ASSOCIATIVE AND RANDOM ACCESS MEMORY USING ONE-DIMENSIONAL MAPS

by

Y.V. Andreyev, A.S. Dmitriev, L.O. Chua, and C.W. Wu

Memorandum No. UCB/ERL M92/32

8 April 1992

ASSOCIATIVE AND RANDOM ACCESS MEMORY USING ONE-DIMENSIONAL MAPS

by

Y.V. Andreyev, A.S. Dmitriev, L.O. Chua, and C.W. Wu

Memorandum No. UCB/ERL M92/32

8 April 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

ASSOCIATIVE AND RANDOM ACCESS MEMORY USING ONE-DIMENSIONAL MAPS

by

Y.V. Andreyev, A.S. Dmitriev, L.O. Chua, and C.W. Wu

Memorandum No. UCB/ERL M92/32

8 April 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

ASSOCIATIVE AND RANDOM ACCESS MEMORY USING ONE-DIMENSIONAL MAPS

Yu.V.ANDREYEV and A.S.DMITRIEV

Institute of Radio Engineering and Electronics of USSR Academy of Sciences, Moscow, 103907, USSR

L.O.CHUA and C.W.WU

Electronics Research Laboratory, and Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, USA

A method for storing and retrieving information on the stable cycles of one-dimensional maps as proposed in [Dmitriev 1991, Dmitriev, Panas & Starkov 1991] is considered. Applicability of this method to store and retrieve two-dimensional pictures is demonstrated. Possible extensions by compressing information are discussed. An implementation of random access memory using a one-dimensional map is also considered.

1. Introduction

Content addressable memory or associative memory, unlike addressed memory used in conventional computers, appears to be the way human memory is organized. The writing and reading out of information are not based on the location of the memory cell but on the content of the information [Kohonen 1980].

There exists a number of ways to realize associative memory. One of the most popular among them is the neural network model [Hopfield 1982, Grossberg 1988, Carpenter 1989]. Such models are described as dynamical systems and objects being memorized or recognized correspond to basic attractors, such as stable equilibrium points.

Experimental data suggests that more complicated dynamics, including chaotic dynamics, play an important role in the information processing of biological neural systems [Skarda & Freeman 1987]. In this regard, it would be interesting to investigate information processing based on complicated nonlinear dynamics.

One-dimensional maps are simple models with complicated nonlinear dynamics [Sharkovsky et al. 1986, Sharkovsky et al. 1989]. They are used in studying how cycles of long periods play a role in information processing [Auerbach et al 1987], in investigating the problems involved in the reconstruction of invariant strange sets in terms of cycles [Cvitanovich 1988], in examining the feasibility of constructing grammars with allowed words on the basis of unstable periodic orbits [Procaccia 1988], and in analyzing information processing in systems with complicated dynamics [Wiegrinck & Tennekes 1990], among others.

A method for storing and retrieving information based on the stable cycles of one-dimensional maps was proposed in [Dmitriev 1991, Dmitriev, Panas & Starkov 1991] and simple examples demonstrating the method were given. In the present paper we discuss the proposed method in detail and extend it in various ways. More complicated examples are used to estimate the potential efficiency of the method. In particular, we will use one-dimensional maps to store and retrieve black and white and color pictures.

Strong and weak points of our proposed approach are elucidated. For example, the presence of identical fragments having a length greater than or equal to the number of storage levels proved rather typical in pictures with a large number of elements. More storage levels are needed to store such images, leading in turn to a sharp increase on the average number of iterations before converging to the cycle corresponding to the stored picture.

To increase the storage capacity we introduce a more effective coding of information blocks and show that this will also alleviate the problem of long transient times.

We will also show how to obtain an explicit compact representation of the maps with the stored information. In particular we will use the canonical piecewise-linear representation in [Chua & Kang 1977].

Finally, we will show how intermittency can be used to realize random access memory on one-dimensional maps.

2. Method for storing and retrieving information in one-dimensional maps.

Let us consider a string of symbols

$$a_1\ a_2\ a_3\ \ldots\ a_n \tag{1}$$

consisting of n elements. We will call such a string an information block. The elements of the string belong to an alphabet of **N** symbols. For example, the alphabet can be the digits in octal, decimal, hexadecimal notation, the English alphabet, or a set of colors.

We want to design one-dimensional discrete-time dynamical systems (one-dimensional maps) with **M** attractors (which in our case are stable cycles) such that the attractors correspond to **M** information blocks. Given such a map, and an initial condition on any of the attractors, the corresponding information block can be retrieved unambiguously from the iterates of the map. If the initial condition is arbitrary, then it can be in the basin of attraction of one of the attractors and the trajectory will converge to the attractor, i.e. an information block will be retrieved from the initial condition. Here we have an analogy with how attractors are used in Hopfield-type neural networks [Hopfield 1982, Hopfield 1984]. The difference is that our attractors are *stable limit cycles* instead of stable equilibrium points.

To design a map f with a cycle of period n consisting of the points $x_1,\ldots,x_n$, the map should satisfy $x_{i+1} = f(x_i)$ for $1 \le i < n$ and $x_1 = f(x_n)$. Therefore the map should have the following points in the graph (the x_m-x_{m+1} plane): (x_1,x_2), $\ldots(x_{n-1},x_n)$, (x_n,x_1).

If we draw a curve through these points, we would have a map with the corresponding cycle. This allows us to create maps containing an arbitrary cycle of a given period, assuming the cycle elements do not repeat themselves within one period. To make the cycle a stable limit cycle, we use the following fact from nonlinear dynamics: a limit cycle of a 1-D map is stable if the product of the derivatives at all of the cycle points on the map has magnitude less than 1. Therefore to make the limit cycle stable, we can simply draw a short line segment through each cycle point having a slope of magnitude less than 1. The intervals resulting from the projection of all of these line segments onto the x-axis will then belong to the basin of attraction of the limit cycle.

If an initial condition is set on any of the intervals belonging to the basin of attraction of a stable cycle, then the corresponding trajectory must asymptotically tend to the cycle. We define a neighborhood of each of the cycle points to correspond to an element of the alphabet, so that the cycle corresponds to an information block consisting of a string of symbols.

To illustrate this, let us design a map with a stable limit cycle corresponding to an information block. Consider an alphabet of **N** symbols. Assume for now that the information block is a string without repeating elements ($a_i \neq a_j$, for $i \neq j$).

Let us divide the unit interval into **N** equal segments and number them from 1 to **N**. Each point in the i-th interval corresponds to the i-th element of the alphabet. The center of the i-th interval is located at (i-0.5)/N. The map will consist of n short line segments inclined by an angle less than $\pi/4$, and connected to each other by straight lines. The coordinates of the middle points of these segments are

$$\begin{aligned}&((m_1 - 0.5)/\mathbf{N},\ (m_2 - 0.5)/\mathbf{N}),\ldots\\ &((m_{n-1} - 0.5)/\mathbf{N},\ (m_n - 0.5)/\mathbf{N}),\\ &((m_n - 0.5)/\mathbf{N},\ (m_1 - 0.5)/\mathbf{N}),\end{aligned} \tag{2}$$

where m_j is the number of the segment corresponding to a_j, the j-th element of the information block. The length of the projection of each segment on the x_m-axis should be less than or equal to **1/N** (though equal for all segments) and is chosen to be **1/N**, for simplicity. Finally, let us connect the endpoints of the segments to obtain a continuous 1-D map. The resulting map will contain a cycle passing through the centers of the segments. The cycle is stable, because

$$|k_1 k_2 \cdots k_n| < 1 \qquad (3)$$

where k_j is the slope of j-th segment.

Example 1. Let the alphabet be the first 10 letters of the alphabet **a, b, c, d, e, f, g, h, i, j**. We want to store the word **big** (an information block consisting of the three elements **b, i, g**) on a map. We divide the interval [0,1] on the x_m-axis into 10 equal segments (Fig.1a). Each segment corresponds to an element of the alphabet. The second, ninth and seventh segment on the interval [0,1] correspond to the letters of the word **big**. The middle points of these segments are x_b=0.15, x_i=0.85, x_g=0.65 which form a cycle of our map. Therefore the points (x_b, x_i), (x_i, x_g), (x_g, x_b) are points of our map in the (x_m,x_{m+1}) plane (Fig.1b). Through each point of the cycle we draw a short line segment centered at this point. The projection of the line segment onto the x_m-axis coincides with the corresponding interval (Fig.1c). For example, the projection of the line segment passing through (x_b, x_i)=(0.15, 0.85) coincides with the interval [0.1, 0.2]. Each line segment, which we call an *information district* has a fixed length and has a slope of magnitude less than unity. We will call the projection of an information district onto the x_m-axis an *information interval.* To complete the map, the endpoints of the information districts are connected to each other and to the ends of the interval [0,1] (Fig. 1d).

To obtain a map containing an arbitrary but finite number of stable cycles the same procedure is applied to obtain line segments going through points in each cycle. If each element of the alphabet occurs in at most one information block and at most once in an information block then the segments' projections onto the x_m-axis do not overlap. The line segments can then be connected with straight lines to obtain a complete map f.

When the initial condition x_0 is set within the segment corresponding to an element of one of the information blocks, then the trajectory $x_0,x_1,\ldots,x_i,\ldots,$ where

$$x_{i+1}=f(x_i), \tag{4}$$

will converge to the limit cycle corresponding to this information block.

Let the alphabet be the set of numbers **0,1,...,N-1**. The information block $a_1,\ldots,a_n$ corresponding to the cycle $x_1,x_2,\ldots,x_n$ is then simply given by the rule

$$a_i = \text{int}\,(\mathbf{N}\cdot x_i) \qquad 1 \leq i \leq n \tag{5}$$

where int(·) denotes the "integer" part of (·).

If the initial condition is one of the elements of the cycle then applying the map f(·) iteratively will generate the cycle which corresponds to the information block of symbols (1). In order to retrieve a particular string of symbols, a trajectory need only originate from one of the information intervals, because according to the correspondence between intervals and symbols of the alphabet, any point within an information interval corresponds to the same information block element.

When considering methods for storage and retrieval of information one of the most important characteristics is the information storage capacity, i.e. the maximum amount of information that could be stored and read out. In the simple version of the method discussed above the storage capacity is not large. However the potential information storage capacity can be substantially increased if the above procedure is generalized as follows.

As before, each element of an information block defines a segment, henceforth called the *level-1 segment*, within the unit interval. This segment is in turn divided into **N** sub-segments of length $1/\mathbf{N}^2$. The location of each sub-segment, called the *level-2 segment*, within the level-1 segment corresponds to the next element of the information block. This procedure can be repeated recursively to obtain level-3, level-4,..., level-q segments, where q is the total number of nested levels in this storage scheme. We will call q the number of storage levels. In this case, the unit interval is divided into

$$L = \mathbf{N}^q \tag{6}$$

sub-intervals. Each sub-interval [a,b] of length $\mathbf{N}^{-q}$ is a subset of "q"-nested sub-interval of length $\mathbf{N}^{-(q-1)},\ldots,\mathbf{N}^{-1}$; i.e. $[a,b] \subset [a_1,b_1] \subset \ldots \subset [a_{q-1},b_{q-1}]$ where $b_j-a_j=\mathbf{N}^{-(q-j)}$. Hence, each subinterval can be thought of as the code of a q-symbol string.

Let the alphabet be the numbers **0,1,...,N-1**. The sub-interval corresponding to the symbol a_i when storing the information block (1) is $[x_i^l,x_i^r]$, where

$$x_i^l = \sum_{j=1}^{q} a_{g(i,j)}/\mathbf{N}^j \tag{7}$$

$$x_i^r = x_i^l + 1/\mathbf{N}^q \tag{8}$$

and

$$g(i,j) = ((i+j-2) \bmod \mathbf{N})+1. \tag{9}$$

In other words, instead of coding n symbols a_1, a_2, ..., a_n, as n intervals of length $\mathbf{N}^{-1}$, we create n associated strings $a_1a_2...a_q$, $a_2a_3...a_{q+1}$, ..., $a_na_1...a_{q-1}$ and code *each* string into a "q"-nested sub-interval of length $\mathbf{N}^{-q}$. Note however that after the map is constructed, and we are retrieving an information block, the correspondence between points in the unit interval and the alphabet is still described by (5).

The retrieval of an information block is performed with a substring of the information block, henceforth called a block fragment. If the fragment length is equal to or exceeds the number of storage levels, then by our coding method, the fragment corresponds to at most one stored information block. The corresponding initial condition, which is extracted from the first q symbols, will automatically fall within the segment belonging to the basin of the corresponding attractor. If the fragment length is less than the number of storage levels, then more than one stored information block can have the same fragment as a substring.

Ambiguous retrieval, because of the map becoming multivalued, occurs only when two or more identical fragments of length exceeding q-1 exist in the stored information blocks. In this case a larger number of storage levels is needed for unambiguous storage and retrieval.

Example 2. Let us return to *Example 1.* The word **big** can be written using not only 1 level as before, but using several levels of storage as well. For instance, if we store the word **big** using two storage levels (q=2), then the coordinates of the periodic points will be $(x_b, x_i)=(0.185, 0.865)$, $(x_i, x_g)=(0.865, 0.615)$, $(x_g, x_b)=(0.615, 0.185)$. In this case, the length of the information intervals is $\mathbf{N}^{-q}=10^{-2}=0.01$. If we choose 3 levels of storage, then the periodic points are $(x_b, x_i)=(0.1865, 0.8615)$, $(x_i, x_g) = (0.8615, 0.6185)$, $(x_g, x_b)=(0.6185, 0.1865)$. The map resulting from storing the word **big** using 2 storage levels is shown in Fig. 2.

When implementing this on a digital computer, the limit on the number of storage levels is determined by the maximum number of significant digits used in the calculations, and by the length of the alphabet. Numerical simulations show that with a 10-symbol alphabet and an accuracy of 10^{-8}, six storage levels can be reliably realized. So we can make the information intervals as small as about 100 ($\approx$ 6 bits) times the accuracy we are working with. Hence an upper estimate of the limit capacity E of a one-dimensional map when using M significant binary digits, $M \geq 6$, is

$$E \propto 2^{M-6} \tag{10}$$

It follows from the above map design rules that the repetition of any subsequence of elements with length equal to or exceeding the number of storage levels in any information block or even in different blocks will lead to the indistinguishability of the corresponding map fragments. So in order to use the above method for storing and retrieving information unambiguously, the repetition of symbol groups longer than (q-1) should be avoided. This is our "time orthogonality" condition for ideal storage of images.

This restriction can be overcome by a further generalization of our map design and our information storage and retrieval algorithms to be discussed in section 7.

3. Relation with the Bernoulli shift.

Given an initial value x_0 in the unit interval, our one-dimensional map generates a sequence of iterations

$$x_0, \quad x_1 = f(x_0), \quad x_2 = f(x_1) = f(f(x_0)) \ \ldots \tag{11}$$

Let **N** be the alphabet length. Any number x_0 in the unit interval can be expressed in a radix **N** representation:

$$x_0 = \sum_{\nu=1}^{\infty} \alpha_\nu N^{-\nu} \equiv (0.\alpha_1\alpha_2\alpha_3 \ \ldots), \tag{12}$$

where $\alpha_\nu \in \{0,\ldots N\text{-}1\}$. The classic Bernoulli shift map for **N** symbols is defined by

$$x_{n+1} = \Psi(x_n) = N \cdot x_n \bmod 1, \qquad n = 0, 1, 2\ldots, \tag{13}$$

and is shown in Fig. 3. The action of the Bernoulli shift of **N** symbols on the radix **N** representation of x_0 consists of discarding the first digit after the **N**-ary point "$\cdot$", and then shifting the remaining sequence to the left. In other words, $(0.\alpha_1\alpha_2\alpha_3 \ \ldots)$ becomes $(0.\alpha_2\alpha_3\alpha_4\ldots)$.

If the length of the information block (or the total length of all information blocks) is big enough, and the "probability" of transition from any of the **N** elements of the alphabet to any other element is nonzero, then the rough structure of our map will resemble the Bernoulli shift map for **N** symbols $\Psi(x)$. In our coding scheme, the p-th digit after the **N**-ary point determines which level-p segment the point corresponds to. In our map, the first digit after the **N**-ary point determines which element the point corresponds to. The second digit after the **N**-ary point determines which element the first iterate of the point corresponds to. This observation is reflected in the fact that the rough structure of the map is close to that shown in Fig. 3.

It should be noted that the rough structure of the map will be close to that shown in Fig. 3 even after the "factor of magnification" is increased by $\mathbf{N}$ times or more, (if the total length of information blocks is sufficiently long) up to the "factor" of $\mathbf{N}^q$, where the map begins to differ from inclined straight lines. It is at this level of resolution where information is stored.

Consider an initial condition with a finite $\mathbf{N}$-ary expansion of q $\mathbf{N}$-ary digits. After q iterations of (13) the last digit will be moved to the first position. But in our map f we insert a new digit at the q-th position at each iteration. If the accuracy of the calculations is less than or equal to $\mathbf{N}^{1-q}$, then the map f is "expansive" and has a uniform invariant measure on the interval [0,1]. The Lyapunov exponent for scales larger than or equal to $\mathbf{N}^{1-q}$ is $\lambda = \ln \mathbf{N}$.

Example 3. Let us illustrate the above observation by the following example. The information block is the following sequence of 0's and 1's of length 25.

1011011000100010001101100

The alphabet is 0 and 1 and we store this information block using 10 levels. A schematic picture of the resulting 1-D map is shown in Fig. 4. Although a magnification of a small portion of this map shows some deviations from a straight line of slope 2, a rough view of the map is very close to the Bernoulli shift for two symbols.

The length of the information districts for this example is 2^{-10}. To verify our statement about the macroscopicaly rough and microscopically fine structure of the map we simply iterate the map using limited precision. We find that the iterations do not converge to the information block when the error made in each iteration is larger than the length of the information intervals.

We can also reduce the precision of the calculation by adding external noise. In this case, our system will be described by the equation

$$x_{n+1} = f(x_n) + \xi_n \tag{14}$$

where ξ_n is a random variable with a uniform probability distribution on the interval $(-\varepsilon,\varepsilon)$. Figure 5 shows schematically how the graph of map iterations versus ε would look like. As ε is increased, the system moves from a stable limit cycle to a chaotic state when the value of ε becomes more than ε_c. In the chaotic state, the iterations have a uniform probability distribution on the interval (0,1). The value of ε_c is approximately equal to one half of the length of the information districts.

4. Algorithm for storing and retrieving pictures.

Let us apply the method described in section 2 to store and retrieve two-dimensional pictures. First we divide a two-dimensional picture somehow into a set of elements. Then we choose an alphabet of these elements, and introduce a scanning order, to generate a string of symbols. Each of these steps may be done in different ways. We will proceed as follows.

The first step consists of a spatial digitization to transform the initial picture into a pattern with mxn cells. The color of each of these cells is digitized to the nearest color from a palette of a finite number of colors. The basis of the alphabet is the color palette, consisting of $\mathbf{N}_c$ colors. The pattern corresponding to the picture consists of mxn elements from this alphabet. We can express this pattern as a two-dimensional matrix

$$\mathbf{A} = [a_{ij}],\ i=1,\dots,m,\ j=1,\dots,n, \tag{15}$$

where each a_{ij} is an element of the alphabet.

The next step is the construction of the information block. As was noted before, this could be done in different ways (depending on the particular problem), but the transformation of the mxn pattern to an information block and the reverse transformation must both be one-to-one.

We will transform the two-dimensional pattern (matrix A) into an information block by reading it line by line from top to bottom. As a result we obtain a one-dimensional sequence of m×n symbols:

$$a_{11}...a_{1n}\ a_{21}...a_{2n}...\ a_{m1}...a_{mn}, \tag{16}$$

As a cycle of period p repeats itself after p symbols (p = mn in (16)), all cyclic permutations are equivalent and generates the same trajectory. We can say that the system loses its initial phase when converging towards the cycle.

This situation is undesirable when working with pictures, since a picture has a definite beginning and a definite end and when a cyclic permutation of such an information block is transformed back into a two-dimensional array, the result is often unrecognizable. Therefore we mark the beginning of the information block by putting a *label*, which constitutes a special symbol in the alphabet, at the beginning of the information block. Correspondingly, when restoring the pattern from the values of the mapping variable at the limit cycle, we consider the first element after the label as the beginning of the picture (the label itself is not displayed). Thus the alphabet is augmented with the special symbol for the label. The label element is used only once in each information block.

Example 4. Let us store in a one-dimensional map a picture of the letter **R**, as shown in Fig. 6. We express this picture as a sequence of binary elements (black and white) and add an element corresponding to the label at the beginning of the sequence. Assume that the alphabet consists of three elements 0, 1, 2, where 0 corresponds to black, 1 corresponds to white, and 2 corresponds to the label. Then the information block is written as

211111001000101000101111001010011 0011.

Now we should determine the minimum number of levels necessary to store the image *unambiguously*. From Fig. 6, observe that the above string contains 2 identical sub-strings or fragments (01000101) of length 8. Hence the number of storage levels cannot be less than 9. Let us use 10 storage levels to store the images in Fig. 7. The resulting map is shown in Fig. 8. The limit cycle corresponding to the picture of the letter **R** is also shown in Fig. 8. Note that the "rough" structure of this map resembles the map of the Bernoulli shift for three symbols.

In the case of color pictures the alphabet consists of $\mathbf{N}_c+1$ elements, the set of colors plus a symbol for the starting label. The main problem in storing such pictures is the possibility of long identical fragments. For example, in Fig. 11a the maximal length of identical fragments is more than 20 (therefore q needs to be larger than 20). Since the length of information intervals is proportional to $(\mathbf{N}_c+1)^{-q}$, the presence of long identical fragments in the picture requires high precision calculations. Furthermore, the average amount of transient time needed before a trajectory converges to a cycle for arbitrary initial conditions also increases with a decrease in the size of the information intervals.

To estimate the average time for a trajectory to converge to a stable cycle, when the initial condition is chosen arbitrarily on the interval [0,1], let us model this convergence process as a transient chaos. The dependence of the average time of a chaotic transition process on the system parameter was studied in [Grebogi, Ott & Yorke, 1986] for the case when the chaotic attractor transforms into a chaotic transition process as the attractor collides with the attractor border (crisis). The typical average "convergence" time T depends on the system parameter p as $T \propto |p-p_c|^{-\gamma}$, where p_c is the parameter value for the crisis point, and γ is called the critical exponent of the transition process. A theory for determining γ for two-dimensional maps has been developed and a comparison with numerical experiments has been given. This theory has been shown to be applicable also for the critical behavior in internal crisis, and for the estimation of the average time of transition process in controlling chaos. We will apply this theory to estimate the average time of convergence to limit cycles in our one-dimensional maps.

Consider the process of convergence to a stable limit cycle as a transient chaos.

The latter arises from a strange attractor by means of an internal bifurcation leading to the birth of a stable cycle. If we simply consider each section with slope k and an x-axis projection equal approximately to $\Delta \propto \mathbf{N}^{-q}$ as a perturbation in the Bernoulli shift map for **N** symbols, then this perturbation would be singular. Therefore we use the following procedure:

1. For every size of information districts and a fixed value for its slopes an average time of the transition process T is assumed to exist, and the probability distribution of the transition process duration is assumed to be exponential

$$p(\tau) \propto T^{-1}\exp(-\tau/T). \tag{17}$$

2. The dependence of T on the length of information intervals, as in many other cases of crises and internal bifurcations, is assumed to be

$$T \propto \Delta^{-\gamma} \tag{18}$$

for small Δ, where γ is the critical exponent of the transition process that is to be determined. The simplest approach for evaluating γ is as follows. The invariant measure of the strange attractor from the Bernoulli map $x_{n+1} = \mathbf{N} \cdot x_n$ **mod 1** before the birth of a stable limit cycle is uniformly distributed over the interval [0,1]. So when we introduce n_Σ information intervals of length Δ (the total measure of these intervals is $\mu(x) = n_\Sigma \cdot \Delta$), the probability of falling onto one of these intervals , assuming a uniformly probability distribution for the initial system, is $\mu(x)/\mu(1)$. Thus the average time in which the trajectory falls onto one of the information intervals is

$$T^{-1} \propto \mu(x) = n_\Sigma \cdot \Delta \tag{19}$$

Hence

$$T \propto (n_\Sigma \cdot \Delta)^{-1}, \text{ i.e., } \gamma = 1 \qquad (20)$$

In our case this equation becomes

$$T \propto (n_\Sigma)^{-1} \mathbf{N}^q \qquad (21)$$

So if the number of the storage levels is too high, the convergence to the cycle will not occur in a reasonable time. We call this excessive convergence time phenomenon an "exponential catastrophe". We will discuss how to overcome this problem in section 7.

5. Information capacity of associative memory based on one-dimensional map.

The information capacity of a one-dimensional map, designed for associative memory, is closely related to the questions arising from storing images. We want to know whether a given image can be stored for a chosen number of storage levels and a chosen alphabet length. Even more important is the question of whether a relatively small number of levels can be chosen to store a given image. These and other questions are typical when analyzing the potential of the method and comparing it with other methods for storing and retrieving information.

Consider the storage of the information block **174**. Let the alphabet be the numbers **0,1,...,9**. This block may be stored using only one level. On the same map, we can also store the information block **286**. But if we then try to store the information block **173**, we will find out that it is impossible, because the map becomes multi-valued. However, if we use 3 levels, then all three information blocks can be stored on the same map.

The absence of identical fragments having a length equal to or greater than the number of storage levels is our condition of "time orthogonality" for images to be stored and retrieved unambiguously using 1-D maps.

To answer the question about the information capacity of the method we should solve the following problem: how many images of length n can be stored using the symbols from an alphabet of size **N** and the number of levels equal to q?

Here we give the solution for the method of image storing without labels. First let us consider how many strings there are of length n in an alphabet with **N** symbols. Since we do not use a label symbol, all cyclic permutations are considered the same. For example, the sequences **174, 741, 417** represent the same information block for this method of storage without label. As there could be strings that are identical to some of their cyclic permutations (for example, a string consisting of the same symbol of the alphabet), the total number of different strings of length n is equal to or larger than $\mathbf{N}^n/n$.

Consider the case $q \geq n$. In this case the number of storage levels is greater than the image length. Let p be the number of information blocks that are identical to some of their cyclic permutations. These p information blocks cannot be stored on our 1-D map because they fail to satisfy the "time orthogonality" condition. All information blocks, except these p blocks, can thus be stored, since two different strings of length n can only have identical fragments of length less than n, and hence less than q. So these strings satisfy the "time orthogonality" condition. The total number of different strings of length n that can be stored is thus $(\mathbf{N}^n-p)/n \leq \mathbf{N}^n/n$

In the case $q \leq n$ the number of storage levels is less than the image length. Consider the information block

$$C = c_1c_2...c_n, \tag{22}$$

where the c_i's are elements of the alphabet. The information block C is stored at the level q, $q \leq n$. To check whether C satisfies the "time orthogonality" condition, we should compare all fragments of length q of the image, i.e.

$$C_i = c_{g(i,1)}c_{g(i,2)}...c_{g(i,q)}, \quad i=1...n, \tag{23}$$

where g(i,j) is as defined in (9). If all these fragments are different, then the information block can be stored. But then none of these fragments could be used in any other image to be stored.

The total number of the fragments with length q is equal to $\mathbf{N}^q$. When storing one information block we use n of them, which are all distinct because of the "time orthogonality" conditon. Any other information block containing any used fragments could not be stored. In other words, a stored information block of length n has n distinct fragments (substrings) of length q. Therefore the number of information blocks of length n that can be stored is less than or equal to $\mathbf{N}^q/n$.

Thus, the information *capacity* (**Cap**) of storing information blocks on stable cycles of one-dimensional maps without labels (with all images having length n) is

$$\mathbf{Cap} \le \begin{cases} \dfrac{\mathbf{N}^n}{n}, & n \le q \\ \dfrac{\mathbf{N}^q}{n}, & n \ge q \end{cases} \tag{24}$$

For example, let us consider how many six-letter words could be stored using 3 levels. The alphabet consists of 26 elements. The total number of information blocks (words), with cyclic permutations considered the same, is larger than $26^6/6$. If we store the word "cipher", then we would not be able to store information blocks containing "cip", "iph", "phe", "her", "erc", "rci" as substrings. Having stored the word "coffee" along with the word "cipher", we exclude six more fragments ("cof", "off", "ffe", "fee", "eec", "eco") from the list of available fragments. To store six-letter words using 3 levels (**N**=26, n=6, q=3), the capacity **Cap** $\le$ 2929.

It follows from equation (24) that **Cap** ≤ 1 when $n=\mathbf{N}^q$, and if $n > \mathbf{N}^q$ then no images can be stored. So for a given **N** and n, the minimum number q of storage levels is given by

$$q = \text{int}(\log_{\mathbf{N}} n) = \text{int}\left(\frac{\ln n}{\ln \mathbf{N}}\right) \tag{25}$$

The *relative capacity* (**RCap**) of the method, defined as the ratio of the maximum number of images that can be stored unambiguously and the total number of images is:

$$\mathbf{RCap} = \frac{\text{max. \# of images stored}}{\text{total \# of images}} = h(\mathbf{N},n,q) \le \begin{cases} 1 & , n \le q \\ \dfrac{1}{\mathbf{N}^{(n-q)}} & , n \ge q \end{cases} \tag{26}$$

The relative capacity depends on both q and n, and drops exponentially with a decrease on the number of storage levels with respect to the length of images.

6. Storing information via canonical piecewise-linear representation of 1-D maps

Maps with the stored information, as studied in this paper are *piecewise-linear* functions on an interval. In the process of studying such systems we have to analyze them in some cases as functions and their transformations. In particular, when analyzing the stability of an n-period cycle, it is convenient sometimes to use the n-th iterate f^n of the map f. Furthermore, it is useful to treat the map as a function when we insert some local corrections for removing the "parasitic" stable cycles or strange attractors. On such occasions and others, it is desirable to express this piecewise-linear function in a compact closed form. Such an explicit formula can be easily derived by applying the results of Chua & Kang [1977].

An arbitrary continuous piecewise-linear function defined on an interval is shown in Fig. 9. If we number the segments of this function from 0 (the leftmost element) to n (the rightmost element), and denote the slope of the j-th segment as "m_j", then this piecewise-linear function f(x) may be represented *exactly* by the formula

$$f(x) = a_0 + a_1 \cdot x + \sum_{j=1}^{n} b_j |x - x_j|, \tag{27}$$

where

$$a_1 = 1/2\,(m_0 + m_n) \tag{28}$$

$b_j = 1/2\ (m_j - m_{j-1}),\ j = 1,2\ldots\ n$ (29)

$$a_0 = f(0) - \sum_{j=1}^{n} b_j \cdot |x_j|. \qquad (30)$$

In our case f(0) = 0, and the initial information needed to specify the function is a prescribed set of points $(u_i,\ v_i)$, $i=1,\ldots L$, where $L = \sum_{j=1}^{M} n_j$, n_j being the number of elements in the j-th information block, and M is the total number of stored information blocks. Each $(u_i,\ v_i)$ is a point of a cycle of our map on the x_m-x_{m+1} plane. The length of the projection of an information district on the x_m-axis (= information interval) is given by

$\Delta = \mathbf{N}^{-q}$, (31)

where **N** is the number of elements in the alphabet, and q is the number of storage levels. The slope k of the information segments is also given.

Now we can express the coefficients a_0, a_1, b_j and x_j in (27) in terms of these parameters as follows:

$x_0 = 0,\ y_0 = 0,\ x_{2n+1} = 1,\ y_{2n+1} = 0$ (32)

$x_j = u_i - 1/2\ \Delta,$ $\quad j = 2i - 1,\ \ i = 1,\ldots n$ (33)

$x_j = u_i + 1/2\ \Delta,$ $\quad j = 2i$ (34)

$y_j = v_i - 1/2\ k{\cdot}\Delta,$ $\quad j = 2i - 1$ (35)

$y_j = v_i + 1/2\ k{\cdot}\Delta,$ $\quad j = 2i$ (36)

$m_0 = y_1/x_1$ (37)

$m_j = k,$ $\quad j = 2i - 1$ (38)

$m_j = (y_{j+1} - y_j)/(x_{j+1} - x_j)$ $\quad j = 2i$ (39)

The expressions (32) - (39) uniquely determine the coefficients for the 1-D map in (27) in explicit form.

7. Coding of information blocks.

In section 2 it was shown that an information block can be stored using q storage levels only if it doesn't contain coinciding fragments having a length greater than or equal to q. Therefore the main difficulty in storing information using the above method is in the storage of information blocks having substrings containing long identical fragments. It is clear that these fragments contain only a small amount of information, yet it is with them that our method has the most difficulty. This observation leads us to increase the the capacity of our 1-D maps significantly by the following generalization.

We will use the following coding scheme for such information fragments. The information block is first scanned from the beginning to the end. If during the scanning of an information block, or a set of blocks that are to be stored at the level q, we find two or more identical fragments with length q, then a *new* element is added to the alphabet, representing that very fragment, and all subsequent presence of this fragment in the block (blocks) are substituted by the new symbol of the alphabet. Then again we scan the information block (set of blocks) and if again we find (now with the alphabet of **N**+1 symbols) coinciding fragments with length q, then another element is added to the alphabet and so on, thereby creating a new alphabet that contains the initial one as a subset. In other words, we keep q fixed, while increasing **N** (and decreasing n in some sense) until the information block can be stored. This procedure allows us to store any information block using any number of storage levels greater than 1.

Example 5. Let the alphabet consist of two elements (black and white), and consider a picture consisting of an 8 by 8 array of black squares (Fig. 10). Let us store this information block using two levels. Following our procedure, the substring consisting of the first two elements of the block constitute a new element of our alphabet, as does the first four elements of the block.

We denote the elements of the initial alphabet as w (white) and b (black), and the added elements as b_2, b_4, b_8 etc. (Fig. 10). The information block will be coded in the new alphabet as a cycle of period 1 consisting of the element b_{64}. For a picture consisting of a 64 by 64 array of black squares, the new alphabet consists of the white element, the black element, 2 black elements (b_2), 4 black elements (b_4), 8 black elements (b_8), b_{16}, b_{32},...,b_{4096}. The new alphabet contains 14 elements, and the information block is described by a cycle of period 1, consisting of the 14-th element of the alphabet (b_{4096}).

Of course, the above examples are degenerate cases for illustrative purposes. Let us use this method to code the following color pictures.

Example 6. Consider the color pictures in Fig. 11 a,b,c. They consist of 16×24, 32×48 and 64×96 elements, respectively. The picture is drawn in 7 colors, so the initial alphabet consists of 8 elements (7 colors + the label marking the beginning of the picture). Table 1 shows the number of elements in the new alphabet (AL) and the period of the cycle (CP) corresponding to the coded pattern for various numbers of storage levels. As can be seen from Table 1 in all cases the storage can be performed with single precision (16 bits) calculations.

The transient process from the initial state converging to a picture with 32×48 = 1536 elements is shown on Fig. 12. Figure 12a corresponds to a typical transient, and Fig. 12b is the corresponding picture which appears right before converging to the stored picture shown in Fig. 11b.

Example 7. Figure 13 illustrates the simultaneous storage of two color pictures (a kitten and a elephant) on a 1-D map using eight colors and 3 storage levels. The number of elements in the new alphabet is 135, where 111 among them are necessary for storing the pictures. Figure 13a shows the typical picture corresponding to transient chaos, which contains portions of both stored images. Figure 13b shows the picture just before converging to the limit cycle in Fig. 13c. The map for storing these two pictures and the corresponding limit cycles are shown in Fig. 14. In Fig. 14a the limit cycle corresponding to the elephant is shown in yellow. In Fig. 14b the limit cycle of the kitten is shown.

The above examples demonstrate the use of coding to increase the storage capacity of the method. In particular it permits us to set the number of storage levels q to 3 or 4. The period of cycles corresponding to the pictures are usually reduced by 5 to 10 times.

How can we use our 1-D map as an associative memory model? In our initial storage method it suffices to take a fragment of the information block having a length not less than q, and use the corresponding point as an initial condition. Due to our map design procedure, the corresponding trajectory will immediately fall into the corresponding information interval, and the information block will be completely restored after the first n iterations.

The situation is the same for the storage scheme based on the coding method discussed in this section with the new alphabet. But we would like to retrieve a picture not by a fragment of the information block expressed in the *new* alphabet, but by a fragment of the picture in the *initial* alphabet. If we know that the fragment of the picture is an initial substring of the information block, then the problem of associative "recalling" can be solved by taking a fragment of the picture expressed in the initial alphabet and matching it from left to right against the elements of the new alphabet, starting from the longest elements in the new alphabet, and replacing it when a match is found. This will generate a fragment in the new alphabet, which is then used as an initial condition for the map.

8. Random access memory via 1-D map

In our preceding method for storing and retrieving information via 1-D maps, a *stable* limit cycle was used to store information blocks. We will show in this section the possibility to realize random access in one-dimensional associative memory, in the sense that the trajectory visits all regions which corresponds to information stored in the map. It is clear that we cannot use stable limit cycles because the trajectory of any point in the basin of attraction of a stable cycle will remain in the basin of attraction.

We want the trajectory to visit each region (or just the regions where information is stored) of the phase space from time to time. For this purpose we can use a strange attractor with a nonzero invariant measure of iteration values in every point on the interval [0,1]. However, a strange attractor will generally visit each region briefly.

But we can exploit the phenomenon of intermittency in dynamical systems for our random access memory. When intermittency occurs the phase trajectory of a system "lingers" in some definite regions of the phase space. In particular, we want the trajectory to be in the vicinities of the limit cycles corresponding to information blocks.

"Random access" in associative memory can therefore be realized by using *unstable* cycles for storing information blocks. Each cycle can easily be made unstable by letting the absolute value of the product of the slopes of the cycle points to be slightly larger than 1. Trajectories starting from the vicinity of a cycle will leave the cycle after some time and begins to wander in phase space until it gets into the neighborhood of the same limit cycle (or another limit cycle with the same stability property.) Then the trajectory will linger near this limit cycle for some time before leaving it again. If independent of the initial condition, the trajectory visits the neighborhoods of all points, then *intermittency* with respect to all cycles corresponding to stored information will be observed over short but *random* time intervals. Therefore it must be noted that intermittency takes place in a rather narrow region of the parameter space only.

Trajectories will leave the vicinities of limit cycles corresponding to stored information very quickly if the slopes of the information districts are too large. Other dynamical phenomena can then occur. Analysis of these aspects requires more investigation and will be reported elsewhere.

Here we restrict ourself to the following examples, which exhibit intermittency designed into our 1-D maps.

Example 8. In Fig. 15 is shown a simple example of intermittency between two cycles corresponding to the information blocks **97583** and **12345**. We choose the numbers **0,1,...,9** as our alphabet and use two storage levels. If the slopes of the information districts are chosen to be less than 1 in magnitude, then the trajectory of this 1-D map will converge to one of the two limit cycles (Fig. 15 a,b), as expected. However, if we make the slopes slightly bigger than 1, then we would observe intermittency (Fig. 15c). In this case, the system "wanders" between the two cycles.

Example 9. Let us demonstrate the possibility of organizing random access memory with a more complicated case. Let us store *three* information blocks corresponding to 3 color pictures of 48x32 elements. We use 8 colors (9 elements in the initial alphabet) and the stored information consists of the images of the 3 words **ORDER, FROM** and **CHAOS**. We store the information blocks using 2 levels. The method for generating the map is the same as that discussed in section 7 except that here, we fix the slope of the information districts to be slightly larger than one. A sequence of snap shots in Fig. 16 taken at various intervals show the intermittent appearance of *all* three pictures, but at a random yet robust order.

The trajectory visits all the regions where information is stored. By switching the slope of the information districts to be less than 1 in absolute value we can make the unstable cycle stable. The trajectory will then converge to a cycle, thereby retrieving the particular information block this trajectory happens to be at when the switching occurs. For example, we iterate the map used in Fig. 15c for 475 iterations so that the trajectory is near the information block **97583**. We then switch the map to the one used in Fig. 15a,b. The block **97583** is then retrieved as the limit cycle becomes stable (Fig. 17a). Similarly when we iterate the map used in Fig. 15c for 500 iterations and switch the map, the block **12345** is retrieved (Fig. 17b).

The idea about memory scanning organization using intermittency was discussed qualitatively by Nicolis [1986,1991]. Intermittency arising in chaotic neural-like systems with stored images was exhibited in [Aihara 1990]. It was shown in [Dmitriev, Idanova and Kuminov 1991] the possibility of using intermittency to achieve memory scanning in neural like systems with chaotic dynamics using an external oscillator which is similar to the pacemaker.

9. Discussion

The method for storing and retrieving information proposed in [Dmitriev 1991, Dmitriev, Panas & Starkov 1991], and applied in storing two-dimensional pictures has some features in common with the traditional models of associative memory developed for the analysis of temporal associations. But unlike these methods, the proposed method is based on the properties of one-dimensional maps as dynamical systems and the information corresponds to stable cycles.

We also discussed the strong and weak points of this method. For example, the presence of identical fragments with length greater than the storage level is rather typical in pictures with a large number of elements. Large storage levels are needed to store such images, leading in turn to the necessary sharp (exponential) increase of the average time for the iterative mapping to converge to the limit cycle corresponding to the stored picture.

To increase the storage capacity a more effective coding of information blocks is introduced, which is shown to prevent "exponential catastrophe".

Storing and retrieving information using stable limit cycles of one-dimensional maps has much in common with the problem of image recognition. Let us compare the main features of this method with the method discussed in [Haken 1987, Haken 1988]. A string of n symbols (information block) is also the basis of the Haken method. However, in Haken's approach the string is treated as a vector or a point in an n-dimensional phase space. In our case an information block is coded as n points in a q-dimensional phase space (where q is the number of storage levels.)

$$\mathbf{X}_1 = x_1 \ldots x_q, \ \ldots \ \mathbf{X}_{n-q+1} = x_{n-q+1} \ldots x_n, \ \ldots \ \mathbf{X}_n = x_n x_1 \ldots x_{q-1}. \qquad (40)$$

Then the points in equation (41) is "packed" into a one-dimensional map (with one-dimensional phase space), designed so as to create a stable limit cycle passing through the points corresponding to the elements of the information block.

It should be noted that information is not compressed in this procedure. The compression occurs when the generalized coding procedure described in section 7 is used.

There is some analogy to the application of Karhunen-Loeve (K-L) decomposition in Haken's method. It gives a method of systematic decomposition of the image by uncorrelated components (features). Then we can choose to ignore some of these components. The coefficients of the K-L decomposition contain all the information necessary to restore an image. The coefficients of the eigenvectors with maximum eigenvalues contain most of information (energy) of the image vector.

Thus, to retrieve the image, it is sufficient to use only the coefficients for the eigenvectors with the largest eigenvalues, because the coefficients for the vectors with smaller eigenvalues contain a small amount of information. The amount of computation can be reduced if the decisions are made in a lower dimensional subspace, by neglecting eigenvectors with smaller eigenvalues and the corresponding coefficients in the K-L decomposition, i.e. by compressing some information and by partly discarding some information. Note that unlike this method of compression, our method doesn't lead to the loss of information and is invertible.

The common feature of our method and the rapidly developing area of implementing associative memory in neural-like structures is the use of attractors as objects, corresponding to the stored images. Our method has the advantage that the algorithm of our map design is sufficiently simple, and is much less computation intensive than the method of searching for connection strengths in neural networks. Furthermore, it possesses considerable capacity and the storage of information blocks is orthogonal by its nature, because no overlapping of information blocks are allowed. On the other hand, it should be noted that the extraction of an information block, unlike in neural networks, is sequential, and hence the number of iterations necessary to retrieve an information block is proportional to the length of the information block.

Finally, we consider the idea of using unstable limit cycles and intermittency to generate random access memory, where the trajectory visits all regions in phase space where information are stored.

Acknowledgements

This work is supported in part by the Office of Naval Research under grant N00014-89-J-1402 and the National Science Foundation under grant MIP-89-12639.

References

Aihara, K. [1990] "Chaotic Neural Networks", in Bifurcation phenomena in nonlinear systems and theory of dynamical systems, ed. Kawakami, H. (World Scientific) pp 143-161.

Auerbach, D et al. [1987] "Exploring chaotic motion through periodic orbits", Phys. Rev. Lett.**58** 2387-2389

Carpenter, G.A. [1989] "Neural network models for pattern recognition and associative memory", Neural networks **2**, 243-257.

Chua, L.O. & Kang, S.M. [1977] "Section-wise piecewise-linear functions: canonical representation, properties and applications", Proc. of the IEEE, **65**(6), 915-929.

Cvitanovich, P. [1988] "Invariant measurement of strange sets in terms of cycles", Phys. Rev. Lett., **61** 2729-2732

Dmitriev, A.S. [1991] "Storing and recognizing information in one dimensional dynamic systems", Radiotechnica i electronika, **36**(1), 101-108.

Dmitriev, A.S., Idanova, L. & Kuminov, A. [1991] "The simplest neural-like systems with chaos" in Proceedings of the international conference on fluctuations and noise 2/5 (Kyoto, Japan).

Dmitriev, A.S., Panas, A.I. & Starkov, S.O. [1991] "Storing and recognizing information based on stable cycles of one dimensional maps", Physics Letters, **155A**(8-9) 494-499.

Grebogi, C., & Ott, E. [1983] "Crises, sudden changes in chaotic attractors, and transient chaos", Physica **7**D, 181-200.

Grebogi, C., Ott, E. & Yorke, J.A. [1986] "Critical exponent of chaotic transients in nonlinear dynamic systems", Physical Review Letters **57**, 1284-1287.

Grossberg, S. [1988] "Nonlinear neural networks: Principles, mechanisms and architectures", Neural Networks **1**, 17-61.

Haken, H. [1987] "Synergetic computers for pattern recognition and associative memory", in Computational systems - natural and artificial, ed. Haken, H. (Springer - Verlag, Berlin) pp. 2-22.

Haken, H. [1988] "Synergetic in pattern recognition and associative action", in Neural and synergetic computers, ed. Haken,H, (Springer-Verlag, Berlin), pp. 2-15.

Haken, H. [1988] Information and self-organization. A macroscopic approach to complex systems (Springer - Verlag, Berlin, Heidelberg).

Hopfield, J.J. [1982] "Neural networks and physical systems with emergent collective computational abilities", Proc. Nat. Acad. Sci. (USA) **79**, 2554-2558.

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

Kohonen, T. [1980] Content-addressable memory (Springer, Berlin).

Nicolis, J.S. [1986] "Dynamics of hierarchical systems: An evolutionary approach", (Springer-Verlag).

Nicolis, J.S. [1991] Chaos and information processing: A heuristic outline (World Scientific)

Ott, E., Grebogi, C. & Yorke, J.A. [1990] "Controlling chaos", Physical Review Letters **64,** 1196-1199.

Procaccia, I. [1988] "The organization of chaos by periodic orbits: Topological universality of complex systems.", in Springer proceedings in physics, **32,** Universalities in condensed matter, ed. Jullien, R. (Springer, Berlin) 213-215

Sharkovsky, A.N., Maisyrenko Yu.L. & Romanenko E.Yu. [1986] "Difference equations and their applications" (Naukova Dumka, Kiev) (in Russian).

Sharkovsky, A.N., Kolyada,S.F., Sivak, A.G. & Fedorenko, V.V. [1989] "Dynamics of one-dimensional maps" (Naukova Dumka, Kiev) (in Russian)

Skarda, C.A. & Freeman, W.J. [1987] "How brains make chaos in order to make sense of the world", Behav. Brain Science **10**, 161-195

Wiegrinck, W. & Tennekes, H. [1990] "On the information flow for one-dimensional map", Phys. Lett. **A144** 145-152

FIGURE CAPTIONS

Fig.1 1-D map design for the information block **big**: (a) correspondence between the alphabet and the segments on the unit interval; x(1), x(2) and x(3) corresponds to x_b, x_i, x_g (b) the cycle coding the word **big** (c) projection of information districts onto the x_m-axis (d) the final map with the stable limit cycle.

Fig.2 1-D map for storing the information block **big** using 2 levels. The diamonds show the positions of cycle values when storing the same information block using only 1 level (fig. 1b).

Fig.3 1-D map corresponding to the Bernoulli shift on **N** symbols.

Fig.4 Schematic picture of a map for storing a string of binary digits.

Fig.5 Schematic graph of map iterations versus the level of noise introduced. For ε less than ε_c the trajectory converges to a limit cycle. For ε greater than ε_c, the trajectory behaves chaotically.

Fig.6 Representation of the letter **R** by adding the "label" symbol to a string of black and white elements.

Fig.7 A set of letters stored on 1-D map. The information block is created by appending a label symbol at the beginning and reading it line by line.

Fig.8 1-D map for storing the letters in figure 7 as stable limit cycles. The cycle corresponding to the letter **R** is shown.

Fig.9 An arbitrary continuous piecewise-linear function defined on an interval.

Fig.10 Use of coding on an 8×8 picture of black squares. The first five elements of the new alphabet are shown as $\mathbf{w},\mathbf{b},\mathbf{b_2},\mathbf{b_4},\mathbf{b_8}$.

Fig.11 Images of a kitten stored on a one-dimensional map. Seven colors are used,

with 3 levels of storage. The array size of the pictures are (a) $16\times24 = 384$; (b) $32\times48 = 1536$; (c) $64\times96 = 6144$.

Fig.12 Transient process from the initial condition to a stable limit cycle. The picture stored is shown in **Fig. 11b.** (a) typical trajectory corresponding to transient chaos; (b) the trajectory right before converging to the stable limit cycle.

Fig.13 Simultaneous storage of two color pictures (elephant and kitten) using 3 levels. Transient process from initial conditions to one of two limit cycles: (a) typical picture corresponding to transient chaos and includes color elements of both pictures; (b) the picture just before converging to the stable limit cycle corresponding to the elephant; (c) the stable limit cycle corresponding to the image of the elephant.

Fig.14 1-D map (shown in blue) corresponding to the simultaneous storage of two color pictures (elephant and kitten). Shown in yellow: (a) cycle corresponded to elephant; (b) cycle corresponded to kitten.

Fig.15 Trajectory of 1-D map for storing the information blocks **97583** and **12345**. (a) the limit cycle corresponding to the block **97583**. (b) the limit cycle corresponding to the block **12345**. (c) intermittency occurring between the two cycles.

Fig.16 Typical time series of pictures exhibiting intermittency. The words **ORDER**, **FROM**, and **CHAOS** appears repeatedly. The pictures are taken at various intervals and appears chronologically from left to right.

Fig.17 Times series showing the retrieval of a selected information block from the random appearance of stored information blocks. (a) the information block **97583** is retrieved. (b) the information block **12345** is retrieved.

TABLE CAPTIONS

Table 1 Table of alphabet lengths and cycle periods corresponding to different storage levels. AL = alphabet length, CP = cycle period.

Number of storage levels	Number of cells					
	16×24		32×48		64×96	
	AL	CP	AL	CP	AL	CP
2	50	80	109	206	>190	-
3	30	102	71	240	162	568
4	34	102	55	300	-	-
5	24	136	-	-	-	-

Table 1

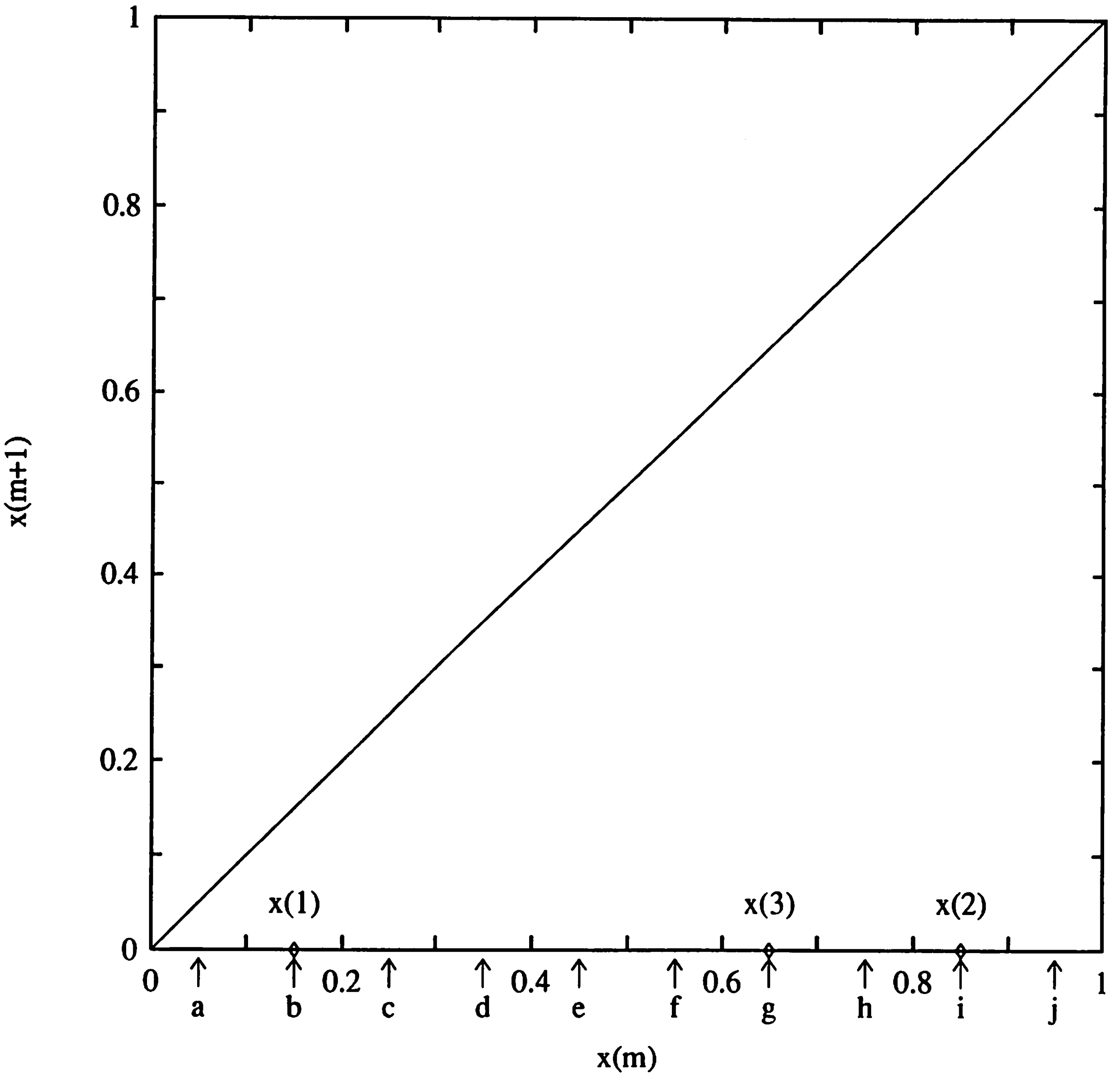


Figure 1a

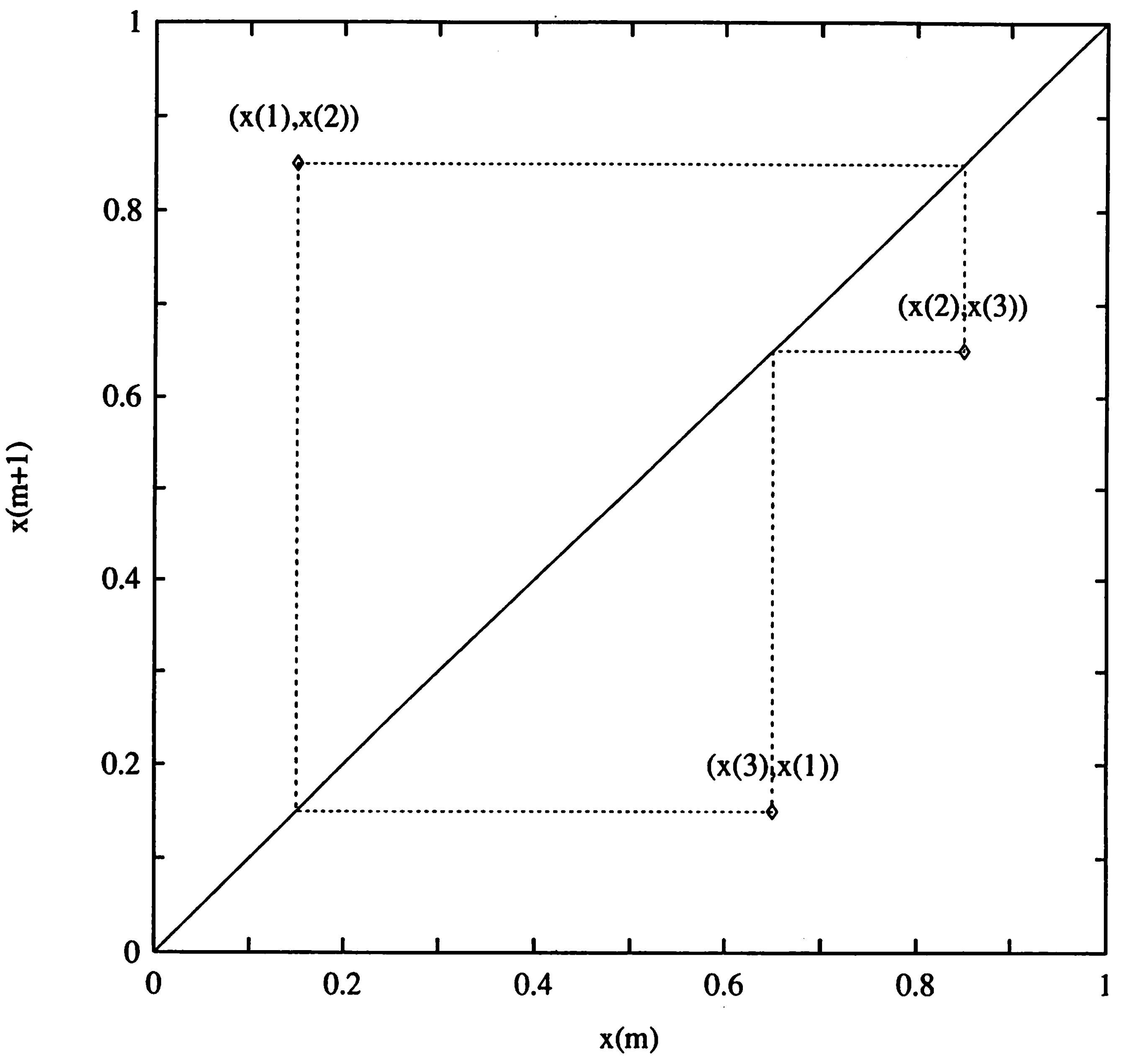


Figure 1b

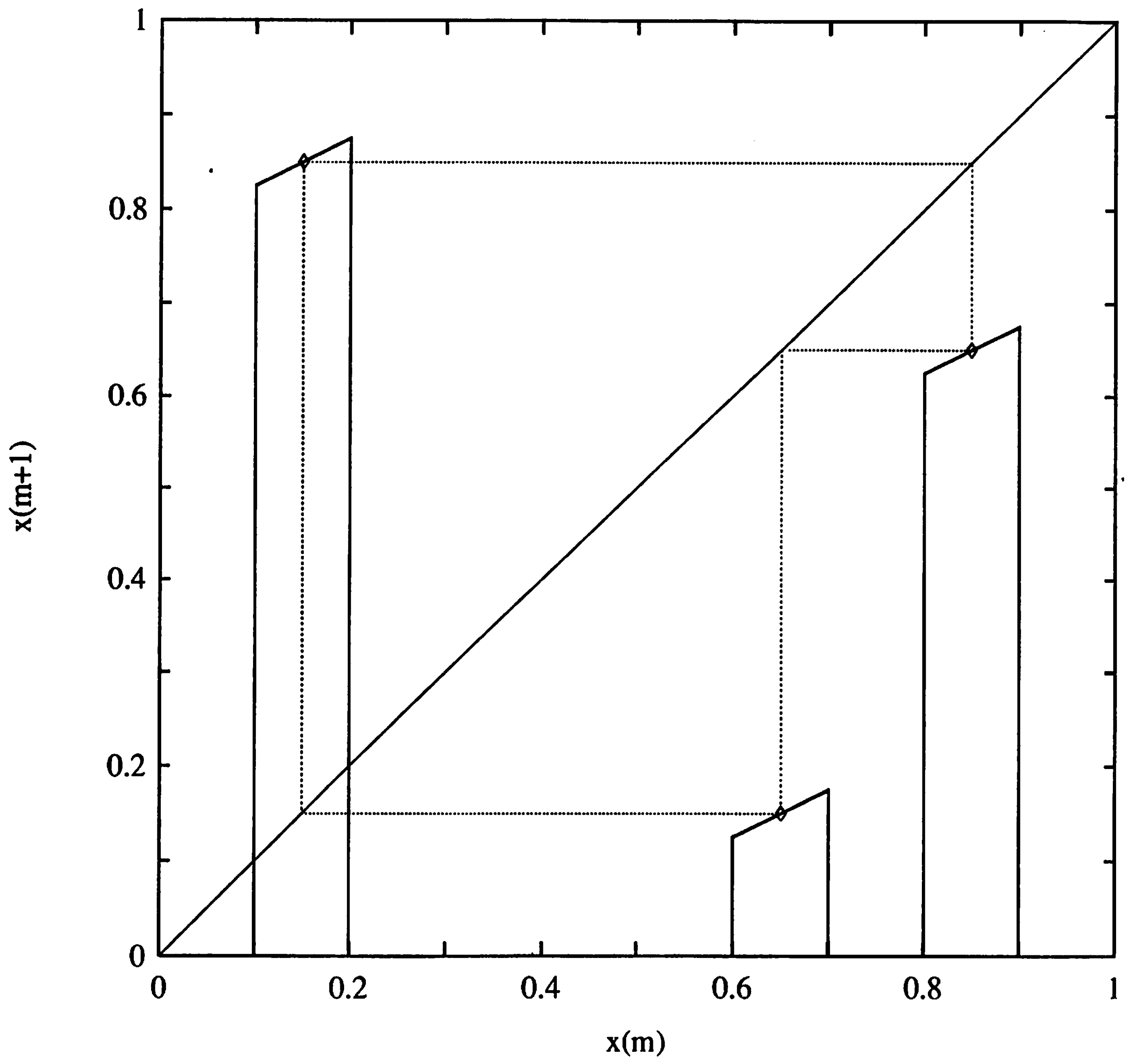


Figure 1c

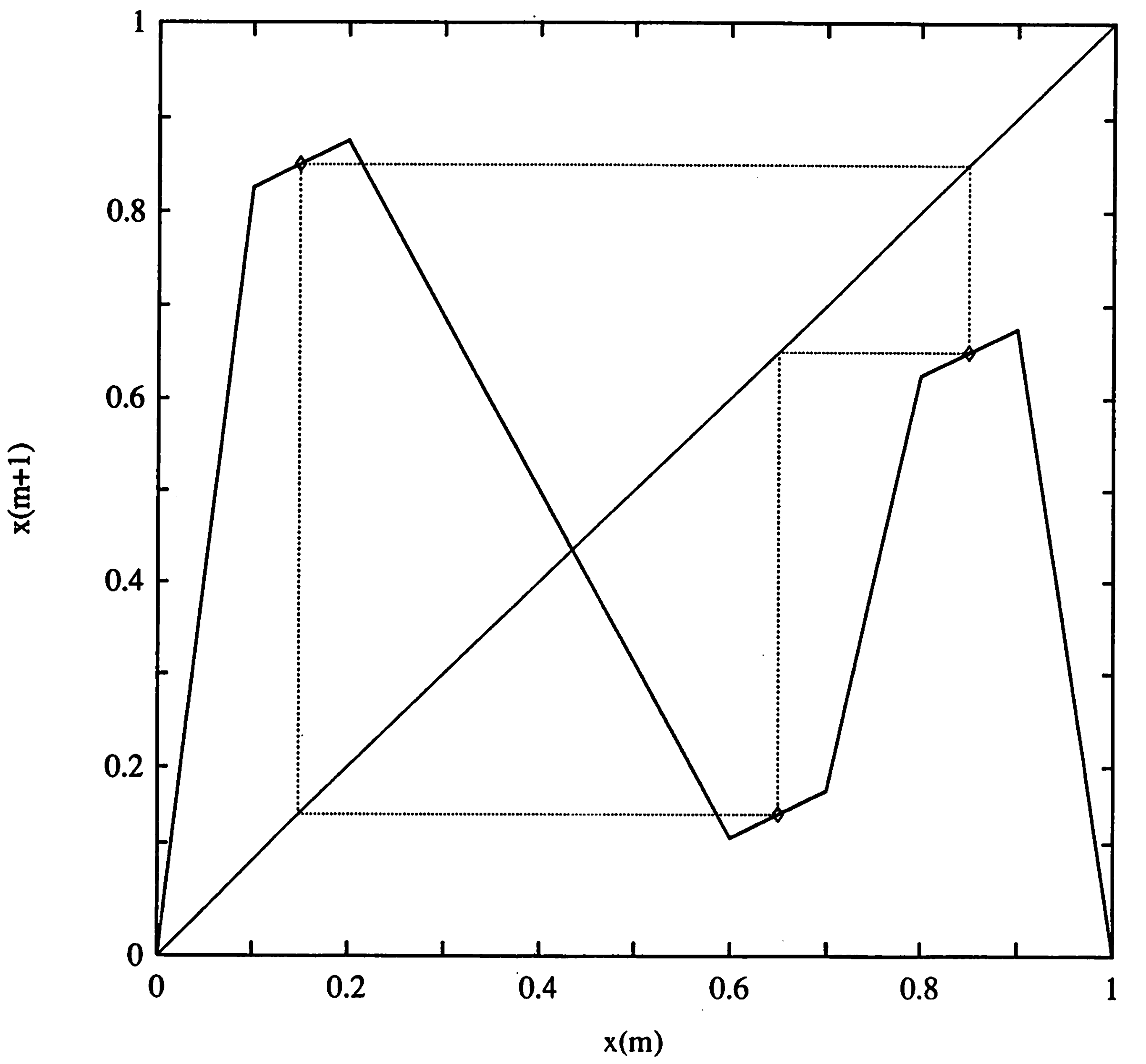


Figure 1d

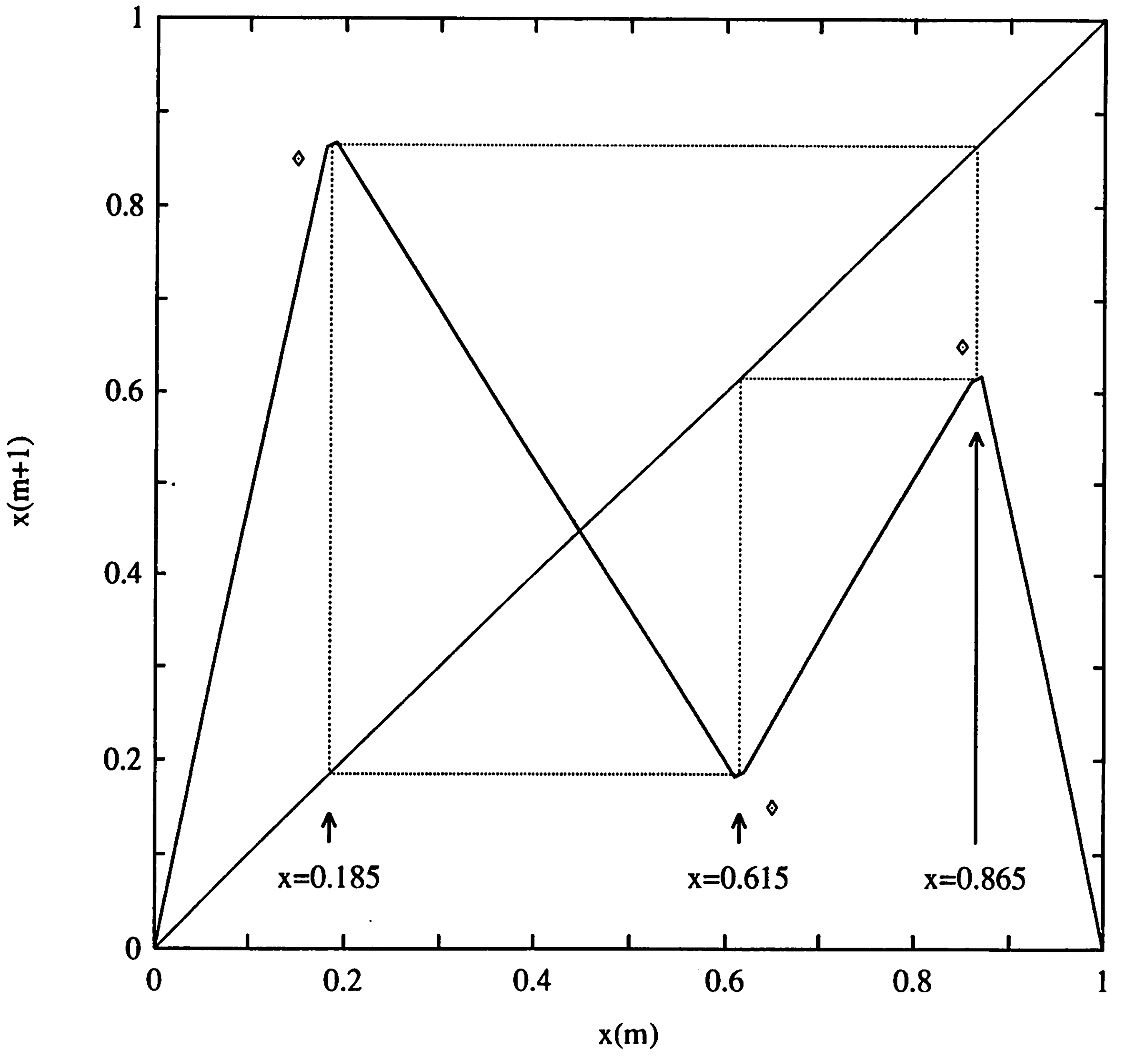


Figure 2

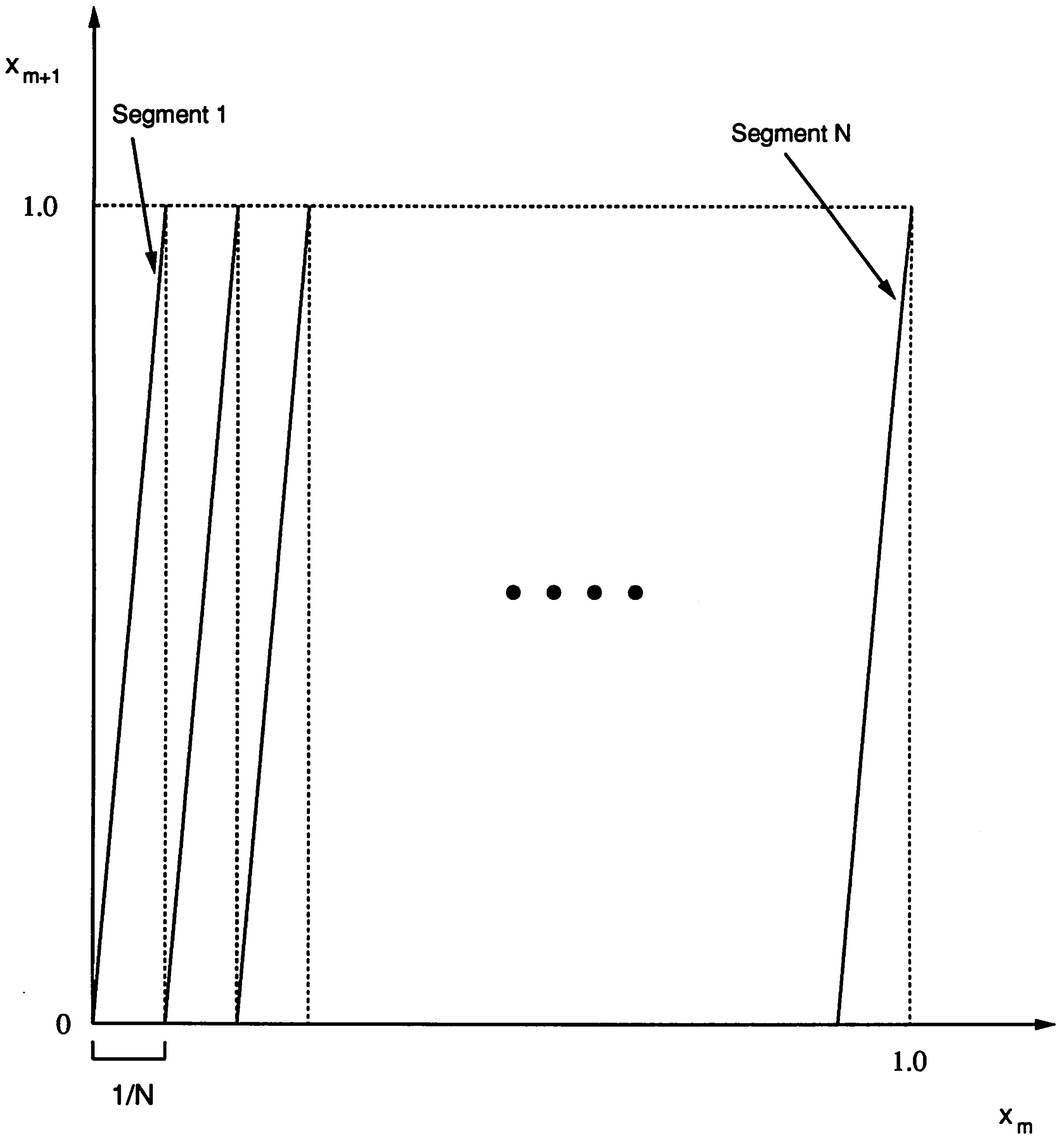


Figure 3

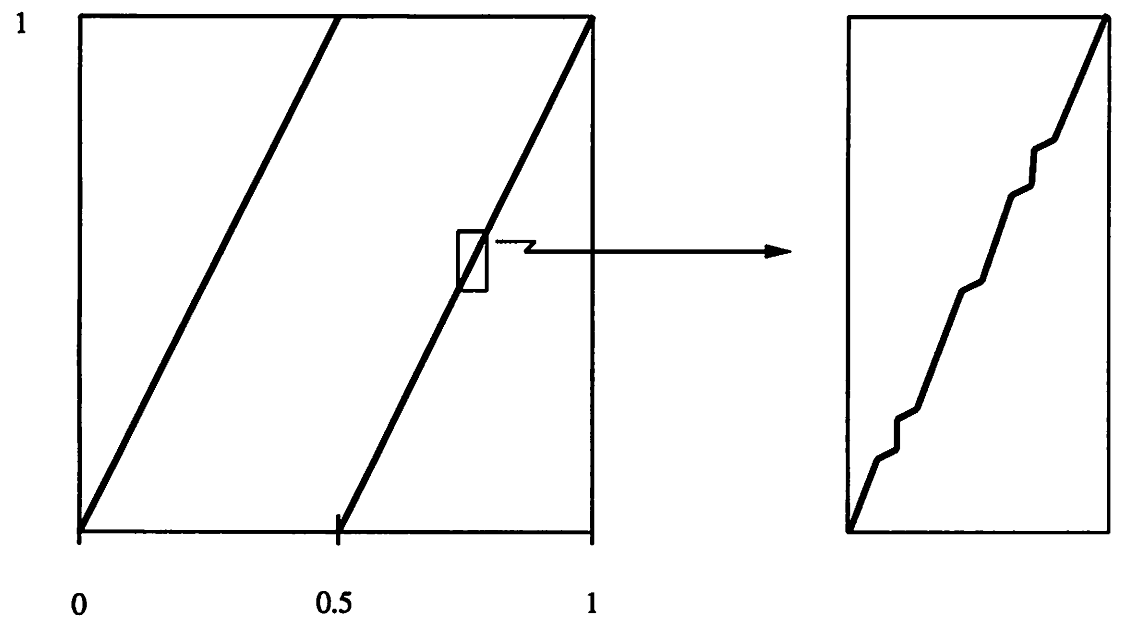


Figure 4

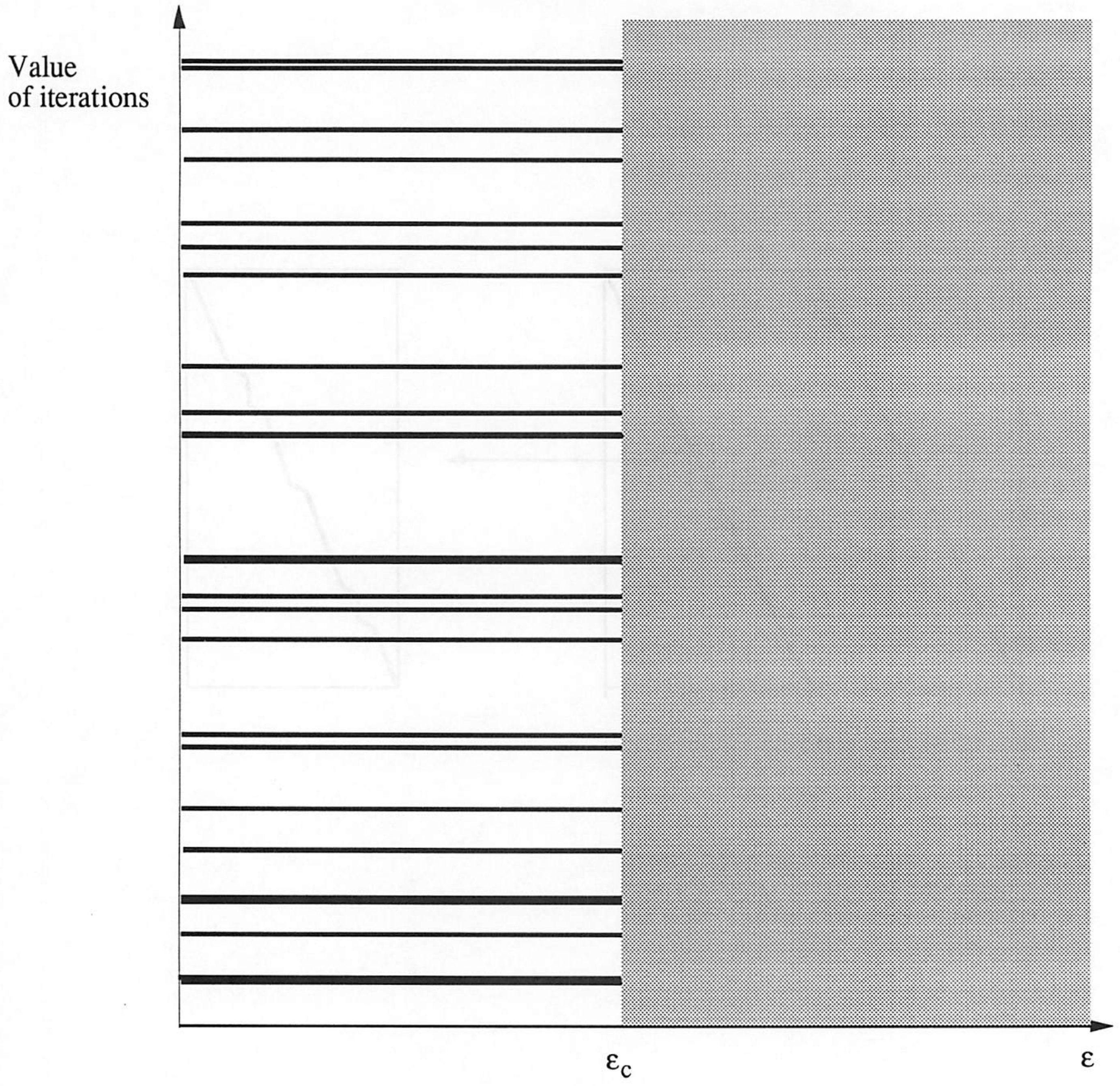
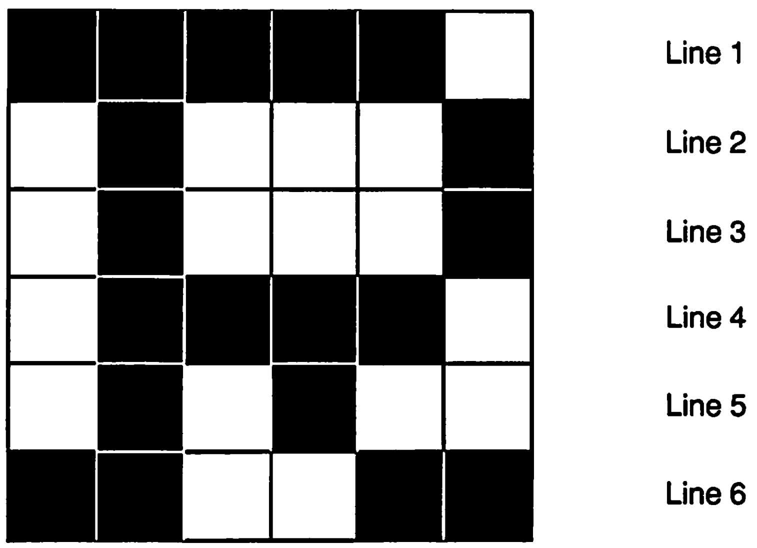


Figure 5



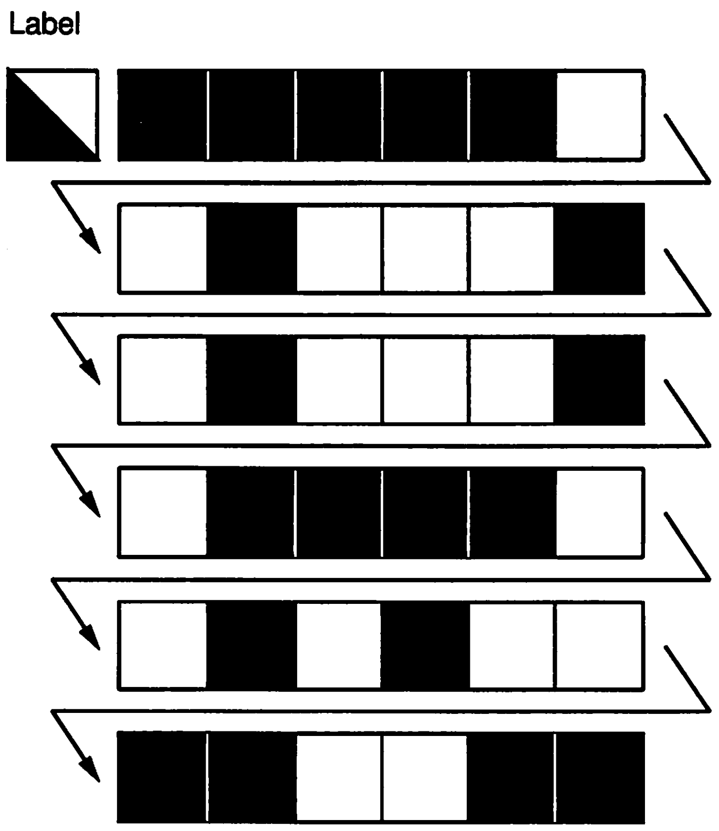


Figure 6

Stored Images

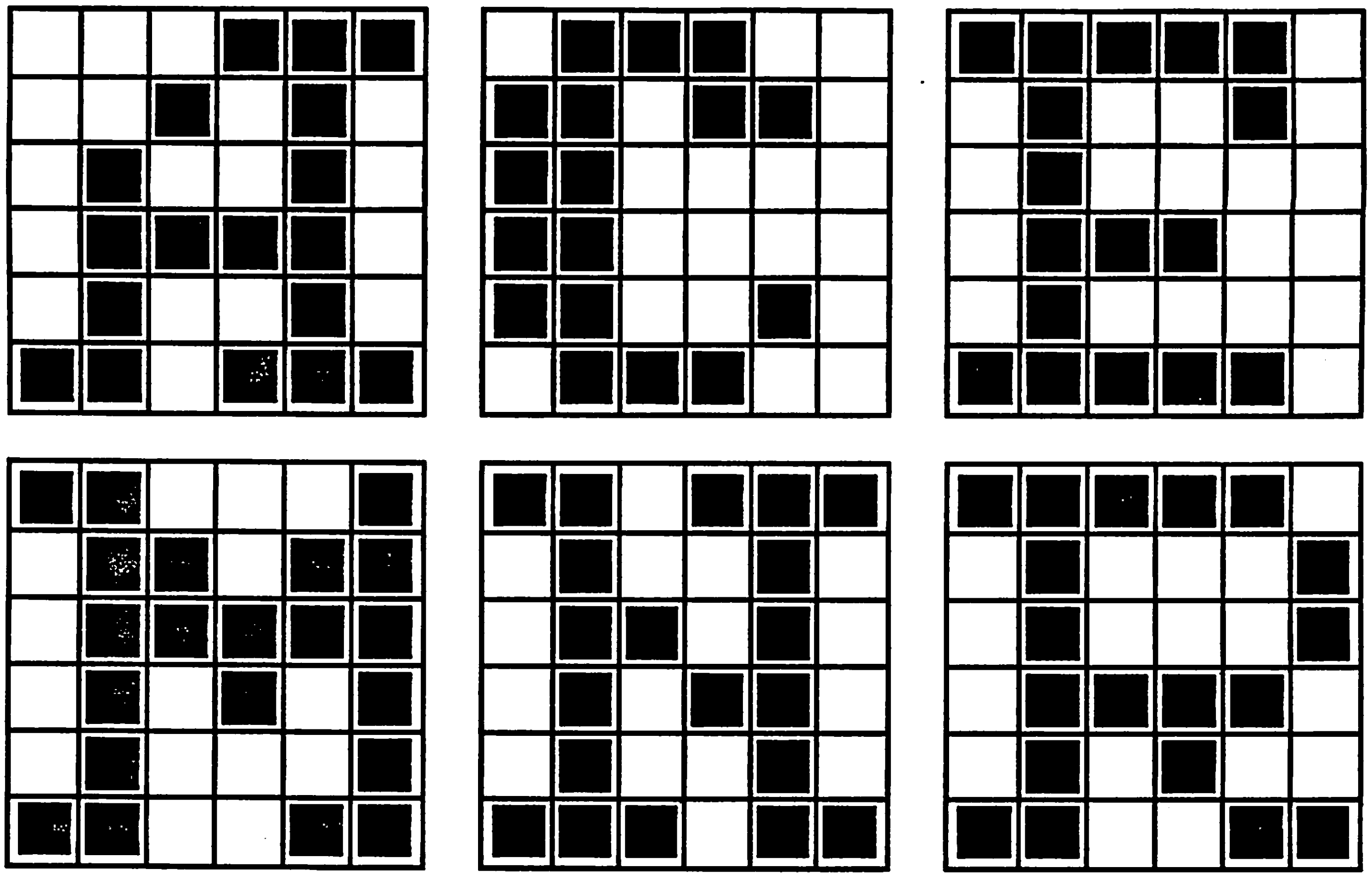


Figure 7

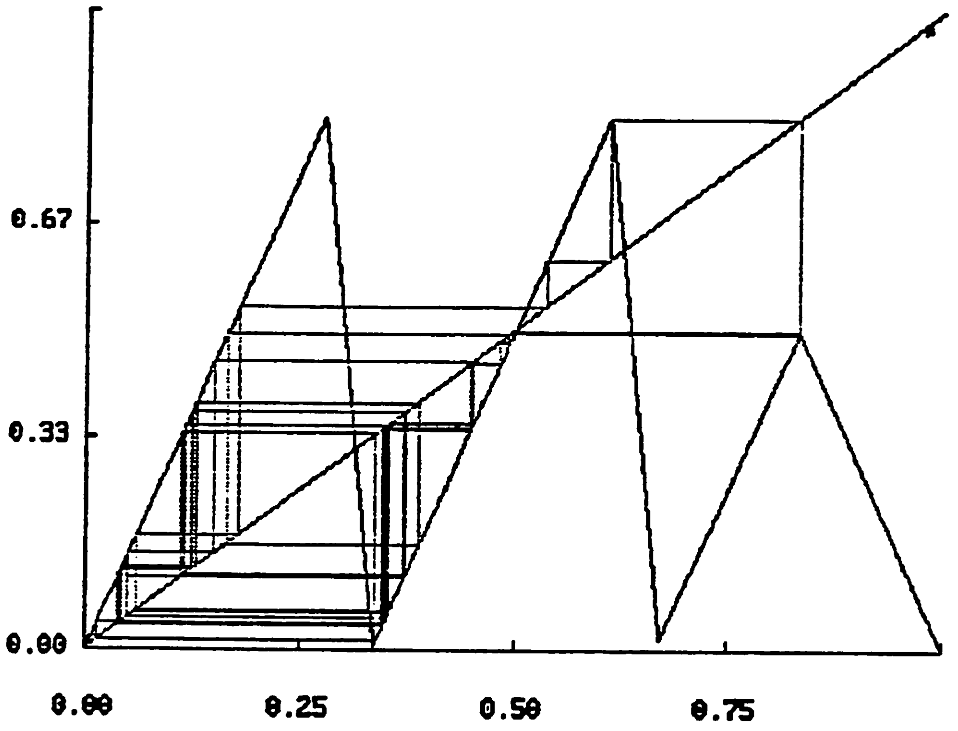


Figure 8

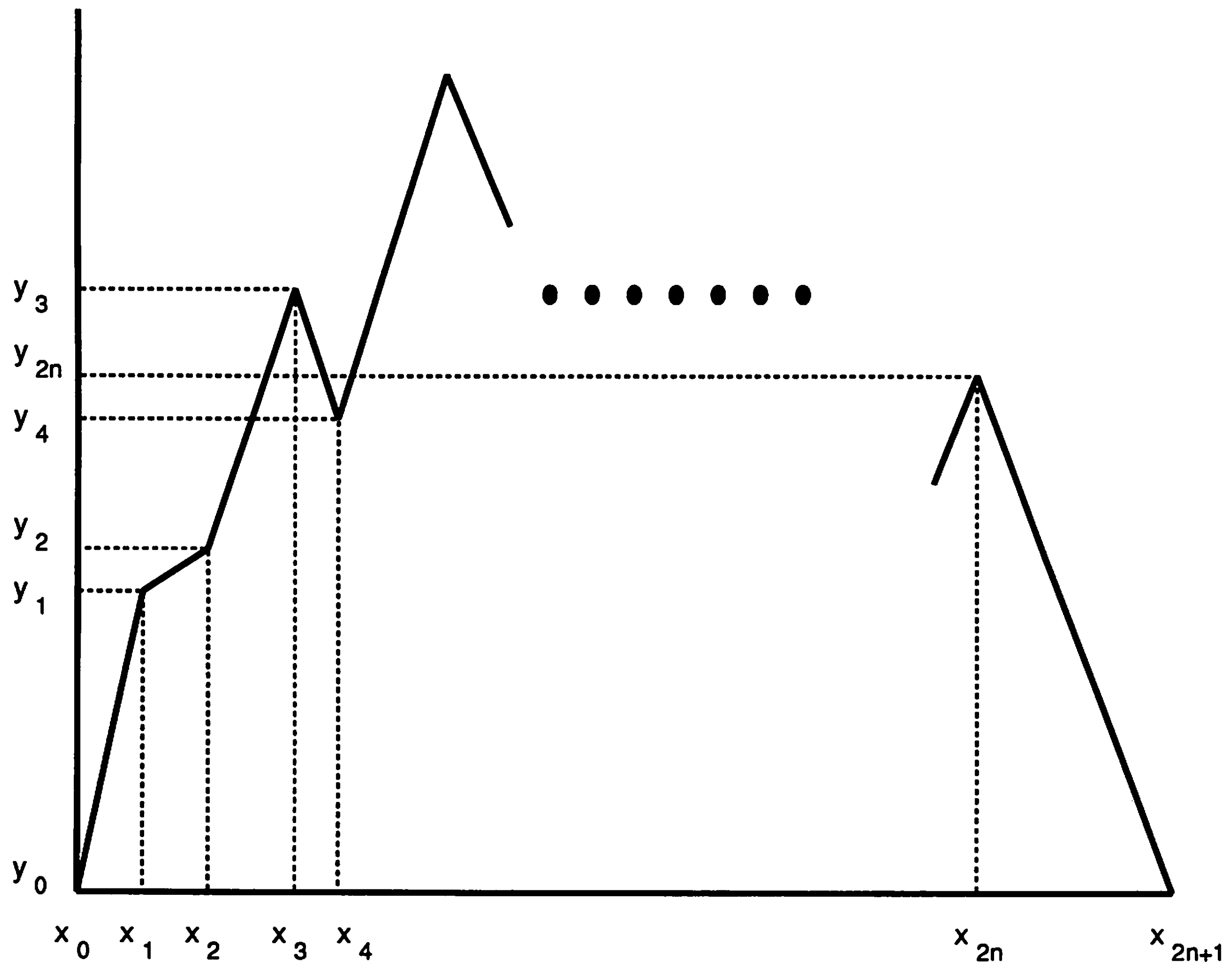
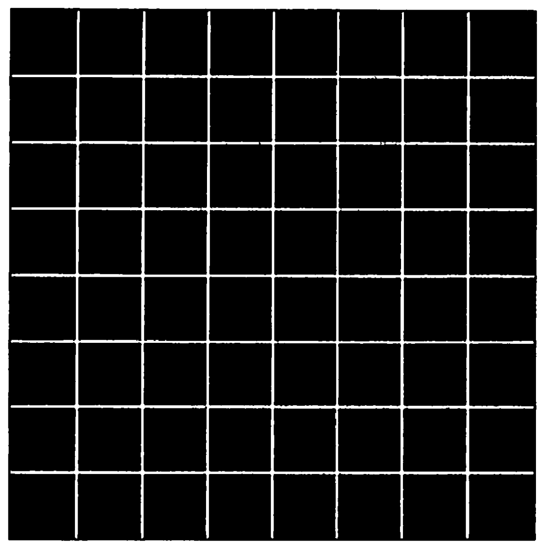


Figure 9



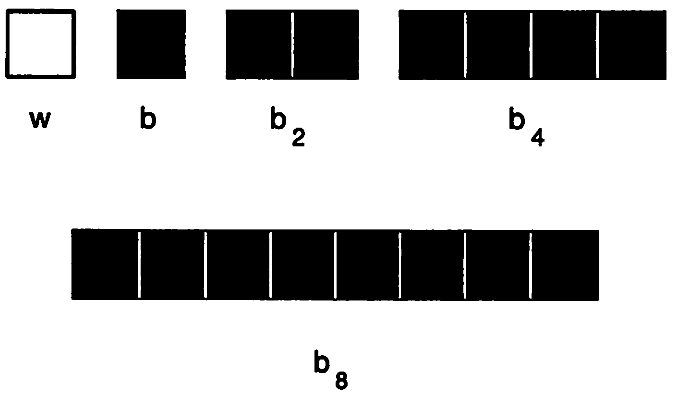


Figure 10

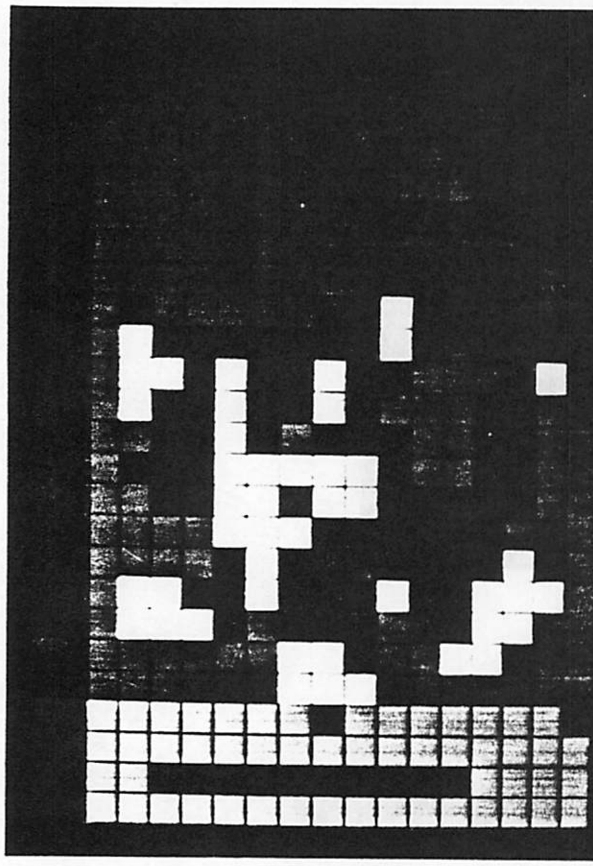


Figure 11a

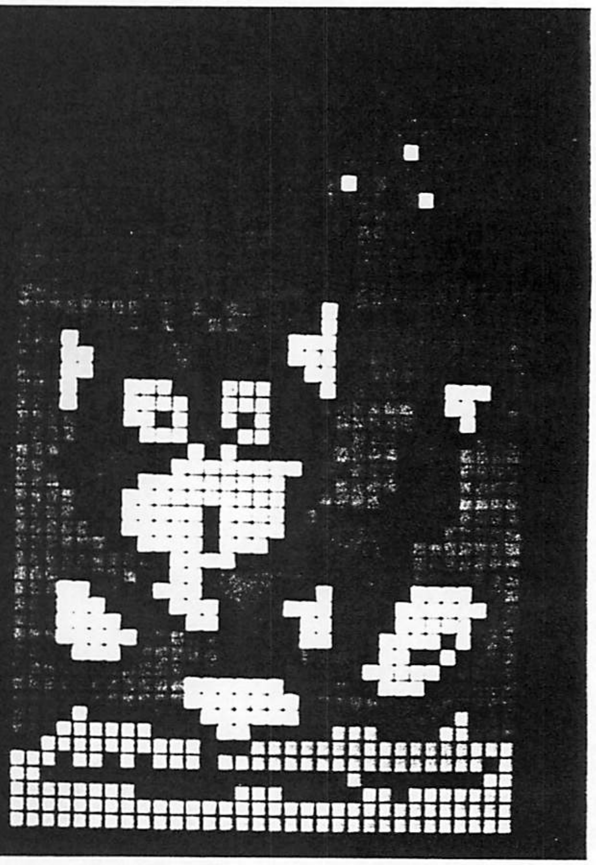


Figure 11b

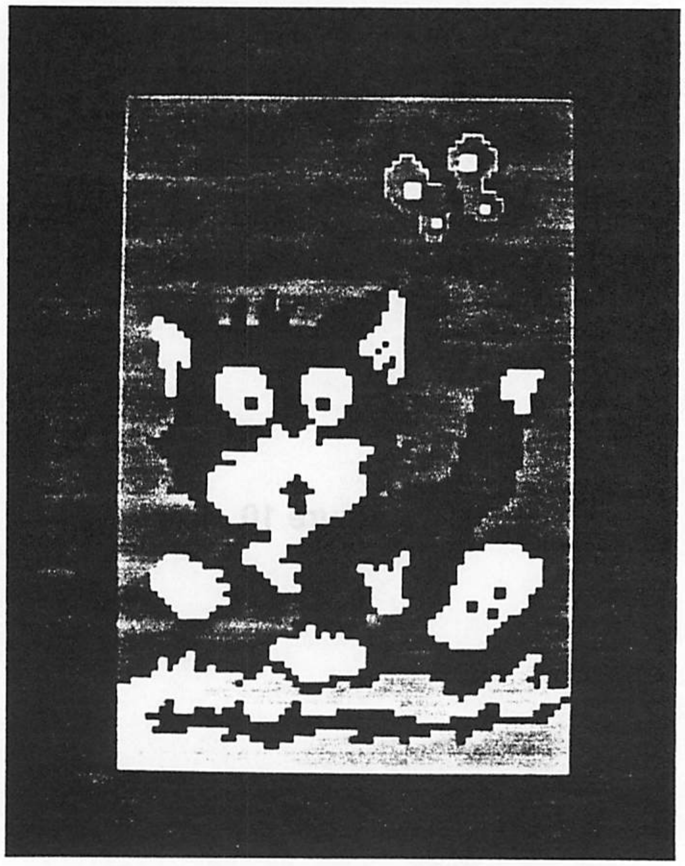


Figure 11c

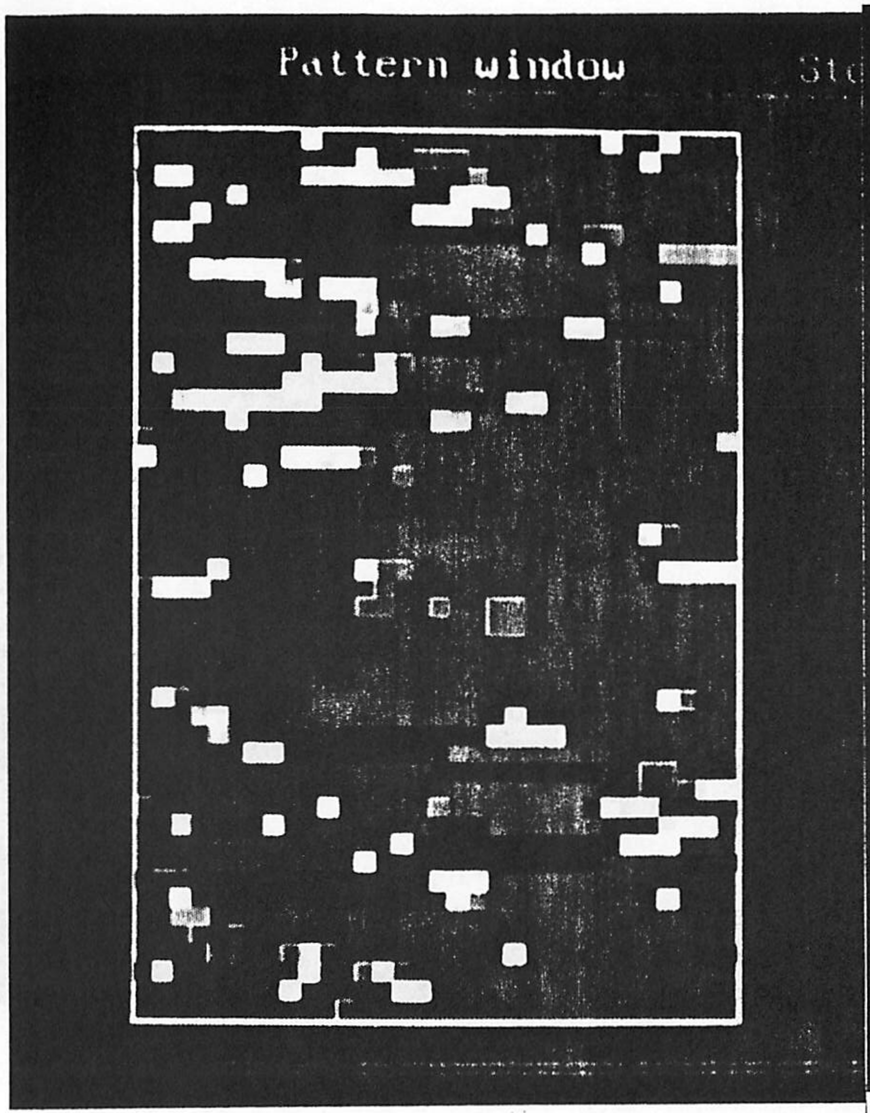


Figure 12a

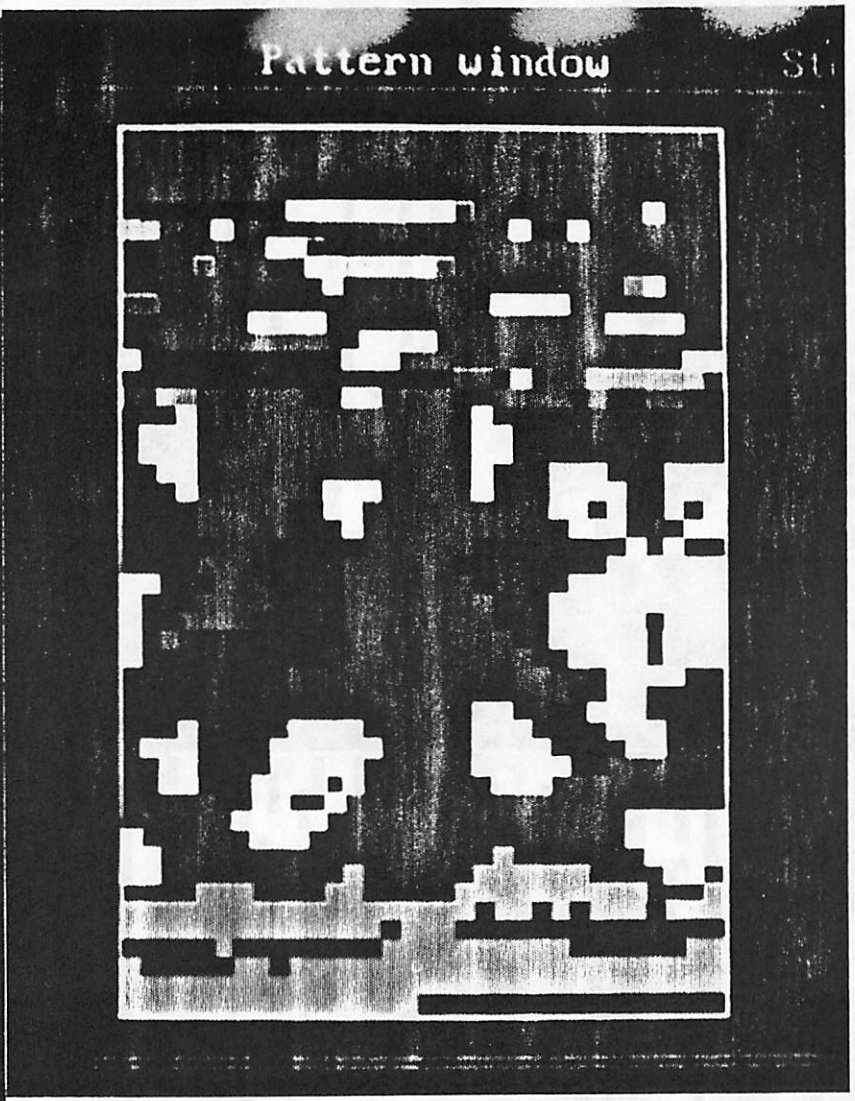
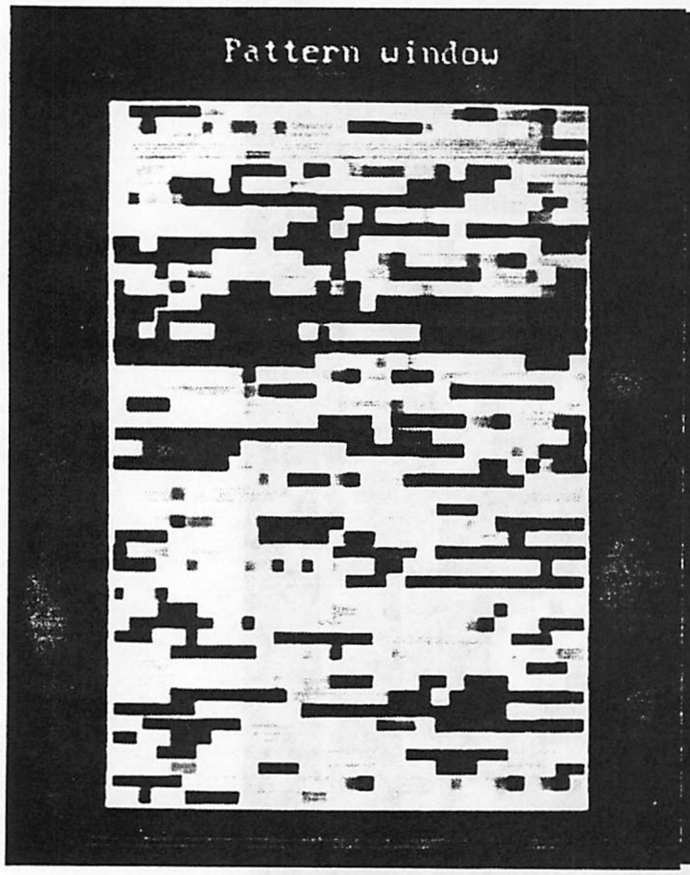


Figure 12b



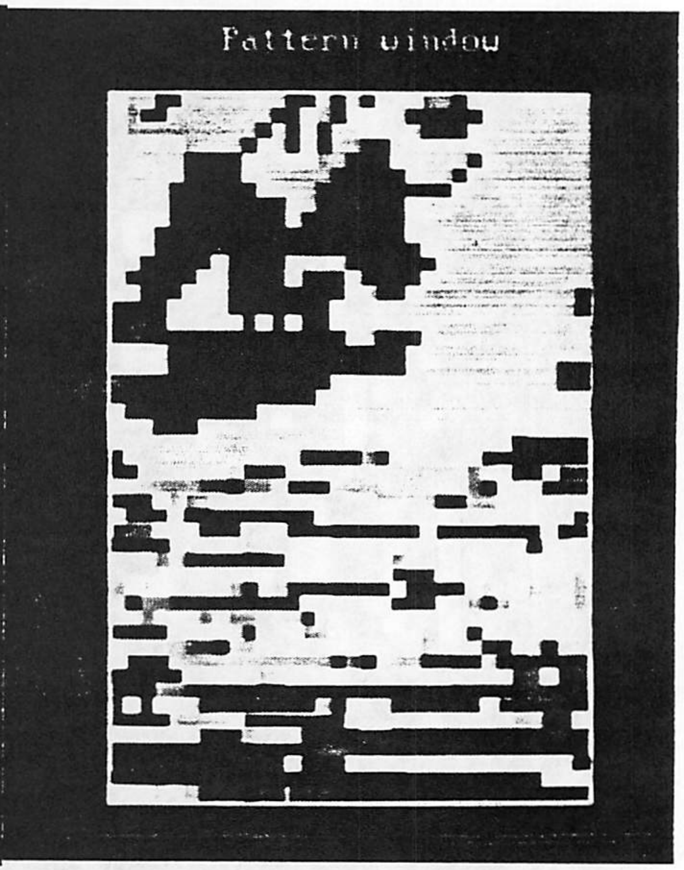


Figure 13a Figure 13b

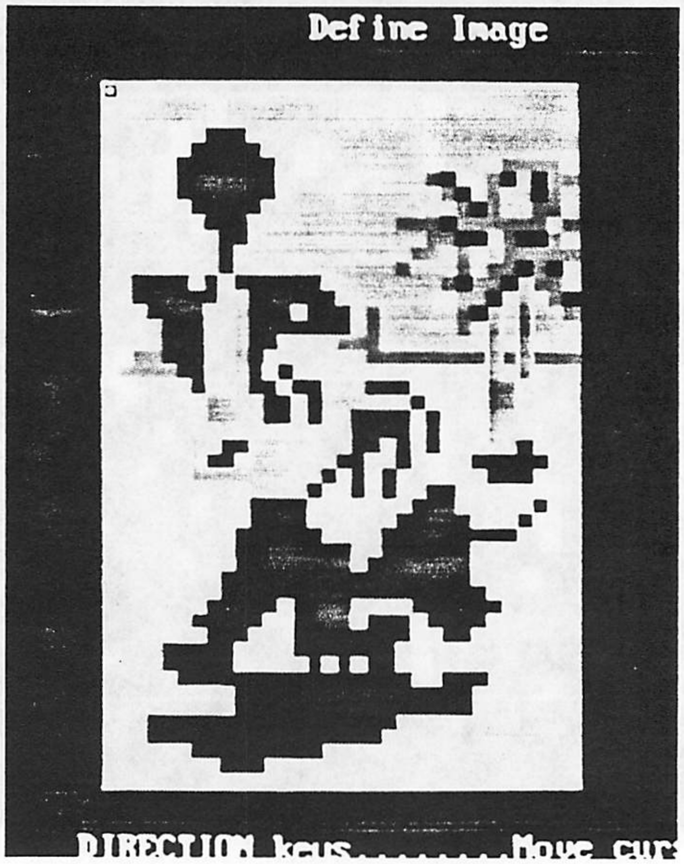


Figure 13c

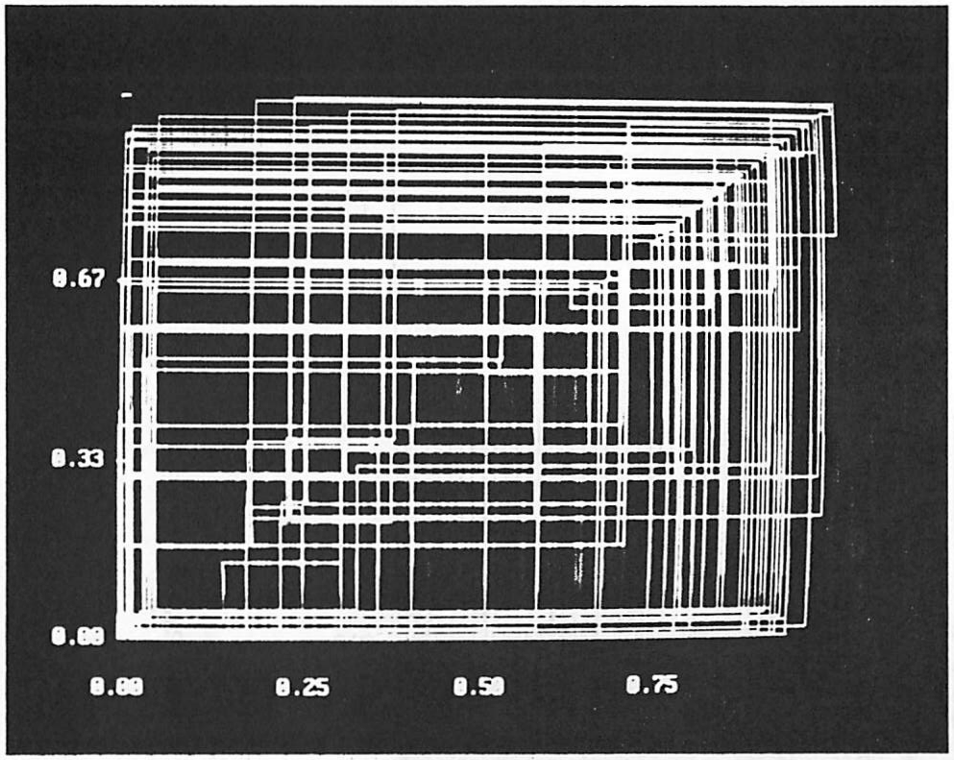


Figure 14a

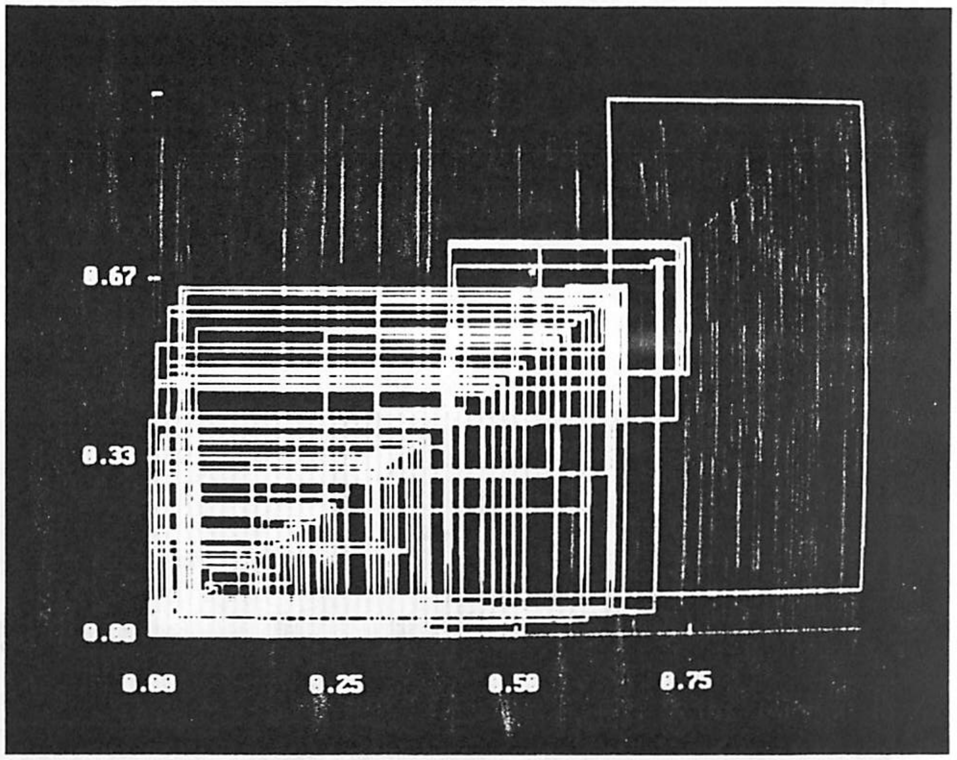


Figure 14b

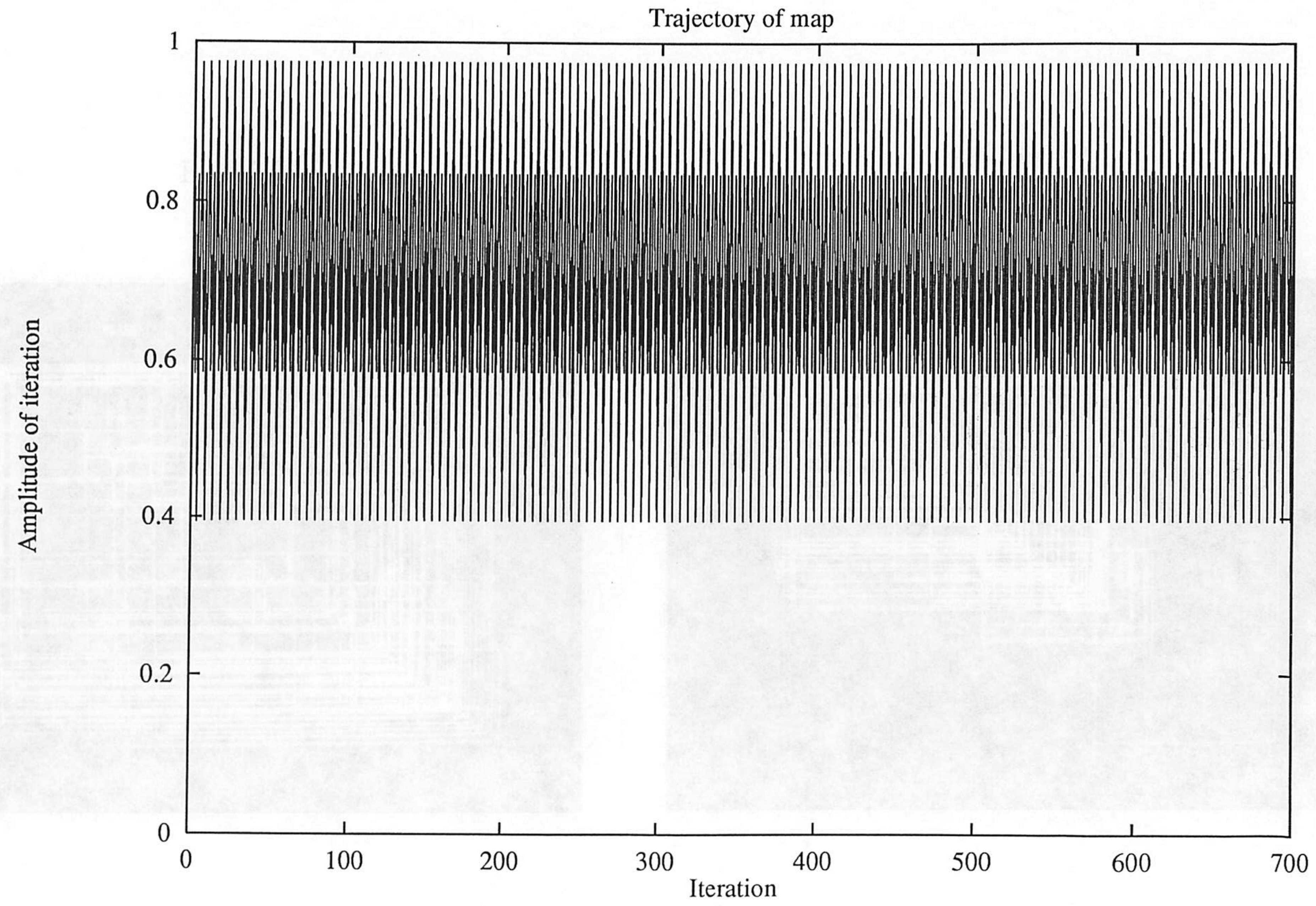

Trajectory of map
Amplitude of iteration
Iteration
0
0.2
0.4
0.6
0.8
1
0
100
200
300
400
500
600
700

Figure 15a

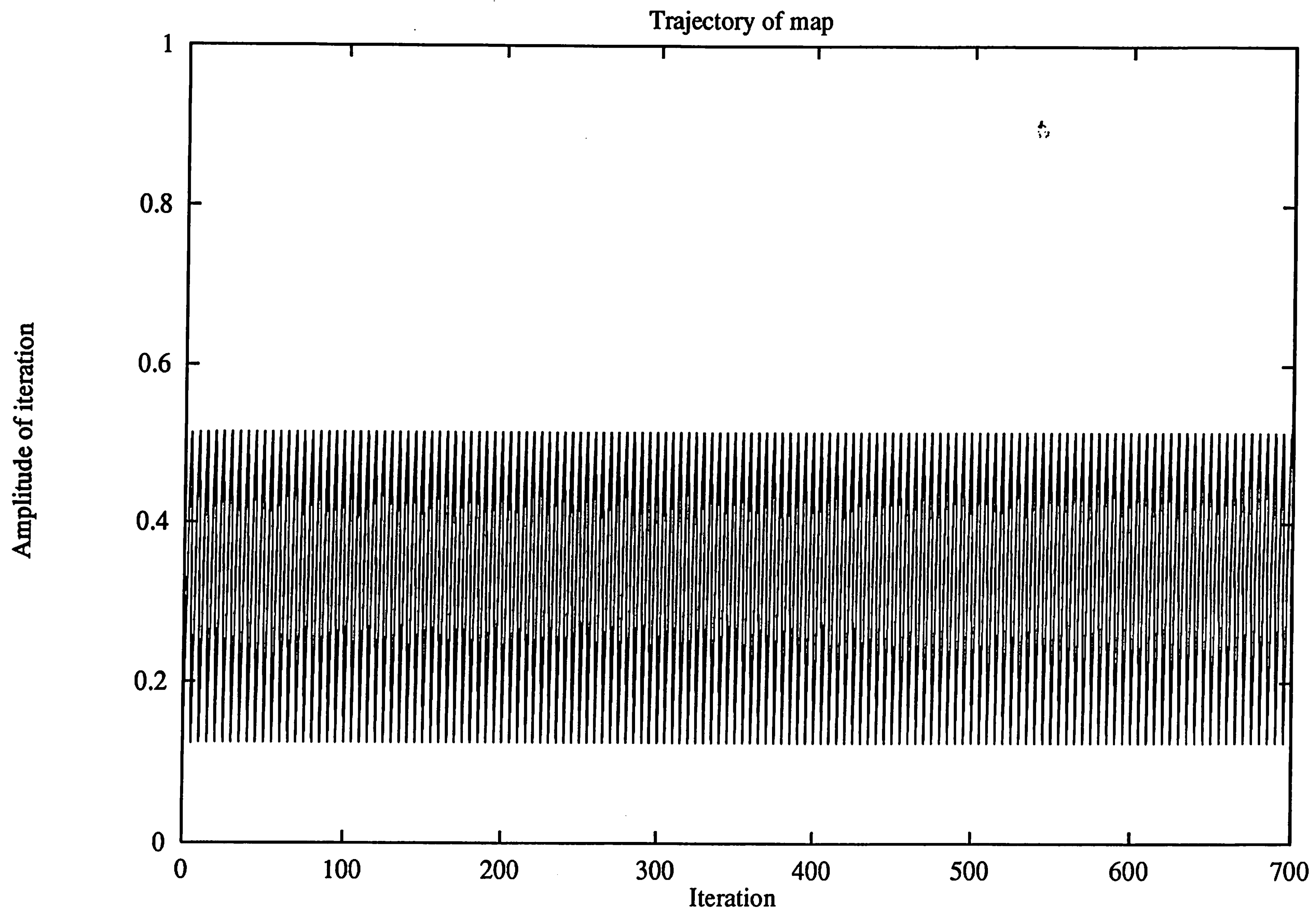

Trajectory of map
Amplitude of iteration
Iteration
0
0.2
0.4
0.6
0.8
1
0
100
200
300
400
500
600
700

Figure 15b

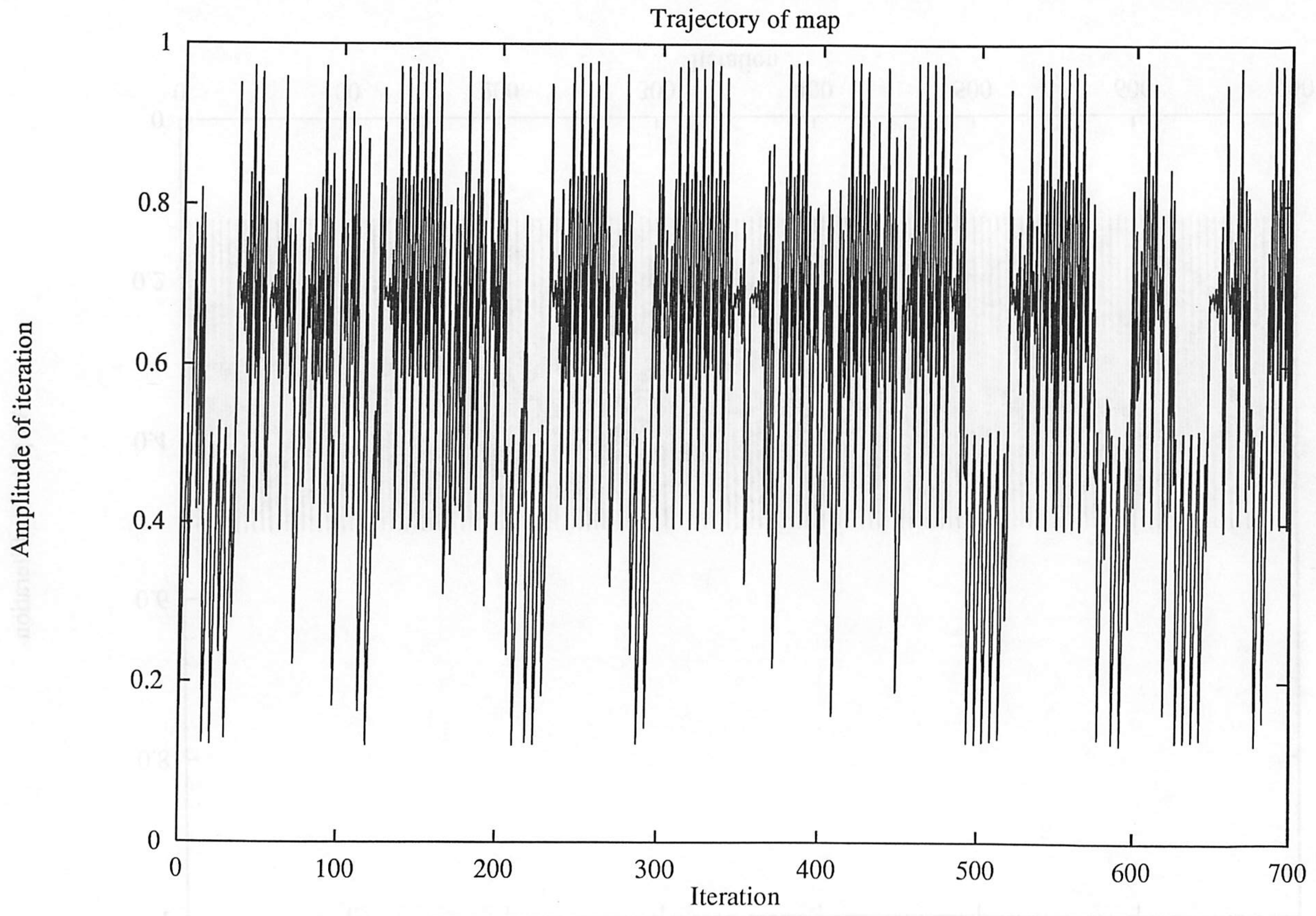

Trajectory of map
Amplitude of iteration
Iteration
0
0.2
0.4
0.6
0.8
1
0
100
200
300
400
500
600
700

Figure 15c

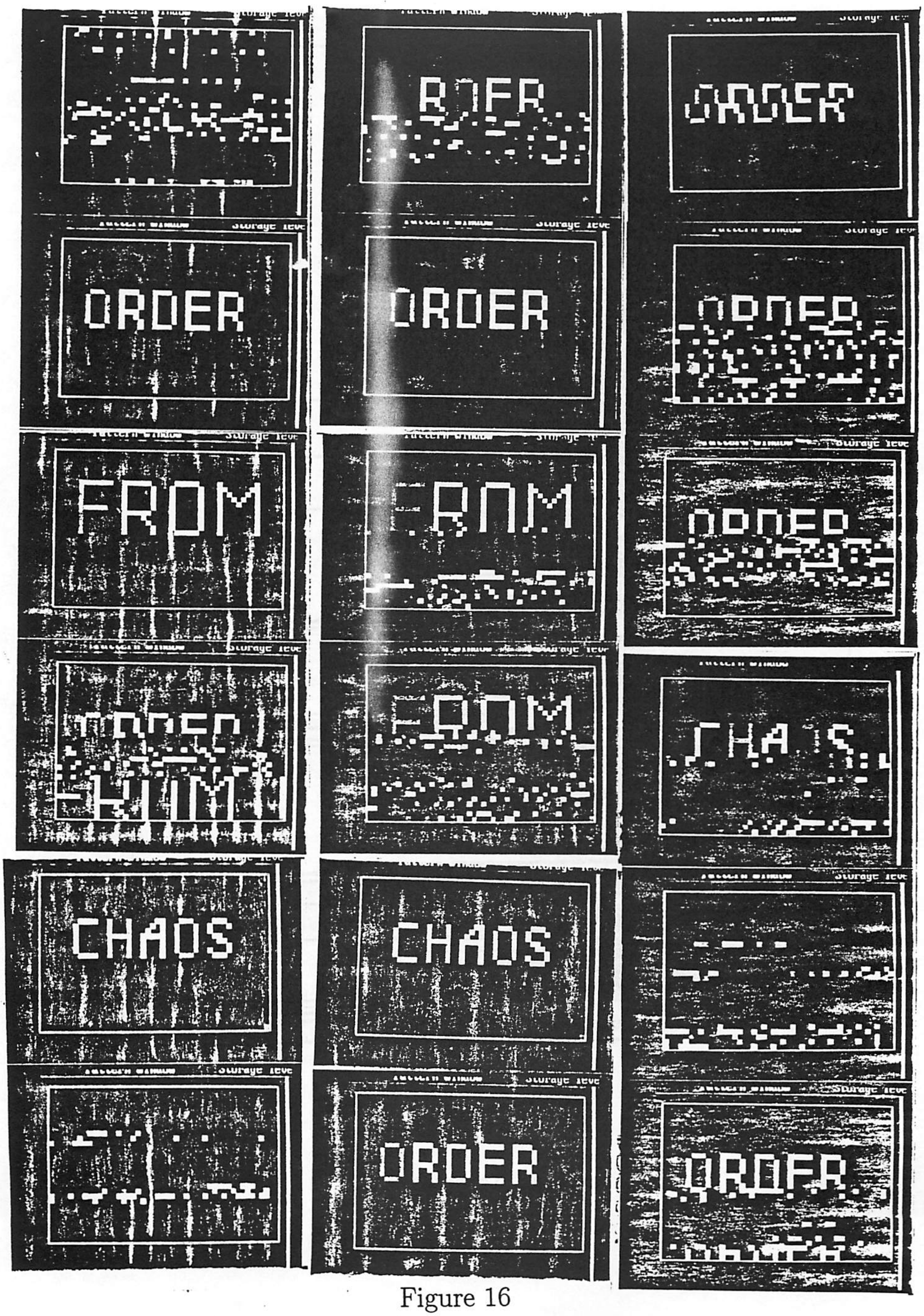


Figure 16

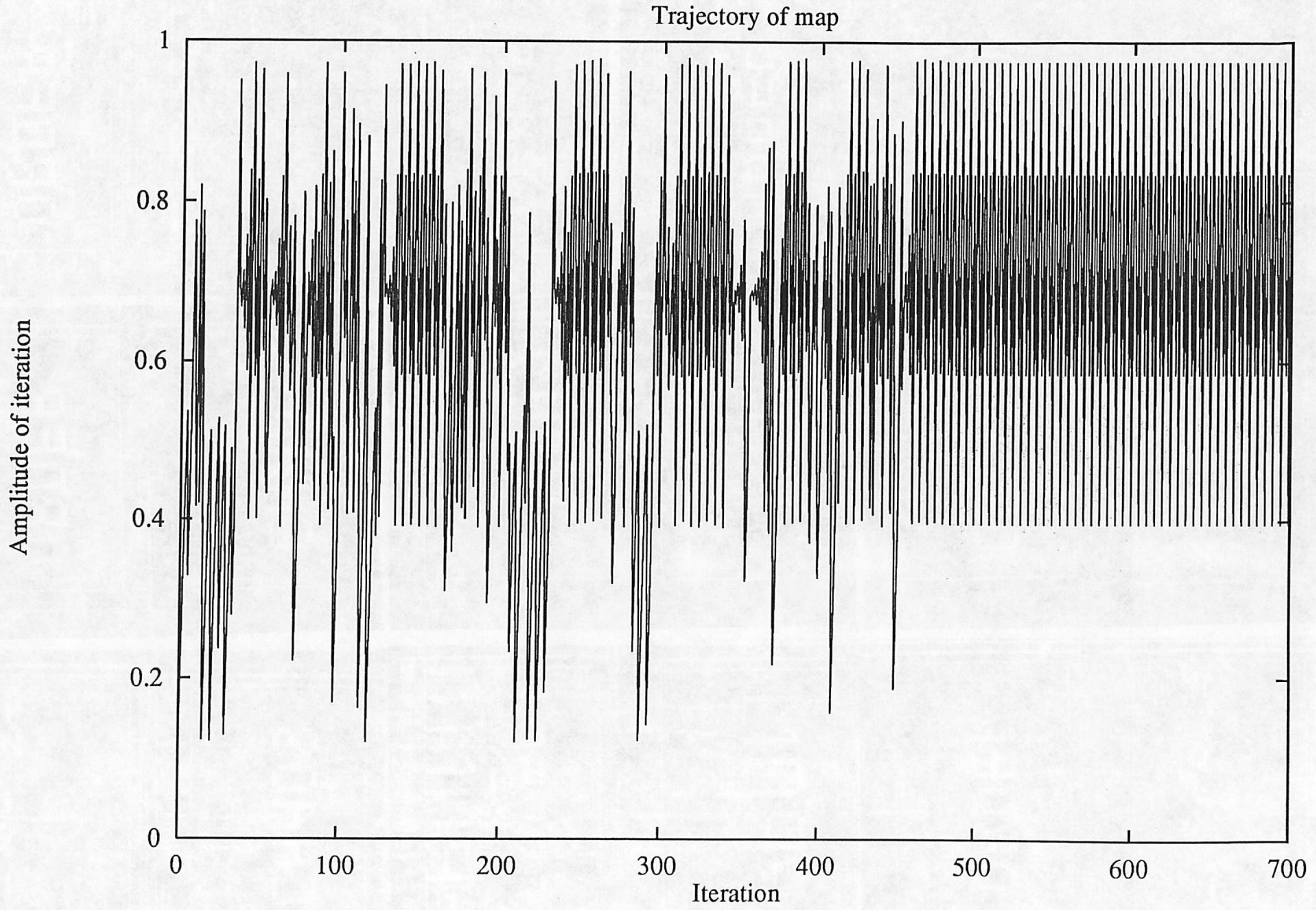

Trajectory of map
Amplitude of iteration
Iteration
0
0.2
0.4
0.6
0.8
1
0
100
200
300
400
500
600
700

Figure 17a

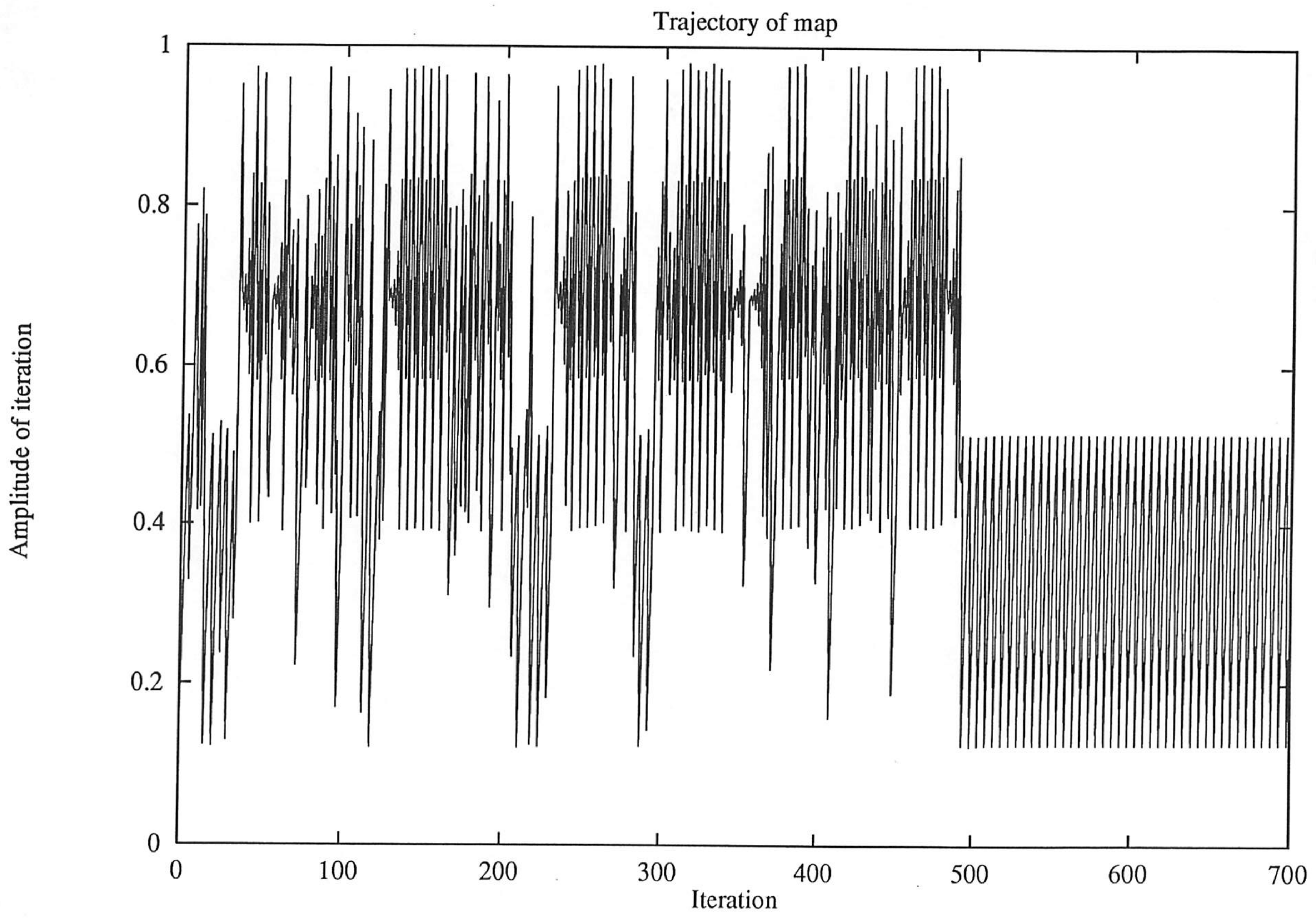

Trajectory of map
Amplitude of iteration
0
0.2
0.4
0.6
0.8
1
0
100
200
300
400
500
600
700
Iteration

Figure 17b