PERGAMON

Chaos, Solitons and Fractals 14 (2002) 57–70

CHAOS
SOLITONS & FRACTALS

www.elsevier.com/locate/chaos

Chaotic maps and pattern recognition – the XOR problem

Alan Rogers [a,*], John G. Keating [b], Robert Shorten [a], Daniel M. Heffernan [c,d]

[a] *Department of Electronic Engineering, National University of Ireland, Maynooth, Co. Kildare, Ireland*
[b] *Department of Computer Science, National University of Ireland, Maynooth, Co. Kildare, Ireland*
[c] *Department of Mathematical Physics, National University of Ireland, Maynooth, Co. Kildare, Ireland*
[d] *School of Theoretical Physics, Dublin Institute for Advanced Studies, Dublin 4, Ireland*

Accepted 31 July 2001

Abstract

In this report, we describe a novel application of the Baker's map. We demonstrate that the chaotic properties of this map can be used to implement basic operations in Boolean logic. This observation leads naturally to the possibility of new computational models and implementations for conventional computational systems. Here we show that by considering the variation of the fractal dimension of its attractor, and using varying parameter values as inputs, the generalised Baker's map can be used as a natural exclusive OR (XOR) gate. Further, this map can also be used to create other logical functions such as the AND gate. The efficacy of our results are demonstrated by means of a concrete application; namely by designing, to the best of our knowledge, for the first time, a half-adder that is constructed entirely by utilising chaotic dynamics.

1. Introduction

Nonlinear dynamics, as a subject, has reached a considerable degree of maturity in recent years. However, despite rapid theoretical advances in the subject, the general area of nonlinear dynamics has also been characterised by a lack of engineering applications (with the exception of nonlinear control [1]) that exploit the fundamental theory and properties of nonlinear systems. This observation is somewhat surprising since many engineering systems are designed to exhibit behaviour commonly found in nonlinear systems. Examples of this abound in the aerospace industry. Fighter aircraft, for instance, are designed to have unstable dynamics to aid manoeuvrability under extreme flight conditions. Such aircrafts are artificially stabilised under normal flight conditions. The properties of local instability of trajectories (rapid manoeuvrability) and global stability of orbits (safety) are often found in nonlinear and chaotic systems. One property in particular is extremely attractive from an engineering perspective: exponential sensitivity to initial conditions, which allows chaotic systems to be hypersensitive to changes in system parameters; and yet underlying this sensitivity, chaotic systems have global properties, such as fractal dimension of state-space attractor, which can be extracted and used as output variables. This observation suggests that chaotic dynamics can be used as the design basis for rapid system identification, and in the design of high performance control systems. Here, we begin the process of examining the suitability of chaotic maps for such engineering applications.

In this paper, we will show how the generalised Baker's map can be used to solve the exclusive OR (XOR) problem. This is a fundamental problem of pattern recognition, and involves telling at a single glance whether a point belongs to one of the two classes: class A or NOT class A (class B), where class A consists of two diagonally opposite corners of a unit square, and class B consists of the other two corners. The inability of a single-layer perception to solve this problem

* Corresponding author. Tel.: 353-1-7086067; fax: 353-1-7083967.
E-mail address: alan.rogers@may.ie (A. Rogers).

0960-0779/02/$ - see front matter
PII: S0960-0779(01)00181-3

is considered to be a severe drawback for ANNs as a mechanism for nonlinear problem-solving. We will consider the XOR problem in more depth later.

The generalised Baker's map is a two-dimensional, three-parameter, nonlinear mapping, which is chaotic for virtually all parameter values. We use it here because it is one of the best-understood chaotic maps, and is particularly suited to rigorous analysis (see [2]). It also has the useful property that its Lyapunov dimension is monotonically increasing for a wide range of parameter values, and we shall utilise this when we develop the XOR gate. To the best of our knowledge, neither Baker's map, nor any other chaotic map, has been previously used to solve the XOR problem in this way.

The rest of the paper is organised as follows. In Section 2 we will describe the XOR problem, and the way in which artificial neural networks (ANNs) can, and more importantly, cannot, solve this problem. We shall describe the Baker's map in Section 3. In Section 4 we show how the Baker's map can act as a natural XOR system, and we present some of the properties, advantages, and drawbacks, of the new system. In Section 5 we show how a half-adder can be built using two Baker's maps. In Appendix A, we briefly describe ANNs and their use as pattern classifiers.

2. Pattern recognition and the XOR problem

The pattern recognition problem consists of designing algorithms that automatically classify feature vectors associated with specific patterns as belonging to one of a finite number of classes. A benchmark problem in the design of pattern recognition systems is the Boolean exclusive OR (XOR) problem. The standard XOR problem is depicted in Fig. 1. Here the diagonally opposite corner-pairs of the unit square form two classes, A and B (or NOT A). From the figure, it is clear that it is not possible to draw a single straight line which will separate the two classes. This observation is crucial in explaining the inability of a single-layer perceptron to solve this problem (an overview of the perceptron is given in Appendix A).

This problem can be solved using multi-layer perceptrons (MLPs), or by using more elaborate single-layer ANNs such as the radial basis function neural network [3]. However, the inability of simple ANNs, such as the Adeline [4], to solve this problem, effectively ended research interest in the area of ANNs for over 20 years, which highlights the importance of the XOR problem in the design of pattern recognition systems. In this paper, we show that the generalised Baker's map can be trained to solve this problem in a straightforward manner.

3. Chaos and the Baker's map

3.1. The generalised Baker's map

In their classic study of fractal dimensions, Farmer et al. [4] introduced the generalised Baker's map in order to obtain rigorous results on the dimension of strange attractors. It is a transformation of the unit square $[0,1] \times [0,1]$, and has three parameters, R_1, R_2 and S:

$$x_{n+1} = \begin{cases} R_1 x_n & \text{if } y_n < S, \\ 1/2 + R_2 x_n & \text{if } y_n \geqslant S, \end{cases}$$
$$y_{n+1} = \begin{cases} y_n / S & \text{if } y_n < S, \\ \dfrac{y_n - S}{1 - S} & \text{if } y_n \geqslant S. \end{cases} \tag{1}$$

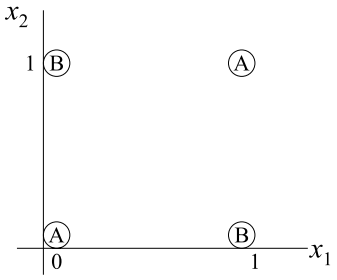

Fig. 1. The exclusive OR (XOR) problem: points (0,0) and (1,1) are members of class A; points (0,1) and (1,0) are members of class B.

We illustrate the Baker's map transformation in Fig. 2. As can be seen from Eq. (1), the mapping depends on whether the point in question is above or below a horizontal line $y = S$.

Since the Baker's Map is a mapping of the unit square, we restrict S to the range (0,1) and R_1 and R_2 to the range (0, 0.5]. In Fig. 2, we show the action of the map on the entire unit square. Iterating the map gives two vertical strips, whose widths depend on R_1 and R_2. Iterating the map again gives four strips, then eight strips, and so on. The attractor is the union of a line segment (vertical direction) and a Cantor set (horizontal direction).

3.2. Lyapunov numbers and Lyapunov dimension of the Baker's map

It can be seen in Fig. 2 that the action of the map leads to 'stretching' in the y-direction and 'compressing' in the x-direction. It is possible to put these actions into a more mathematical framework by using the notion of Lyapunov numbers. These numbers characterise the stability of the map, and are defined as follows:

Let $J_n = [J(x_n) \cdot J(x_{n-1}) \cdot \ldots \cdot J(x_1)]$, where $J(x)$ is the Jacobian of the map, $J(x) = (\partial F/\partial x)$, for some map F.

Let $j_1(n) \geqslant j_2(n) \geqslant \cdots \geqslant j_p(n)$ be the magnitudes of the p eigenvalues of J_n.

Then the Lyapunov numbers are given by

$$\lambda_i = \lim_{n\to\infty} [j_i(n)]^{1/n}, \quad i = 1, 2, \ldots, p. \tag{2}$$

Since the Baker's map is two-dimensional, it will have two Lyapunov numbers, characterising the average stretching/compression factors in the x- and y-directions (see Fig. 3). Note that the *Lyapunov exponents* are simply the logarithms of the Lyapunov numbers. It is customary to order the Lyapunov numbers, so that $\lambda_1 > \lambda_2 > \cdots > \lambda_n$.

The Lyapunov dimension was introduced by Kaplan and Yorke [5] in the so-called Kaplan–Yorke conjecture: that the Lyapunov dimension D_L is the same as the information dimension for "typical" attractors. For the Baker's map,

$$D_L = 1 + \frac{\log \lambda_1}{\log 1/\lambda_2}. \tag{3}$$

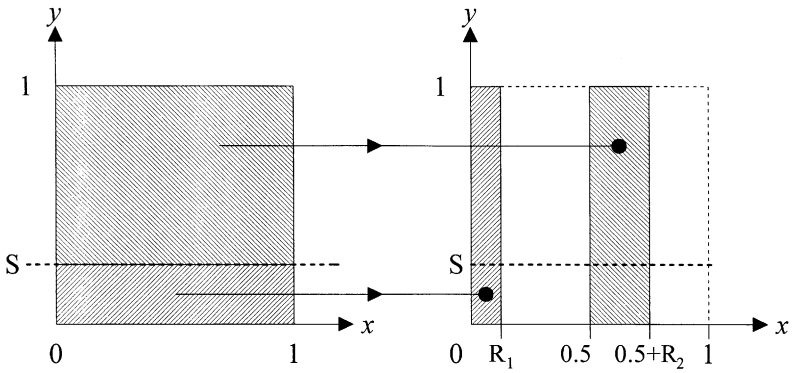

Fig. 2. Action of the Baker's map on unit square: transforms square into two strips, then four strips, eight strips, and so on.

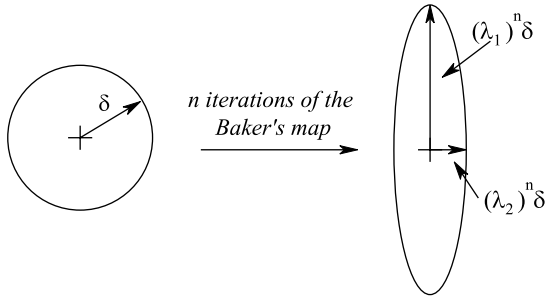

Fig. 3. Lyapunov Numbers characterise the average stretching factors of some small circle of radius δ. In this case, $\lambda_1 > 1$ and $\lambda_2 < 1$.

The Jacobian of Eq. (1) can be written in the following form:

$$J = \begin{pmatrix} L_2(y) & 0 \\ 0 & L_1(y) \end{pmatrix}, \quad \text{where} \quad L_1(y) = \begin{cases} 1/S & \text{when } y < S, \\ 1/(1-S) & \text{when } y > S, \end{cases} \quad L_2(y) = \begin{cases} R_1 & \text{when } y < S, \\ R_2 & \text{when } y > S. \end{cases}$$

So from Eq. (2) we get

$$\lambda_1 = \lim_{n\to\infty} [L_1(y_n)\dots L_1(y_1)]^{1/n},$$

$$\lambda_2 = \lim_{n\to\infty} [L_2(y_n)\dots L_2(y_1)]^{1/n}.$$

By taking logs, and with some manipulation, noticing that the orbits are ergodic in the y-direction, we find that the Lyapunov exponents are:

$$\log \lambda_y = S \log \frac{1}{S} + (1-S) \log \frac{1}{1-S}, \tag{4a}$$

$$\log \lambda_x = S \log R_1 + (1-S) \log R_2. \tag{4b}$$

In our implementation of the XOR gate, we only require two input parameters, so we shall let $R_2 = R_1$, in which case we find that

$$\log \lambda_x = \log R_1. \tag{5}$$

4. Using the Baker's Map to solve the XOR problem

4.1. Background

If we plot the fractal dimension of Baker's map for varying values of R and S, it becomes obvious how we can use the map to solve the XOR problem. Firstly, we show how the Lyapunov exponents (Eqs. (4a) and (5)) vary with R and S (see Fig. 4). Clearly, since the map is contractive in x-direction, the Lyapunov exponent in that direction is always negative. Conversely, the map is expansive in the y-direction, and therefore that Lyapunov exponent is always positive.

From Eq. (3), the Lyapunov dimension is given by

$$D_{\mathrm{L}} = 1 - \frac{\log \lambda_y}{\log \lambda_x}. \tag{6}$$

In Fig. 5, we plot D_{L} against R, with S as a parameter. Notice that the fractal dimension varies between 1 and 2, as we would expect. Due to the symmetry of Fig. 4(b), the fractal dimension is symmetrical about $S = 0.5$. We have chosen slightly asymmetrical values of S to illustrate this.

We can choose values of R and S, so that a pair (low R, high S) and another pair (high R, low S) give the same fractal dimension, say D_{A}. This corresponds to a diagonally opposite corner pair in the XOR problem. We can say, therefore, that if the fractal dimension $D_{\mathrm{L}} = D_{\mathrm{A}}$, then the inputs are in class A, and if $D_{\mathrm{L}} \neq D_{\mathrm{A}}$, then the inputs belong to class B. Note that we always limit S to the range [0, 0.5], to ensure a unique fractal dimension for any given (R, S) pair.

For example, in Fig. 6, we could say that the following pairs of parameters form classes.

Obviously, the points in Table 1 do not lie on a perfect square, but that is unimportant. The key idea is that two pairs of diagonally opposing points are mapped to the same class. It is also clear that we are quite restricted in the possible pairs of points which we can map to the same fractal dimension. However, if we choose any four (R, S) pairs of points corresponding roughly to (low, low), (low, high), (high, low) and (high, high), then by drawing a straight line through the (low, high), (high, low) points and intersecting the y-axis, we can effectively solve the XOR problem for much larger set of inputs. We call the intersection of this line with the y-axis, D_{M}, the (modified) Lyapunov dimension. This is illustrated in Fig. 6.

Procedure for calculation of D_M:

(i) Given four points in the R–S plane, select the two points belonging to the same class: (R_a, S_b), (R_b, S_a) in Fig. 6.
(ii) Calculate the Lyapunov dimensions corresponding to the two points, called D_1, D_2.
(iii) Calculate the slope, $m = (D_1 - D_2)/(R_a - R_b)$.
(iv) The dimension $D_{\mathrm{M}} = D_1 + m \cdot R_a = D_2 + m \cdot R_b$.

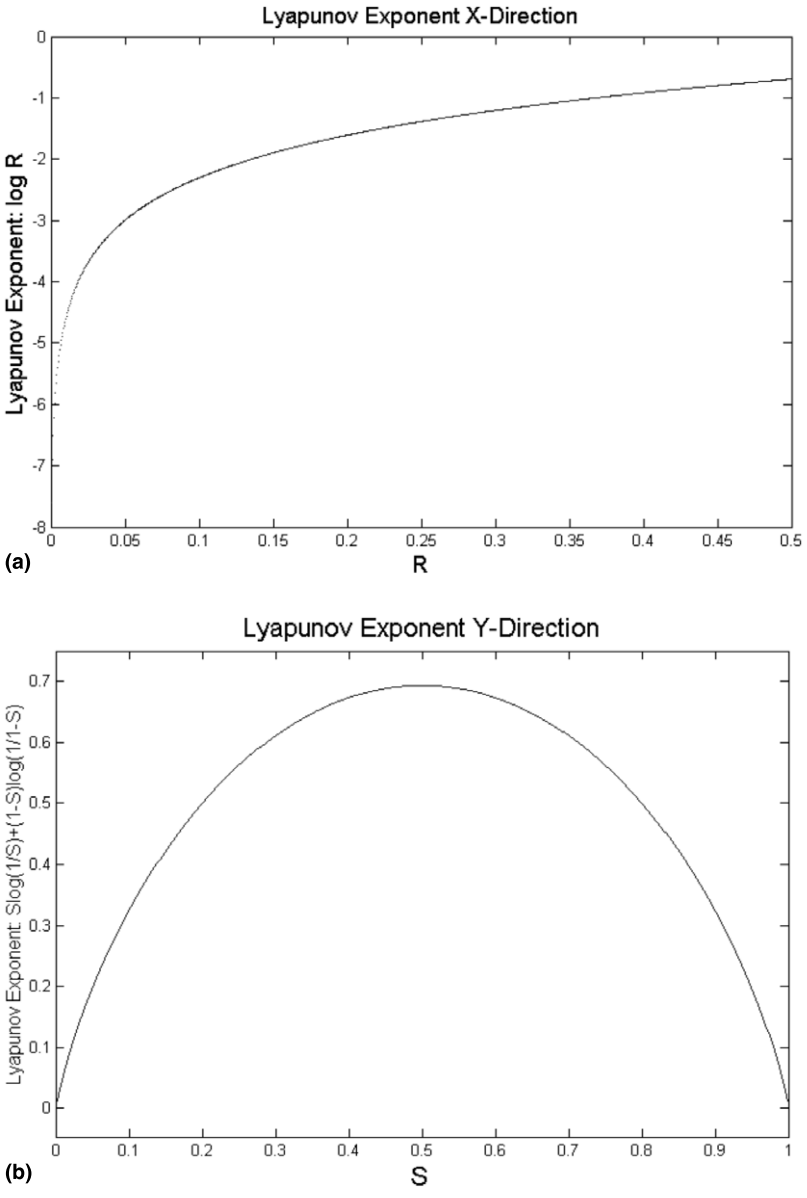

Fig. 4. Variation of Lyapunov exponent in the (a) x-direction, (b) y-direction.

As D_M is constantly calculated, we can tell whether the inputs are in class A, or not. An algorithm of this form is referred to as a training algorithm in the ANN and statistical pattern recognition literature [8]. The availability of such an algorithm, and its complexity, ultimately determines the applicability of a particular paradigm for a given problem. In our case, given a set of class labels, and a set of vectors, the training parts of the pattern recognition problem is trivial, involving only the simple calculation of a slope. For an ANN, solving this problem requires repeated calculation of the slope for at least two hyperplanes, and so is more computationally intensive.

4.2. Computer simulation

The system is easily implemented with a few lines of code. Essentially, we need to simulate Baker's map, given its input parameters, and then, using its state variables x and y, compute the Lyapunov dimension D_L of the attractor (see Fig. 7).

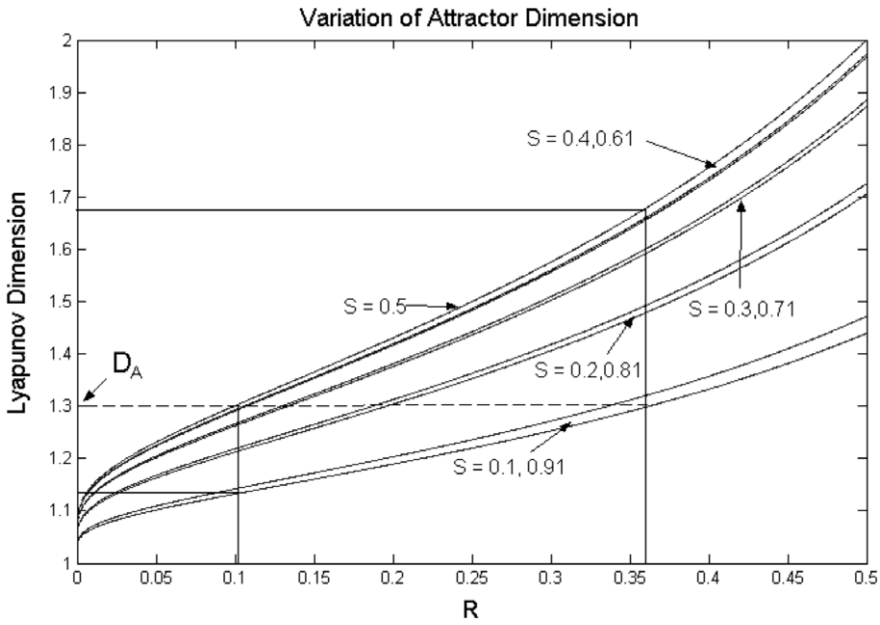

Fig. 5. Variation of fractal dimension with varying parameter values.

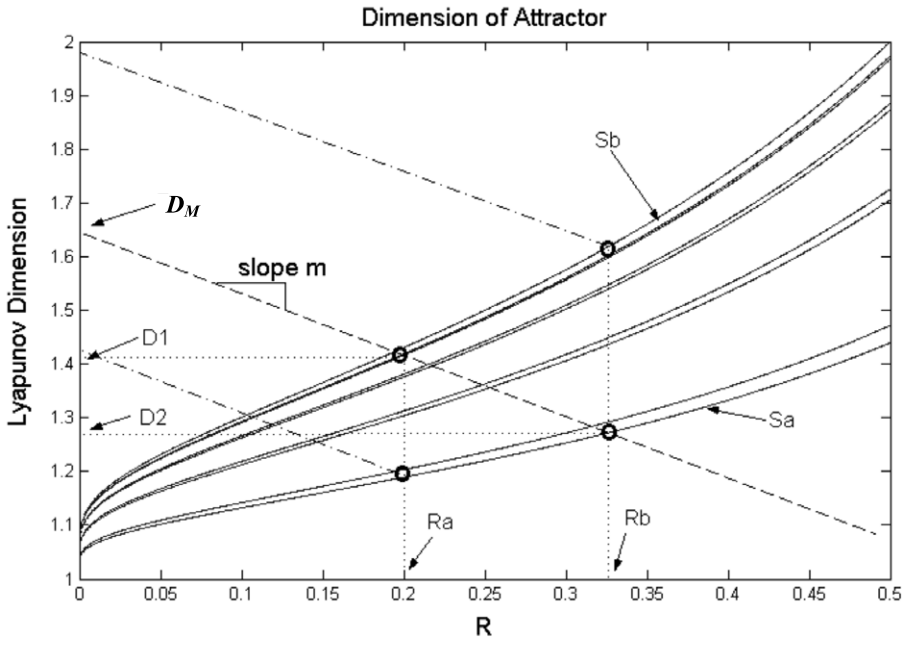

Fig. 6. A more general way of solving the XOR problem: draw a straight line through the two points belonging to class A (say), and find where the line intersects the y-axis.

Obviously, the speed of the system depends on the computation of the Lyapunov dimension. The traditional way to do this is quite slow [6], and assumes that no detailed information is available about the system, that is to say, only a time-series $x_0, x_1, x_2, \ldots$, is available from the system. Given this time series, some value from the sequence is selected,

Table 1
Parameter values and their corresponding fractal dimension, and class, as in Fig. 5

R value	S value	Fractal dimension	Class
~0.1	0.5	1.3	A
~0.36	~0.1	1.3	A
~0.1	~ 0.1	~1.14	B (NOT A)
~0.36	0.5	~1.68	B (NOT A)

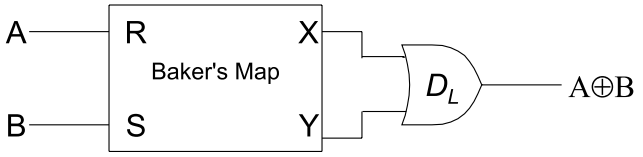

Fig. 7. The chaotic XOR system: the outputs from the map are the state variables x and y, and these are used to compute the Lyapunov dimension.

say x_i, and then one searches the sequence for another value x_j that is close to x_i. The sequence of differences is assumed to diverge exponentially, on the average:

$$\begin{aligned} d_0 &= |x_j - x_i| \\ d_1 &= |x_{j+1} - x_{i+1}| \\ &\vdots \\ d_n &= |x_{j+n} - x_{i+n}| \end{aligned} \tag{7}$$

We assume that

$$d_n = d_0 e^{\lambda n},$$

which, after taking logarithms, gives

$$\lambda = \frac{1}{n} \log \frac{d_n}{d_0}. \tag{8}$$

Since we would like our system to be as fast as possible, this method is computationally expensive, as it involves continually searching through some large array of numbers, and then performing additional calculations given in Eqs. (7) and (8). As we have Baker's map data readily available, and since we have its input parameters already, we have found a quicker way to compute the Lyapunov numbers. They are calculated as follows: we iterate the Baker's map $f(x,y)$ as normal (call it B_1), but we also iterate another Baker's map (B_2) in parallel with it. For each pair (x_n, y_n) generated by B_1, we use some nearby pair of numbers $(x_n + \delta, y_n + \varepsilon)$ as initial conditions for B_2. We then iterate B_2 once (we have found that iterating more than once does not improve accuracy, but merely slows things down). Now, we compute the Lyapunov numbers

$$\begin{aligned} \lambda_x &= \log \frac{|f(x_n, y_n) - f(x_n + \delta, y_n + \delta)|_{X \text{ comp.}}}{\delta}, \\ \lambda_y &= \log \frac{|f(x_n, y_n) - f(x_n + \varepsilon, y_n + \varepsilon)|_{Y \text{ comp.}}}{\varepsilon}. \end{aligned} \tag{9}$$

The numbers thereby computed tend to be noisy, but when averaged, they give the expected theoretical values. Note also that since the map is always contracting in the x-direction, choosing a very small value for δ gives entirely inaccurate results, hence we tend to use $\delta \approx 1$.

To illustrate the action of the system, we choose four distinct points, more or less arbitrarily (see Fig. 8).

We compute the slope of the line between points 1 and 2, belonging to class A, to be $m = -1.60667$, and the (modified) Lyapunov dimension $D_M = 1.752$ (see Table 2).

In Fig. 9, we plot the output from Baker's map system. Here, we have averaged every 200 points, to smooth the output. Clearly, there is a tradeoff between speed of pattern classification, and accuracy. If we average more points, then

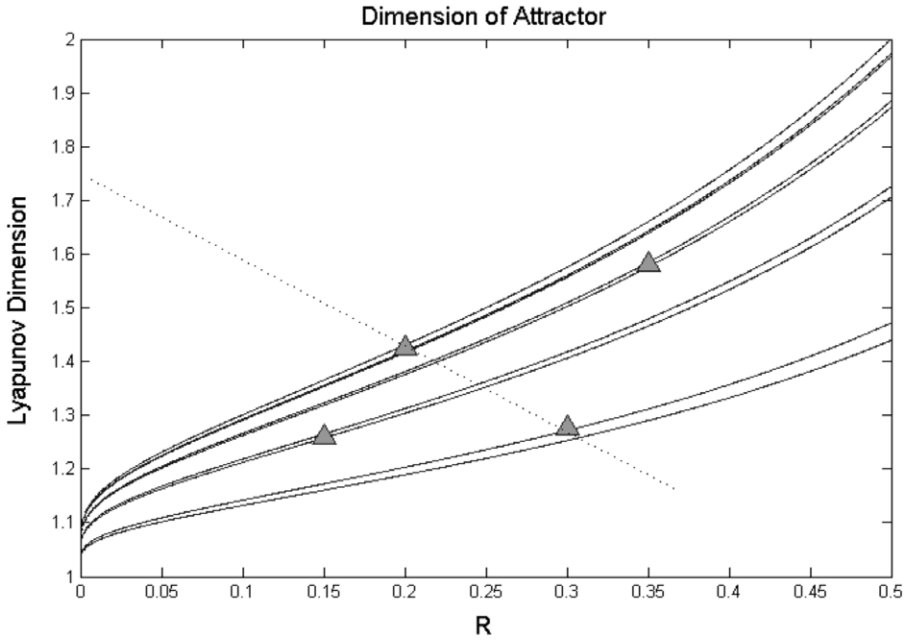

Fig. 8. The four points selected to illustrate the chaotic XOR system.

Table 2
Parameter values and their corresponding classes, as shown in Fig. 6

Point no.	*R* value	*S* value	Class
1	0.2	0.5	A
2	0.3	0.1	A
3	0.15	0.2	B
4	0.35	0.4	B

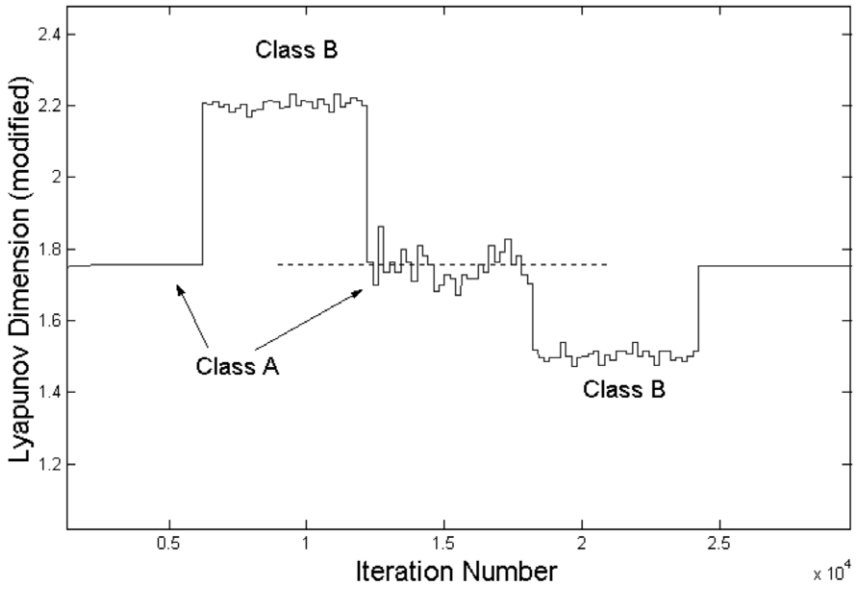

Fig. 9. Output from the chaotic XOR system, with inputs as in Table 2. Contiguous sets of 200 points are averaged. Class A corresponds to a modified Lyapunov dimension $D_M \approx 1.75$. Note that we cycle through points (1), (4), (2), (3) and (1), respectively.

we get a smoother output, but this introduces a delay into the recognition process (see Figs. 10 and 11). Note also that the relative smoothness also depends on how large the value of S is, with $S = 0.5$ giving a perfectly smooth output. This is because the expansion rates in the y-direction are the same only when $S = 0.5$ (see Fig. 2).

It is clear from Fig. 11 that by merely observing if the output dimension lies in some suitable range about 1.75, we can tell if the input is in class A, or class B.

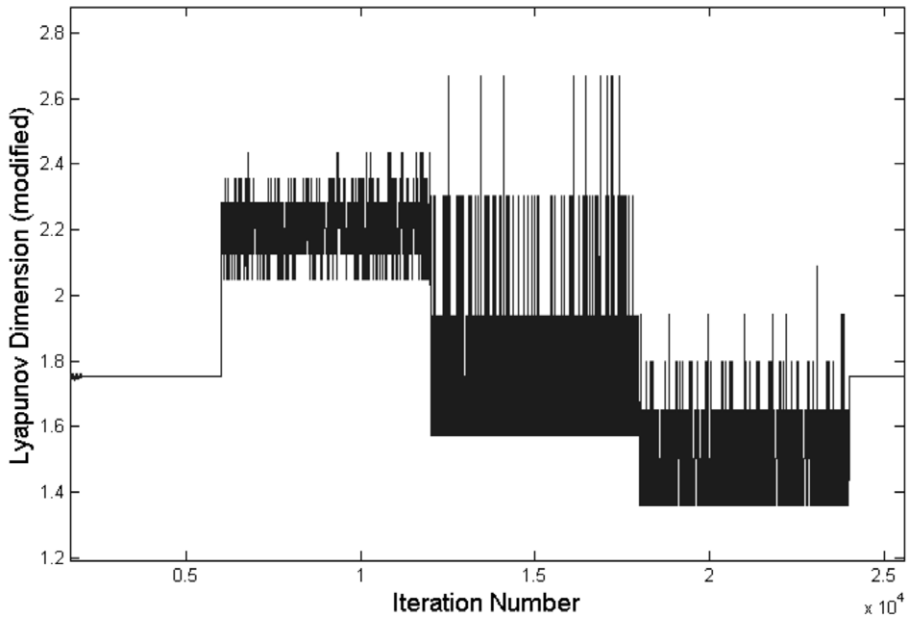

Fig. 10. Output from the system with a 5-point averaging window. Since the output dimension switches between two values only, the 5-point averaging leads to six possible values for the output.

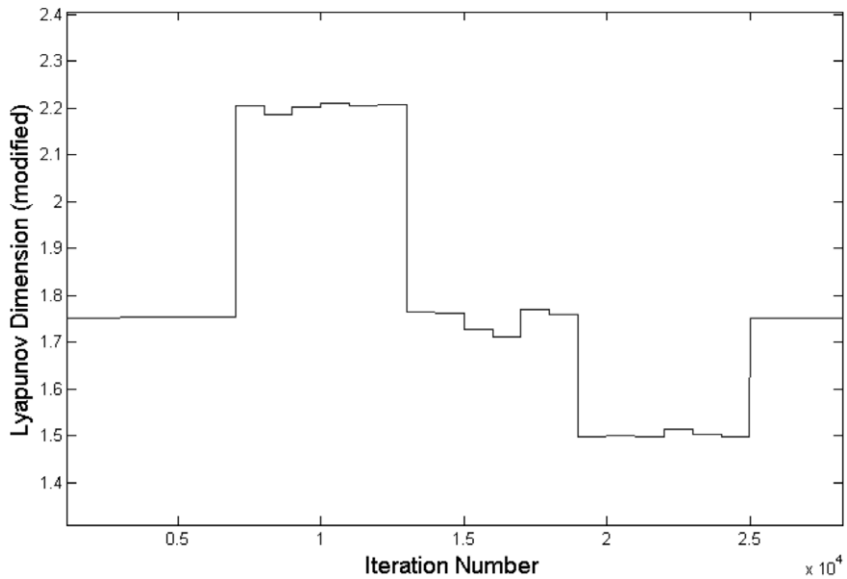

Fig. 11. Output from the system with 1000-point averaging windows.

5. Implementation of half-adder

Arithmetic circuits, constructed using simple Boolean logic gates, lie at the heart of modern digital computers. The half-adder is one of the most fundamental of these circuits. It adds two binary numbers, and generates a carry out signal, as illustrated in Table 3.

Table 3
Truth Table for a standard half-adder

Inputs		Outputs	
A	B	Sum	Carry out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1
	A+B	$\sum$	C_{out}

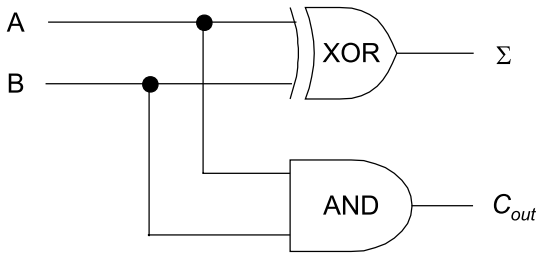

Fig. 12. Schematic diagram of 1-bit half-adder.

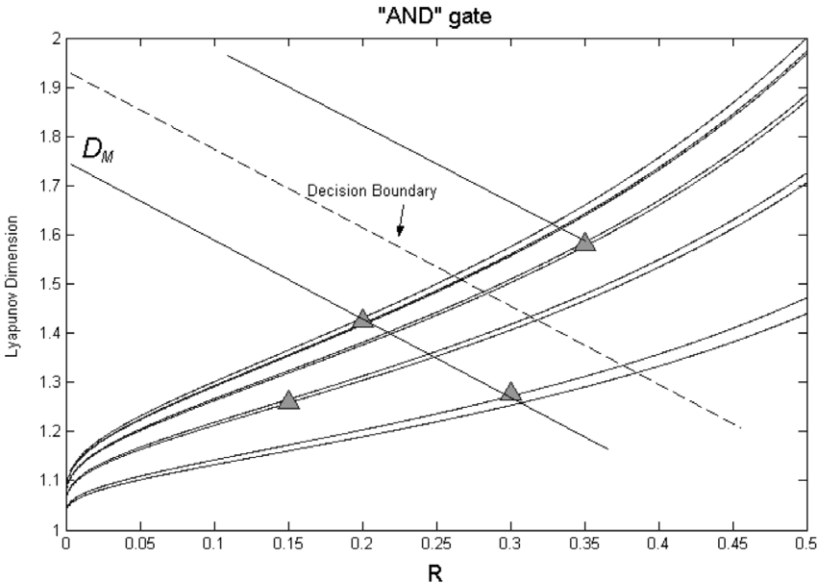

Fig. 13. Chaotic AND gate: only points lying above the decision boundary are designated TRUE (1). Therefore only the point with (high R, high S) is designated TRUE, giving an AND gate.

Table 4
Results from chaotic half-adder, with parameter values as in Table 2

XOR gate				AND gate			
R	S	D_M	Class	R	S	D_M	Class
0.15	0.20	1.513	0	0.15	0.20	1.513	0
0.15	0.20	1.508	0	0.15	0.20	1.508	0
0.15	0.20	1.502	0	0.15	0.20	1.502	0
0.2	0.50	1.744	1	0.2	0.50	1.744	0
0.2	0.50	1.752	1	0.2	0.50	1.752	0
0.2	0.50	1.752	1	0.2	0.50	1.752	0
0.3	0.10	1.736	1	0.3	0.10	1.736	0
0.3	0.10	1.740	1	0.3	0.10	1.740	0
0.3	0.10	1.755	1	0.3	0.10	1.755	0
0.35	0.40	2.174	0	0.35	0.40	2.174	1
0.35	0.40	2.200	0	0.35	0.40	2.200	1
0.35	0.40	2.198	0	0.35	0.40	2.198	1

The class columns correspond to the *Sum* and *Carry Out* columns of the standard half-adder.

Half-adders can be combined to make more complex logical units such as full-adders/subtractors, and one can combine many 1-bit full-adders to give higher-order adders, such as the 64-bit and 128-bit adders present in modern day calculators, computers, and other such technologies (see, for example [7]).

As can be seen from Fig. 12, a half-adder consists of an XOR gate and an AND gate. It only remains, therefore, for us to fashion an AND gate using Baker's map. This is quite straightforward – we place a decision boundary between D_M and the point corresponding to (high R, high S), as shown in Fig. 13.

If the modified Lyapunov dimension lies above the decision boundary, we classify it as 1 (TRUE *or* Class A), and if it lies below, we classify the output as 0 (FALSE *or* NOT Class A). This gives us the carry out signal, C_{out} as shown in the schematic. In a practical implementation, this would merely involve placing a comparator at the output of the system. Note also that the chaotic AND gate allows considerable flexibility in which set of inputs gives a TRUE output.

We have simulated the half-adder using the same set of points as earlier. The results are shown in Table 4. The decision boundary for the AND gate was taken as being halfway between the value of D_M and the value of the modified Lyapunov component of the (high R, high S) point. We obtain our binary outputs by denoting values above the decision boundary as being in class 1, and below as being in class 0. Similarly, for the XOR gate, values lying close to the D_M line are denoted class 1, and outlying points are denoted class 0.

6. Conclusions

We have described how the Baker's map can be used to provide an elegant solution to a long-standing problem in pattern recognition: the exclusive OR (XOR) problem of single-layer perceptrons. By using the parameters of a chaotic map as inputs, and the fractal dimension of the map as the output, it has been shown that the generalised Baker's map acts like a natural XOR gate. Further, we have shown that the map can also be used to function as other Boolean logic functions, such as the AND gate. We have demonstrated the effectiveness of our results by means of a half-adder, constructed entirely by utilising chaotic dynamics.

Acknowledgements

This work was supported by the Irish Science and Technology agency, Enterprise Ireland, under research grant No. SC-00-86.

Appendix A. The perceptron

The term ANN is used to describe a wide-class of statistical pattern recognition and function approximation paradigms [3,4,8]. Originally motivated by the perceived structure of the human brain, neural neworks consist of a large

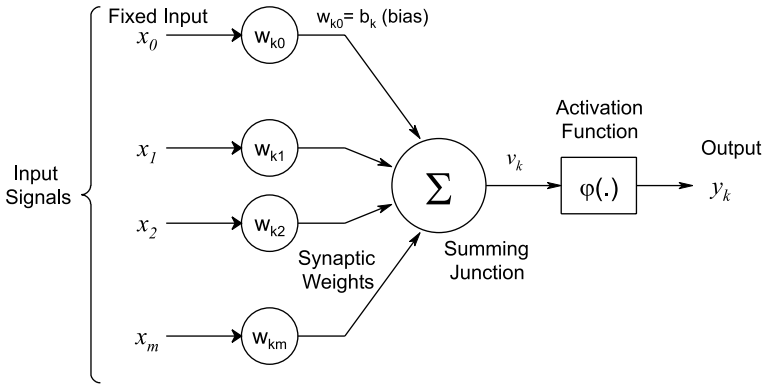

Fig. 14. Model of a neuron.

number of individual units, or neurons, connected together in some way. Information is usually stored in the ANN by modifying the connection strength between neurons, and by modifying the nonlinear activation characteristics of the individual neurons. Perhaps the most widely used ANN is the multi-layer perceptron (MLP). Originally derived as an extension of the single-layer perceptron, the MLP has found widespread academic application in pattern recognition problems, and in problems that require nonlinear static and dynamic function approximation [9]. The popularity of this network type stems directly from the availability of computationally tractable algorithms for adapting the parameters of the network based upon training data.

In Fig. 14, we show a typical neuron, with its constituent parts. Here, we have m inputs, which are modified by the synaptic weights and summed at the junction. The activation function determines the output from the neuron. Typically, a sigmoid function or a Heaviside function is used as the activation function, although there are many other possibilities. The MLP is constructed by connecting together many neurons.

A.1. Single-layer perceptron and the XOR problem

A single-layer perceptron (introduced by Rosenblatt [10]), or Adaline (introduced by Widrow and Hoff [11]), is a particular type of neuron, based around the model shown in Fig. 14, with a hard limiter as the activation function. That is, when the hard limiter input is positive, the neuron produces $a+1$ output, and it produces $a-1$ output when the input is negative. For some pattern recognition problems, single-layer perceptrons can be used to separate a set of inputs into two classes. [1] We shall now describe how this is achieved.

First of all, it is customary to show the single-layer perceptron as a signal-flow graph.

As can be seen from Fig. 15, the inputs are given by $x_1, x_2, \ldots, x_m$. The externally applied bias b, can be used to modify the activation potential v (also called the *local induced field*). The hard limiter input is given by

$$v = \sum_{i=1}^{m} w_i x_i + b. \tag{A.1}$$

In Pattern Recognition, the task of the perceptron is to classify some set of inputs $\{x_1, x_2, \ldots, x_m\}$ into one of two classes, C_1 or C_2. In other words, if the inputs belong to class C_1, then the neuronal output, y, is +1 (say), and the output y is −1 if the inputs belong to class C_2. If we look closely at Eq. (A.1), we can see how this is done. A hyperplane in m-dimensional space is given by the following relation:

$$\sum_{i=1}^{m} w_i x_i + b = 0. \tag{A.2}$$

This hyperplane forms the *decision boundary* between the two classes C_1 and C_2. If v is positive, then the points lie on the one side of the decision boundary, and lie on the other side if v is negative. This is illustrated in Fig. 16.

[1] If we have more than one perceptron, then we can separate the inputs into more than two classes.

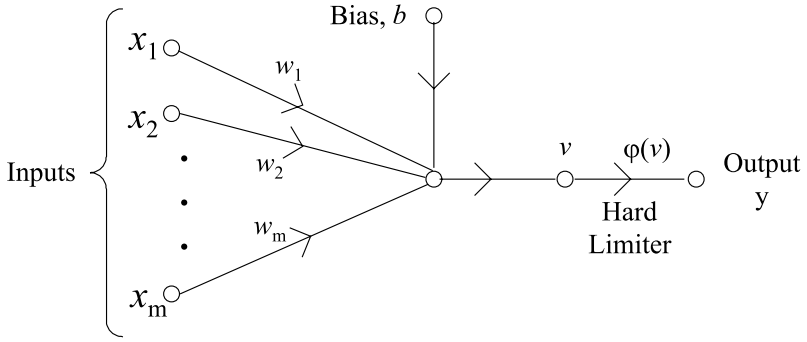

Fig. 15. Perceptron as a signal-flow graph.

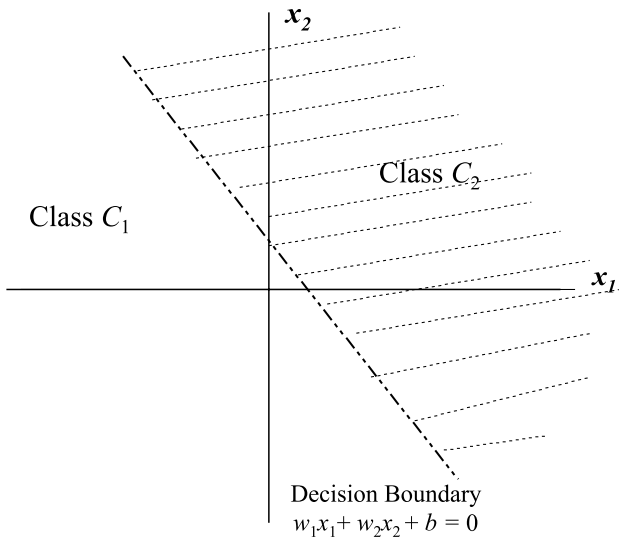

Fig. 16. Hyperplane as a decision boundary for a two-class, two-dimensional pattern classification problem.

For simplicity, we have only illustrated the two-dimensional case (i.e. two inputs x_1, x_2). It is also possible to have several perceptrons in parallel, which allows the classification of points into more than two classes. Clearly, if we had two perceptrons, then we would have two hyperplanes, allowing the classification of points into one of four classes. It should also be clear that though, because the hyperplane acts as a linear boundary, the classes must be linearly separable, i.e., one must be able to draw a line (hyperplane) down the middle separating the two sets of points. Otherwise, the perceptron may give an incorrect classification for some points (see Fig. 17).

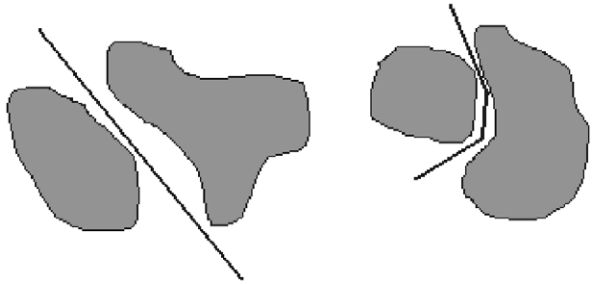

Fig. 17. The sets on the left are linearly separable; the sets on the right are not.

Based upon the above discussion, it is clearly not possible to solve the XOR problem using a single-layer perceptron since the members of each pattern class are not linearly separable. Its use is however possible to solve this problem using an MLP, more specifically using a two-layer perceptron; the first layer transforms the problem into a linearly separable problem, which the second layer then solves.

References

[1] Ott E, Grebogi C, Yorke J. Controlling chaos. Phys Rev Lett 1990;64:1196.
[2] Bishop CM. Neural networks for pattern recognition. Oxford: Clarendon Press; 1995.
[3] Haykin S. Neural networks – a comprehensive foundation. Englewood Cliffs, NJ: Prentice-Hall; 1999.
[4] Farmer JD, Ott E, Yorke JA. The dimension of chaotic attractors. Physica D 1983;7:153–80.
[5] Kaplan J, Yorke J. Chaotic behaviour of multidimensional difference equations. In: Peitgen HO, Walther HO, editors. Functional differential equations and the approximations of fixed points, Lecture Notes in Mathematics, vol. 730. Berlin: Springer; 1979. p. 204–7.
[6] Hilborn RC. Chaos and nonlinear dynamics. Oxford: Oxford University Press; 1994.
[7] Nelson VP et al. Digital logic circuit analysis and design. Englewood Cliffs, NJ: Prentice-Hall; 1995.
[8] Ripley BD. Pattern recognition and neural networks. Cambridge: Cambridge University Press; 1996.
[9] Narendra KS. Neural networks for control theory and practice. Proc IEEE 1996;84:1385–406.
[10] Rosenblatt F. Principles of neurodynamics. New York: Spartan; 1962.
[11] Widrow B, Hoff ME. Adaptive switching circuits. In: 1960 IRE WESCON Convention Record, vol. 3. New York: IRE; 1960. pp. 96–104.