(x_0)$. Since Z in Step 3 should be 0 in this case, we must have $f(x_0) \le x^*$.
Case 2: One of X_1, X_2 is 0, the other δ, i.e., (X_1, X_2) = (0, δ) or (δ, 0). In this case, x in Step 1 is $x_0 + \delta$, and f(x) in Step 2 is $f(x_0 + \delta)$. Since Z in Step 3 is 0 in this case for AND, we must have $f(x_0 + \delta) \le x^*$.
Case 3: Both X_1, X_2 are δ. In this case, x in Step 1 is $x_0 + 2\delta$, and f(x) in Step 2 is $f(x_0 + 2\delta)$. Since Z in Step 3 is δ in this case, first we must have $f(x_0 + 2\delta) > x^*$. Furthermore, $f(x) - x^* = f(x_0 + 2\delta) - x^*$ must end up with the value of $\delta > 0$ consistent with the value of δ in other places. Combining these two requirements, we write condition for Case 3 to be $f(x_0 + 2\delta) - x^* = \delta$.

It is necessary to have all the three conditions, $f(x_0) \le x^*$, $f(x_0 + \delta) \le x^*$, and $f(x_0 + 2\delta) - x^* = \delta$, to be satisfied simultaneously to implement Z = AND (X_1, X_2), since mapping from (X_1, X_2) to Z must hold for all combinations of (X_1, X_2). Conversely, when these three conditions are satisfied, Z = AND (X_1, X_2) holds. That is, the three conditions are necessary and sufficient for

$Z = \text{AND}(X_1, X_2)$. Similar considerations for the remaining logical operations lead to Table 1, a summary of necessary and sufficient conditions to be satisfied by each of the operations:

Table 1. Necessary and sufficient conditions to be satisfied by a chaotic element to implement each of the logical operations. Row 1 corresponds to input $(X_1, X_2) = (0, 0)$, Row 2 $(0, \delta)$ or $(\delta, 0)$, and Row 3 (δ, δ). In this table, δ is a common positive constant for all the operations, whereas, x_0 and x^* differ among the operations.

Operation	AND	OR	XOR	NAND	NOT
Conditions	$f(x_0) \le x^*$	$f(x_0) \le x^*$	$f(x_0) \le x^*$	$f(x_0) = \delta$	$f(x_0) - x^* = \delta$
	$f(x_{0+}\delta) \le x^*$	$f(x_{0+}\delta) - x^* = \delta$	$f(x_{0+}\delta) - x^* = \delta$	$f(x_{0+}\delta) = \delta$	$f(x_{0+}\delta) \le x^*$
	$f(x_0 + 2\delta) - x^* = \delta$	$f(x_0 + 2\delta) - x^* = \delta$	$f(x_0 + 2\delta) \le x^*$	$f(x_0 + 2\delta) - x^* \le x^*$	

Table 2. Numeric examples for x_0 and x^* for $f(x) = 4x(1 - x)$ and $\delta = ¼$.

Operation	AND	OR	XOR	NAND	NOT
x_0	0	1/8	¼	3/8	½
x^*	¾	11/16	¾	11/16	¾

As a case study, we consider a specific form of $f(x)$ called the logistic map: $f(x) = rx(1 - x)$, $0 < r \le 4$. This is a well-known one-dimensional function for its historic background and the chaotic characteristics it can exhibit despite its simplicity [13]. This map has widespread relevance to physical and biological chaotic phenomena. For example, electrical circuits and nonlinear oscillators can exhibit such phenomena. In the following, we consider a special case of the logistic map by selecting $r = 4$. Although other values of r close to 4 can also be used, $r = 4$ guarantees that the function is chaotic. Under this consideration, the expressions in Table 1 become more explicit, for example, $f(x_0) = 4x_0(1 - x_0)$.

Numeric Examples

We select the constant δ, common to both input and output and all logical gates, to be ¼. For AND operation, for example, selecting $x_0 = 0$ and $x^* = ¾$ suffices the three conditions given in Table 1 as follows.

$f(x_0) = f(0) = 0 \le ¾ = x^*$, $f(x_{0+}\delta) = f(1/4) = ¾ \le x^*$,
$f(x_0 + 2\delta) - x^* = f(½) - ¾ = 1 - ¾ = ¼ = \delta$

Table 2 shows such numeric values for x_0 and x^* for the logical gates under consideration.

Adders and Memory

We can implement a half adder by combining an AND and an XOR gates. We can build full and higher bit adders by combining the logical gates. Furthermore, by using logical operations such as AND or NAND, flip-flop computer memory based on integrated circuits can be constructed [3, 10].

Parallel and distributed chaos computing

The above *sequential* computing technique can be extended to *parallel* and *distributed* computing by various schemes. One approach is basically the same as many traditional parallel computers – execute many "processors," in this case chaotic elements, at the same time. Another approach is to utilize the high dimensionality, or the number of variables, associated with some chaotic systems. For example, in the case of a two-dimensional system, a dynamical system can generally be described by two recurrence equations, $x_{n+1} = f(x_n, y_n)$ and $y_{n+1} = g(x_n, y_n)$. By using these two variables we can achieve parallelism of degree 2. In general, a system

with n variables can implement parallelism of up to degree n. There are chaotic systems as high as n = 120; for example, certain neuronal cells can be realistically described by 120 variables [21, 22]. Furthermore, by combining the above two schemes in a hybrid approach with L n-dimensional elements, we can achieve parallelism of degree nL.

2.2. Chaos engineering

Although we place various techniques under chaos engineering, they come in many forms. The characteristics common to all these techniques are: 1) they all exploit chaos, and 2) they do not implement logic gates, adders or memory as does chaos computing.

Physical quantities in chaotic systems are all continuous. This is also true in the digital chaos computing discussed in the previous subsection. In a sense, we digitize the continuous physical quantities in the digital chaos computing to 0 and δ, representing binary 0 and 1, respectively, by using the threshold mechanism. We can use these continuous physical quantities in chaotic systems without digitizing them. This is the basis of *analog chaos computing*. For a system with one input x and one output z, for example, we can obtain an approximate value of z by chaotic update: $x \rightarrow z = f(x, t)$, where $f(x, t)$ is a chaotic function. The ergodic properties of chaotic systems guarantee that we can obtain any desired value close enough for practical applications.

Most chaos engineering applications are based on analog systems, the remainders are digital. Both analog and digital forms can be candidates for fuzzy chaos computing. Past application examples of analog based chaos engineering include the following [5, 6]:

• Stabilization and Control

The extreme sensitivity of the chaotic systems to tiny perturbations can be manipulated to stabilize or control the systems. The fundamental idea is that tiny perturbations can be artificially incorporated to either stabilize or control a large system - to keep it stable (stabilization) or to direct a chaotic system into a desired state (control). Examples of such applications include NASA satellite control, multimode lasers, and chemical reactions.

• Synthesis of chaotic systems

Artificially generated chaotic systems may be applied to certain types of problems to make the systems, either chaotic or non-chaotic, work better. The fundamental idea is that regularity is not always the best solutions, depending on the types of the problems. For example, artificially stimulated chaotic brain waves may some day help break up epileptic seizures. We can synthetically generate chaotic output for consumer products, such as air conditioners and fan heaters, to increase natural feeling for human comfort.

• Synchronized Chaos for Communications and Information Processing

Applications include transmitting messages and encryption.

The analog aspects of these techniques may be applied to analog-based fuzzy chaos computing. There has also been a study on a chaotic system that exhibits a fuzzy-like behaviour [1, 8, 24]. Examples of digital chaos engineering include applications to communications [9].

3. Implementation of Fuzzy Systems by Chaos Computing and Engineering

We now consider how fuzzy systems such as fuzzy sets, logic, inference, knowledge-based, databases and control, are implemented by chaos computing and engineering discussed above. Many schemes are possible depending on the types of fuzzy systems and also the types of chaos computing and engineering platforms. Detailed discussions of these topics are beyond the scope of this short report, and we describe only the basic ideas common to all types of fuzzy systems. Table 3 summarizes the types of chaos computing and engineering that can be applied to fuzzy system implementation. More details follow.

Table 3. Summary of chaos computing and engineering techniques for fuzzy system implementation.

Digital or Analog	Chaos Technique Employed	Fuzzy Implementation Note
Digital	Logic-gate based	Basically the same as silicon-based chip
Digital	Chaos engineering	Some selected applications
Analog	Chaos engineering	Similar to other analog fuzzy systems
Hybrid	A combination of the above	More versatile for best mixtures

3.1 Logic-gate based digital chaos computing as a general-purpose computer

We first describe a unique feature called *dynamic logic architecture* that can be realized by our chaos computing scheme. Because of the characteristics of chaos, we can achieve various logic gates, adders and memory as discussed in this paper by applying small changes to the parameters on the same element. For example, consider the one-dimensional logistic map example discussed earlier. By selecting $x_0 = 0$ and $x^* = ¾$, we have an AND gate; by changing them as $x_0 = 1/8$ and $x^* = 11/16$ on the same chaotic element, we can have an OR gate. Other types of gates such as XOR and NAND can be obtained in the same fashion by slightly (all in the order of 1/16) changing these parameter values. The flexibility of achieving various operation types by chaos computing lead to the new computing architecture concept called here dynamic logic architecture.

Since chaos computing is a general-purpose computation method, all types of fuzzy systems can be implemented on such a system. Features such as high speed, robustness and parallel computing can be utilized for fuzzy applications. A particularly interesting application domain unique to chaos computing is to utilize the dynamic logic architecture described above for new types of fuzzy system applications, collectively called *dynamic fuzzy systems*. They include *dynamic fuzzy knowledge base, dynamic fuzzy database and dynamic fuzzy control.* Here the term "dynamic" refers to a system that can change itself by either an internal mechanism and/or by acquiring new information from outside as time progresses. A knowledge base or a control system can evolve in many ways including changing its rules and the logic behind these rules.

One simple illustration may be: inference rules are combined by AND operations, but at some point of time, they need to be changed to OR operations. Instead of rewriting the rules, this dynamic approach would perform the same effect by simply changing the values of x_0 and x^*. Another example may be that there are similar inference rules where one version involves the AND operations corresponding to min operations, while another version replaces them with the OR operations corresponding to max operations. Instead of having the two versions, switching between AND and OR on one version could be more efficient. For many other types of such dynamic evolution, the dynamic logic architecture of chaos computing can be an effective platform. In turn, such dynamic architecture may lead to development of new dynamic fuzzy system concepts.

3.2 Digital chaos engineering

As stated before, certain chaotic systems can implement digital information processing. This in turn can implement fuzzy systems in digital form. This approach would be less versatile than the logic-gate based approach since it is not a general purpose computer. However, it may give high speed, efficient performance for selected applications.

3.3 Applications of analog chaos computing

In general, analog implementations of fuzzy systems (not necessarily based on chaos) have advantages and disadvantages when compared with their digital counterparts [11]. Advantages include: processing speed; functional density; continuous nature of input (e.g., sensors) and output (e.g., actuators), internal fuzzy values (e.g., membership functions), and processing time; no need to convert between analog and digital values; and possibly low cost. Disadvantages include: lack of reliable memory

and extensive programmability. These advantages and disadvantages may also apply to analog chaos computing.

Because of these advantages of analog implementations of fuzzy systems, there have been extensive studies on this subject [11]. The techniques can be applied to many areas of fuzzy systems, in particular to control. There has also been extensive research on the chaos side, on topics such as generating desired function forms by chaotic devices. We can combine these two areas – realizing fuzzy systems on the chaotic platform.

3.4 Digital-analog hybrid chaos computing

It is conceivable to employ both digital and analog computing in one system. The idea is that some operations can be executed more efficiently and easily by digital, components while the other operations are better performed by analog components, hence, hybrid. For example, typical fuzzy control requires three major steps: fuzzification, fuzzy inference, and defuzzification. In the fuzzification step, to determine applicable fuzzy membership functions, i.e., fuzzy variables (e.g., Positive Big) for a given input value, the digital scheme may be used. To determine the degrees of the input, the analog computation may be used. Similarly, in the fuzzy inference step, determining applicable fuzzy rules can be done by the digital scheme, and evaluating firing strengths is performed by analog components, and so on. This hybrid concept can be applied to other fuzzy systems dealing with uncertain information and knowledge.

Conclusions

The proposed fuzzy chaos computing may open up completely new ways of processing uncertain information. In addition to the above, look-up tables are simple ways of implementing fuzzy systems. For commercial applications, the principles of fuzzy systems (e.g., control) may not actually be carried out inside individual machines, since such realtime computation is too costly and time consuming. Instead, input to output mappings are determined at the factories, these mappings are recorded on computer chips, and the machines operate based on this information. This scheme can also be implemented by either digital or analog chaos computing.

Acknowledgement

The author is grateful to Professor Hans-Dieter Burkhard for his valuable comment.

References

[1] K. Aihara, "Neural networks and chaos," (in Japanese), *Journal of the Institute of Electrical Engineers of Japan*, vol.111, no.4, pp. 271-5, April 1991.

[2] K. Aihara and R. Katayama. "Chaos engineering in Japan," *Comm. ACM*, 38, 11, Nov., 1995, pp. 103-107.

[3] T.C. Bartee, *Computer Architecture and Logic Design*, New York, McGraw-Hill, 1991.

[4] W.L. Ditto and L.M. Pecora, "Mastering chaos," *Sci. Am.*, vol. 269, no. 2, pp. 78-84, 1993.

[5] W. L. Ditto and T. Munakata, "Principles and applications of chaotic systems," *Comm. ACM*, vol. 38, no. 11, pp. 96-102, 1995.

[6] Y. Dote and S.J. Ovaska, "Industrial applications of soft computing: a review," *Proceedings of the IEEE*, vol.89, no.9, pp. 1243-65, Sept. 2001.

[7] S. Hayes, C. Grebogi, E. Ott and A. Mark, "Experimental control of chaos for communication," *Phys. Rev. Lett.*, vol. 73, pp. 1781-1784, 1994.

[8] K. Judd and K. Aihara, "Neural networks and chaos," *1992 IEEE International Symposium on Circuits*

and Systems (Cat. No.92CH3139-3), May 1992, vol.6, p. 2765-8.

[9] G. Kolumbán, M. P. Kennedy, and L. O. Chua, "The role of synchronization in digital communications using chaos—Part I: Fundamentals of digital communications," *IEEE Trans. Circuits Syst. I*, vol. 44, pp. 927–936, Oct. 1997.

[10] M. M. Mano, *Computer System Architecture*, 3rd edition, Englewood Cliffs, Prentice Hall, 1993.

[11] D. Mlynek and Y. Leblebici, *Design of VLSI Systems*, posted at http://lsiwww.epfl.ch/LSI2001/teaching/webcourse/index.html as of Dec. 2005.

[12] T. Munakata and Yashvant Jani. "Fuzzy Systems: An Overview," *Comm. ACM*, vol. 37, no. 3, pp. 69-76. 1994.

[13] T. Munakata, *Fundamentals of the New Artificial Intelligence: Beyond Traditional Paradigms*, New York, Springer-Verlag, 1998, chap. 7.

[14] T. Munakata, S. Sinha and W. L. Ditto, "Chaos computing: Implementation of fundamental logical gates by chaotic elements," *IEEE Trans. Circuits and Systems – I*, 49, 11, Nov. 2002, pp. 1629-1633.

[15] T. Munakata and S. Sinha. "Implementation of Fundamental Logical Gates by 1-D Chaotic Elements," COOL Chips VI, Yokohama, Japan, April 16 – 18, 2003, p. 73.

[16] L.M. Pecora and T.L. Caroll, "Driving systems with chaotic signals," *Phys. Rev. A*, vol. 44, pp.2374-2383, 1991.

[17] T. Shinbrot, C. Grebogi, E. Ott, J. Yorke, "Using small perturbations to control chaos," *Nature*, vol. 363, pp. 411-417, 1993.

[18] S. Sinha and W. L. Ditto, "Dynamics based computation," *Phys. Rev. Lett.*, vol. 81, pp. 2156-2159, 1998.

[19] S. Sinha, T. Munakata and W. L. Ditto, "Flexible parallel implementation of logic gates using chaotic elements," *Physical Review E*, 65, March 2002, 036216, pp. 1-9.

[20] S. Sinha, T. Munakata and W. L. Ditto, "Parallel computing with extended dynamical systems," *Physical Review E*, 65, March 2002, 036214, pp. 1-7.

[21] R. Traub, and R. Miles, *Neuronal Networks of the Hippocampus*, New York, Cambridge Univ. Press, 1991.

[22] B.J. West (Ed.), *Patterns, Information and Chaos in Neuronal Systems*, Singapore, World Scientific, 1993, and references therein.

[23] G.D. vanWiggeren and R. Roy, "Communications with chaotic lasers," *Science*, vol. 279, pp. 1198-1200, 1998.

[24] W. Zhijie and K. Aihara, "A fuzzy-like phenomenon in chaotic autoassociative memory," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol.E85-A, no.3, pp. 714-22, March 2002.