Encoding information in chemical chaos by controlling symbolic dynamics

Erik M. Bollt
Department of Mathematical Sciences, United States Military Academy, West Point, New York 10996-1786

Milos Dolnik
Department of Chemistry and Center for Complex Systems, Brandeis University, Waltham, Massachusetts 02254-9110

(Received 5 November 1996)

In this paper we describe a technique of encoding and then decoding symbol sequences containing information into chaotic oscillations of the Belousov-Zhabotinsky reaction. The encoding technique is based on controlling the chaotic oscillations by applying small parameter perturbations and on learning the grammar of corresponding symbol dynamics produced by the free-running chaotic system. The use of small parameter perturbations requires that we respect the grammar of the symbol dynamics which represents the physical dynamical system. We present a method for learning the grammar of a symbol dynamics in terms of allowed transitions between the bins defined by the symbol generating partition. The encoding technique can be easily utilized for targeting and stabilization of any unstable periodic orbit. [S1063-651X(97)10904-7]

PACS number(s): 05.45.+b

I. INTRODUCTION

A great deal of recent research in applied and theoretical dynamical systems has been focused on taking advantage of the fact that a chaotic dynamical system can be controlled. The sensitive dependence characteristic of chaos is actually advantageous to building a highly agile control system in which a small deliberate perturbation can have a large response. A chaotic attractor can be considered as an unlimited reservoir of periodic behavior. Ott, Grebogi, and Yorke introduced the OGY method [1] to stabilize an unstable period orbit embedded in a chaotic attractor. This original idea opened the field of controlling chaos. OGY uses a local linear feedback control loop by targeting the stable manifold of an unstable fixed point through small parameter variations. Ergodicity causes an arbitrary initial condition to eventually wander close enough to the fixed point that the tiny parameter variations are sufficient to capture the orbit. When ergodicity does not cause the arbitrary initial condition to wander close and fast enough to the unstable fixed point, the ‘‘butterfly effect’’ allows us to steer trajectories to targets with only small perturbations [2]; this is called ‘‘targeting.’’

Controlling chaos has provided many exciting interdisciplinary applications, including control of lasers, magnetoelastic materials, living heart tissue, and interplanetary travel. Of particular relevance to this paper, Petrov *et al.* [3,4] have experimentally demonstrated the OGY method to control chemical chaos. They applied a map-based, proportional-feedback algorithm to stabilize periodic behavior of the Belousov-Zhabotinsky (BZ) reaction in its chaotic regime.

The link between symbol dynamics and a physical dynamical system is well established [5] as a descriptive tool. The link can be thought of as a change of coordinates in which properties of the dynamics are preserved. The link also implies a justification for many of the information theoretic tools used in characterizing chaos, including topological entropy and Markov partitions [6–8]. Corresponding to a physical trajectory, there exists an infinite symbol sequence. Whether or not the link is realized, the existence of the symbol dynamical description implies that controlling a trajectory of the physical dynamical system is equivalent to controlling a symbol sequence. Recently, Hayes *et al.* [9,10] demonstrated, numerically and experimentally, that the connection between information theory and chaotic systems can be used to encode a message into a chaotic electronic circuit by controlling the symbol dynamics through *small* perturbations.

In this paper, we demonstrate, by numerical experiment, the possibility that information can be encoded into the chaotic oscillations of the BZ reaction. There exists a possibility that biological systems might hold and control information flow in the oscillations of their defining dynamical systems. However, a main technical problem of encoding is learning the grammar of the corresponding physical dynamical system. We present a practical grammar learning algorithm, and then we encode and decode information in the form of binary sequences. The algorithm is generally applicable to other systems.

In Sec. II we present the Györgyi-Field [12] model of the BZ reaction. In Sec. III we review the theoretical background for a symbol dynamics description of a map in physical coordinates, and then we present our technique to learn the grammar in terms of all permitted transitions between the bins defined by the generating partition. In Sec. IV, given the rules of the grammar, we present the technique of encoding and then decoding information in chaotic oscillations by using the form of the grammar defined in Sec. III. The fact that complete control of symbol dynamics implies complete course-grained control of *all* possible orbits of the physical dynamical system is noted in Sec. V, which discusses targeting. The technical issues concerning how to achieve a required small variation of the map on the surface of the section and thus transmit a desired bit is described in Sec. VI. In Sec. VII we present the numerical experiments in which we encode a short message, and we also target simple unstable periodic orbits. We conclude with an assessment of the technique and its potential applications.

TABLE I. Parameters used in the simulations.

Parameter	Value	
k_1	4×10^6	$\mathrm{dm}^6\ \mathrm{mol}^{-2}\ \mathrm{s}^{-1}$
k_2	2	$\mathrm{dm}^6\ \mathrm{mol}^{-2}\ \mathrm{s}^{-1}$
k_3	3×10^3	$\mathrm{dm}^3\ \mathrm{mol}^{-1}\ \mathrm{s}^{-1}$
k_4	55.2	$\mathrm{dm}^{7.5}\ \mathrm{mol}^{-2.5}\ \mathrm{s}^{-1}$
k_5	7×10^3	$\mathrm{dm}^3\ \mathrm{mol}^{-1}\ \mathrm{s}^{-1}$
k_6	0.09	$\mathrm{dm}^3\ \mathrm{mol}^{-1}\ \mathrm{s}^{-1}$
k_7	0.23	$\mathrm{dm}^3\ \mathrm{mol}^{-1}\ \mathrm{s}^{-1}$
α	600/9	
β	8/23	
A	0.1	$\mathrm{mol\ dm}^{-3}$
B	0.25	$\mathrm{mol\ dm}^{-3}$
H	0.26	$\mathrm{mol\ dm}^{-3}$
C	8.33×10^{-4}	$\mathrm{mol\ dm}^{-3}$
X_o	0	$\mathrm{mol\ dm}^{-3}$
Z_o	0	$\mathrm{mol\ dm}^{-3}$
V_o	0	$\mathrm{mol\ dm}^{-3}$

II. THE MODEL

In this paper we employ the most studied chaotic chemical system—the BZ reaction in a continuous-flow stirred-tank reactor (CSTR). We use the Györgyi-Field [12] model of the BZ reaction:

$$\frac{dX}{dt}=-k_1HXY+k_2AH^2Y-2k_3X^2+0.5[k_4(HA)^{1.5}(C-Z)X^{0.5}-k_5XZ]+k_o(X_o-X), \tag{1}$$

$$\frac{dZ}{dt}=k_4(HA)^{1.5}(C-Z)X^{0.5}-k_5XZ-\alpha k_6VZ-\beta k_7BZ+k_o(Z_o-Z), \tag{2}$$

$$\frac{dV}{dt}=2k_1HXY+k_2AH^2Y+k_3X^2-\alpha k_6VZ+k_o(V_o-V), \tag{3}$$

where

$$Y=\frac{\alpha k_6ZV}{k_1HX+k_2AH^2+k_o} \tag{4}$$

and the variable X denotes the concentration of HBrO_2, and Z denotes the concentration of Ce^{4+}, V the concentration of bromomalonic acid, and Y the concentration of Br^-. The concentrations of these species in the inflow stream are denoted with subscript o. The parameter A represents the concentration of HBrO_3, B is the concentration of malonic acid, and C is the total concentration of catalyst $C=[\mathrm{Ce}^{4+}]+[\mathrm{Ce}^{4+}]$. Kinetic parameters are denoted α and β [12], and k_1–k_6 are rate constants. In our study we allow small variations of the adjustable control parameter k_o, which denotes the flow rate.

The values of the rate constants and fixed parameters used in our simulations are given in Table I. Györgyi and Field [12] showed that the model displays chaos both at high and low flow rates. In this paper we choose the low flow rate chaotic region to demonstrate the encoding procedure. Low flow rate chaos is found for flow rates in the vicinity of $k_o=3.5\times10^{-4}\ \mathrm{s}^{-1}$ for parameters from Table I (see the bifurcation diagram, Fig. 1 in Ref. [12]). The same model [Eqs. (1)–(4)] and similar parametric conditions have been previously used by Petrov *et al.* [3] to demonstrate the OGY method of chaos control.

Equations (1)–(4) are stiff ODE's and we use a modified semi-implicit Runge-Kutta type fourth order method with automatic step length control for numerical integration.

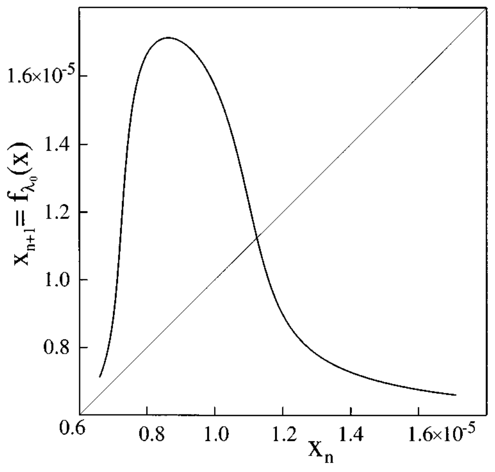

FIG. 1. The one-dimensional map $f_\lambda(x)$ derived from the three-dimensional flow on the surface of the section corresponding to the nominal parameter value $\lambda_0=3.5\times10^{-4}\ \mathrm{s}^{-1}$. The discrete set consisting of many successive piercings of the section can be used to approximate an arbitrary $f_\lambda(x)$ for a point x not in the data set by using a spline fit of the data to represent the map. The technique is equally accessible to experimental data using delay coordinates.

III. LEARNING THE GRAMMAR OF THE SYMBOLIC DYNAMICS

Consider the one-dimensional map

$$x_{n+1}=f_\lambda(x_n), \tag{5}$$

derived from the flow by the Poincaré section. We use a special case of Poincaré section—maxima of Z, which can be easily determined in experiments. The adjustable parameter used for control, the flow rate k_o, is written here as λ. The map corresponding to the nominal parameter value $\lambda_0=3.5\times10^{-4}\ \mathrm{s}^{-1}$ is shown in Fig. 1. The link between chaotic evolution of phase space trajectories in Eq. (5) and orbits of the Bernoulli shift map on symbol space is well established [5].

In the case of a one-hump map, where $f_\lambda(x)$ has a single interior maximum or minimum, a two-character symbol space Σ is sufficient to describe the dynamics of Eq. (5). Given a decision point d, in the range of f_λ, a phase space trajectory, starting at the initial condition x_0, corresponds to a symbol sequence as follows:

$$\sigma_i=\begin{cases}0 & \text{if } x_i\in L, \text{ where } L=[x_{\min},d]\\ 1 & \text{if } x_i\in R, \text{ where } R=(d,x_{\max}].\end{cases}\tag{6}$$

A symbol sequence $\{\sigma_i\}_{i=0}^{\infty}$ may be written

$$\sigma=\sigma_0.\sigma_1\sigma_2\sigma_3\cdots,\tag{7}$$

which can be thought of as a point $\sigma\in\Sigma$, where Σ is the space of all possible two-character, one-sided symbol sequences. Alternatively, the symbol point σ, corresponding to an initial condition x_0, can be thought of as the itinerary of successive left-right positions of the trajectory relative to the decision point d.

The Bernoulli shift map $s:\Sigma\rightarrow\Sigma$ is defined

$$s(\sigma)=s(\sigma_0.\sigma_1\sigma_2\sigma_3\cdots)=\sigma_1.\sigma_2\sigma_3\sigma_4\cdots.\tag{8}$$

As the decimal is shifted to the right one symbol, the left-most symbol is forgotten, as a new symbol is brought into focus. Bringing new symbols into ''focus'' is equivalent to creating information, which we control by small parameter perturbations, $\delta\lambda$. The expression ''in focus'' describes measurement accuracy, which is made rigorous by equipping the symbol space Σ with the following norm:

$$\|\sigma\|=\sum_i \frac{\sigma_i}{2^i}.\tag{9}$$

The shift map on the full symbol space, $s:\Sigma\rightarrow\Sigma$, defines the *fullshift*. However, given an arbitrary map f_λ, not all symbol sequences correspond to the trajectory of an initial condition x_0. Restricting the shift map to a subset of Σ consisting of all the itineraries that are generated by Eq. (6) yields the *subshift* $\Sigma_{(f_\lambda,d)}\subset\Sigma$. The location of the decision point $d\in[x_{\min},x_{\max}]$ effects the set of admissible symbol sequences. For example, if d is chosen far to the right, symbol sequences tend to consist mostly of ''0's.'' We choose d as the interior maximum point, which maximizes the topological entropy of the resulting subshift [6].

Equipped with the topology induced by the symbol space norm, the dynamics of the subshift $s|_{\Sigma(f_\lambda,d)}$ is semiconjugate to the dynamics of the map, $f_\lambda|_\Lambda$ in Eq. (5), restricted to its invariant set $\Lambda\subset[x_{\min},x_{\max}]$. Therefore, trajectories correspond to digital symbolic codes, which can be controlled by parameter perturbations. Since we wish to use only small parameter perturbations, $\delta\lambda$, the goal is to work within the existing dynamics, rather than to create new dynamics with large or rude parameter variations. Therefore, we will work within the existing grammatical limitations of the subshift corresponding to f_{λ_0}, for the chosen nominal parameter value λ_0.

The one-hump map f_λ is not uniformly hyperbolic; specifically, the zero slope at the critical point d, chosen as the maximum, corresponds to nonhyperbolicity for d and its orbit. The partition defines bins, corresponding to n-bit words. The bins are generated by the sequence of inverse images $\{d,f^{-1}(d),f^{-2}(d),\ldots,f^{-(n-1)}(d)\}$, when these inverses exist [6]. When the map is not everywhere two onto one, some $f^{-i}(d)$ will not exist, corresponding to illegal i-bit words. A one-hump map is Kupka-Smale complete if it is onto the interval, and for such a map, all possible subsequences correspond to orbits of the map [5]. We can see that the attractor displayed in Fig. 1 folds the interval $[x_{\min},x_{\max}]$ over itself less than exactly twice.

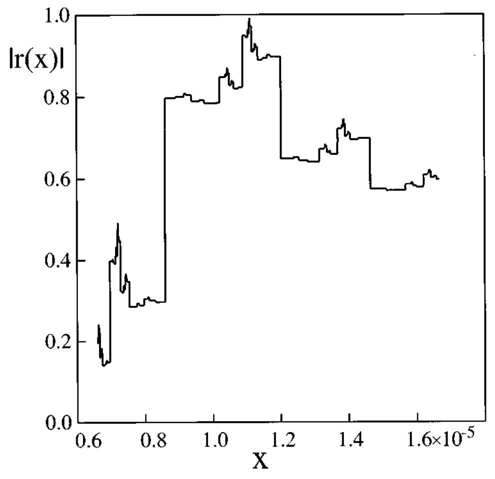

FIG. 2. The norm $\|r(x)\|$ as a function of x.

Learning the grammar of the subshift is a nontrivial task, consisting of finding all the forbidden n-bit words, which are n-symbol sequence combinations. In general, the subshift may not be of finite type; the grammar may not consist of a finite list of n-bit forbidden words for any finite n. However, for our purposes, we do not need the full grammar; a finite approximation will suffice.

With a computer, the goal is to learn an n-bit approximation to the grammar, for $x\in[x_{\min},x_{\max}]$ on a grid. The grid points need to be fine enough to capture the natural partition generated by the bins, with mean size of order $2^{-n}(x_{\max}-x_{\min})$. With this in mind, the bit length n should be chosen proportionally to the minimum experimental resolution $\delta\lambda_{\min}$, approximated according to $\delta x_s \geqslant \max_{x\in[x_{\min},x_{\max}]}\partial f_\lambda/\partial\lambda|_{(\lambda_0,x)}\delta\lambda_{\min}$, where δx_s is the size of the smallest n-bit bin. In practice, the simple rule of thumb is that a larger n-bit word size causes smaller corresponding bin sizes in the phase space, and this requires a higher resolution both in measurements of the phase variable x and in parameter control. So, choosing n too large may require perturbations or measurements beyond the accuracy of the experiment. The payoff is that smaller perturbation is required to encode the message for larger n. In the numerical experiment, described in Sec. VII, we let $n=4$ corresponding to a total of $2^n=16$ possible words, and we use $N=1000$ evenly spaced grid points.

For each grid point y_j, a symbolic itinerary is generated according to Eq. (6), creating an explicit correspondence $\sigma=r(x)$, where $r:[x_{\min},x_{\max}]\rightarrow\Sigma$. Figure 2 displays the norm of these itineraries as a function of x using Eq. (9). Due to the continual refolding of the interval into itself, the function is not monotonic. It is more useful to use the gray-code ordering according to the following formulas, which can be found in Cvitanovic *et al.* [7]. Given σ_i from Eq. (6), define

$$c_k=\left(\sum_{i=1}^{k}\sigma_i\right)(\mathrm{mod}2)\tag{10}$$

and

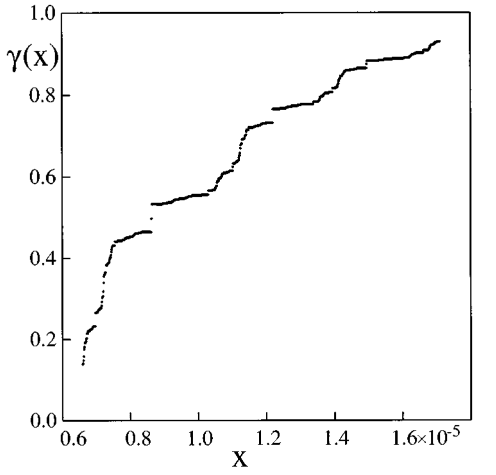

FIG. 3. The monotonic nondecreasing $\gamma(x)$ function due to the gray-ordered symbolic codes. Note that since no symbolic code can have a value $\gamma > \gamma(x_{\max})$, the gray-ordered value of the maximum of the map, a hole must be removed from the function. Likewise removing all preimages removes a Cantor set.

$$\gamma = 0.c_1c_2\cdots = \sum_{k=1}^{\infty} c_k 2^{-k}. \tag{11}$$

The order of γ is the same as the ordering on the interval $[x_{\min}, x_{\max}]$, which causes the monotonic nondecreasing nature of the $\gamma(x)$ function, in Fig. 3. By the natural ordering of the gray-code, $\gamma(x) \in [\gamma_{\min}, \gamma_{\max}]$, where $\gamma_{\min} = \gamma(x_{\min})$ and $\gamma_{\max} = \gamma(x_{\max})$. Thus the grammar must have limitations since the function $\gamma(x)$ does not pass through the origin, as seen in Fig. 3. Therefore, any n-bit symbol sequence, corresponding to a gray-code γ, such that $\gamma < \gamma_{\min}$, does not correspond to a trajectory of f_λ.

We describe here a method of learning an n-bit approximation to the full grammar. The goal is to catalogue all of the n-bit words, and transitions between these words, which are observed among the itineraries of the grid points. This can be decided by a single pass, yes or no, flagging algorithm for each n-bit word. The n-bit approximation to the grammar can be visualized as a 2^n node directed graph (see Fig. 4). Each node represents one n-bit word, and each node can have at most two arrows leading into it and two arrows leading out of it, corresponding to the choice of shifting in a ''0'' or a ''1.'' The n-bit approximation to the grammar is equivalent to a $2^n \times 2^n$ transition matrix A, where an allowed transition from node j to node i is denoted by $A_{i,j} = 1$, and the disallowed transition is denoted by $A_{i,j} = 0$. However, A is necessarily sparse, and requires at most $2n$ computer checks against the grid to decide if the transition occurs, and at most a $2 \times n$ array for storage. The topological entropy, calculated as the natural logarithm of the largest eigenvalue of the corresponding transition matrix [8], is equal to $h = \ln 2$ for the maximal case representing an n-bit approximation of the fullshift grammar.

For an arbitrary grammar, there will tend to be forbidden n-bit words. Hence, the corresponding node is forbidden, which also erases the arrows both leading into and out of the node. Thus, in terms of the transition matrix representation,

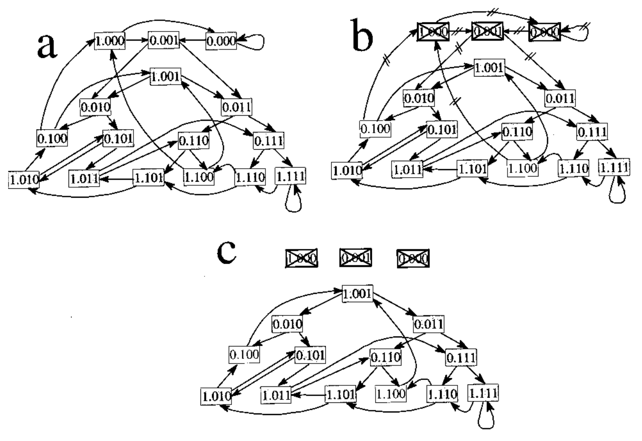

FIG. 4. (a) The fullshift grammar (no words are forbidden) on the two symbols ''0'' and ''1''; all permutations of two symbols, four at a time, and all possible Bernoulli-shifts, to 4-bit word accuracy. The only restriction on transitions is that one application of the Bernoulli shift map can only shift in a ''0'' or a ''1'' in a single iteration. Thus, there are two arrows into and out of each node. (b) A subshift which is of finite type and can be represented by the forbidden word ''000'' requires that all nodes representing words with three ''0's'' in a row must be canceled. To maintain the closed and invariant property of the subshift, all arrows into and out of these nodes must also be eliminated. (c) The resulting subshift has a smaller directed graph. Some nodes, such as ''0.100,'' have no information carrying capacity. At this node, only a ''1'' can be shifted in; $s(0.100) = 1.001$ is the only possibility, because $s(0.100) = 1.000$ is a forbidden word. Thus, node ''0.100'' is ''dead,'' and so the bandwidth is not full.

erasing the ith node requires that the ith row and column are effectively erased from A. The topological entropy is less than the maximal value, ln2, corresponding to diminished information carrying capacity. For example, Fig. 4(b) depicts a directed graph in which the 3-bit word ''000,'' three zeros in a row, is forbidden. Hence, the forbidden words include ''0000,'' ''0001,'' and ''1000.'' These are all the 4-bit words, which include ''000.''

It follows that from the node ''0100,'' there is no choice in what node can follow. In 4-bit accuracy ,''1001'' *must* follow by requirement of the grammar. In terms of the shift map

$$s(0.100\cdots) = 1.001\cdots, \tag{12}$$

written to 4-bit accuracy. At this node, only the single transition, shifting in a ''1,'' can occur, and hence there is no information carrying capacity at the ''0100'' node.

In contrast, the node ''0010'' ends with ''10'' and does not have the possibility of a transition to the forbidden ''000'' sequence in one application of the shift map. Hence, ''0010'' is an information bearing node; there are two arrows leading out of ''0010,'' one to ''0100,'' and one to ''0101.'' In terms of the shift map,

$$s(0.010\cdots) = 0.100\cdots \quad \text{or} \quad 0.101\cdots \tag{13}$$

is permitted by the grammar. The directed graph, or equivalently the transition matrix, represents the n-bit approximation of the grammar, which consists of all the allowed tran-

TABLE II. The encoded message.

Character	ASCII Decimal	Binary
C	67	1000011
h	104	1101000
a	97	1100001
o	111	1101111
s	115	1110011
	32	0100000
i	105	1101001
s	115	1110011
	32	0100000
u	117	1110101
s	115	1110011
e	101	1100101
f	102	1100110
u	117	1110101
l	108	1101100
!	33	0100001

sitions between all the n-bit words. For a finite-type grammar, there exists a finite n such that all forbidden words are considered.

At this point, we have access to all possible transitions; the grammar is recorded as the transition matrix in compact form by the $2\times n$ array of yesses or nos. We also have an itinerary at each grid point y_i. Thus, we know the grammar, and we know the code at each grid point. We still need to learn how to encode a desired bit at each grid point.

IV. ENCODING AND DECODING A MESSAGE

A. Encoding

The directed graph represents the n-bit grammar of the grid points in which a digital message can be encoded according to the following scheme. Suppose we wish to send the message from Table II. Note the distinction between the n-bit register (node) and the 7-bit ASCII codes. The message, in ASCII, is coded bit by bit into the dynamics as the transitions from information bearing nodes.

Starting at an arbitrary initial condition x_0, a simple search locates the nearest grid point y_j. We associate to x_0 the node $r(y_j)$ [which we know as the course grained n-bit itinerary, $r:[x_{\min},x_{\max}]\to\Sigma$ according to Eq. (6)] corresponding to y_j, along with the node's permitted transitions. Throughout the on-the-fly control experiment, the orbit of x_0 will be stabilized to grid points, for insurance of knowledge of the outcomes.

Examining the message in Table II. the first message bit is a ''1.'' If the node corresponding to y_j is information bearing, then the ''1'' can be encoded. In this case, if the code of the natural image of y_j contains in the least significant position a ''1'' bit, which is the desired bit for encoding the message, then y_j should be stabilized to its natural image—to the grid point y_j' closest to $f_{\lambda_0}(y_j)$. If the natural image of y_j corresponds to encoding a ''0'' bit, then by construction there exists a nearby grid point y_k such that the grid point y_k' closest to $f_{\lambda_0}(y_k)$ corresponds to bearing the opposite of the natural bit—a ''1'' bit. Since y_k can be found nearby y_j, a *small parameter perturbation* can be used to stabilize x_0 to the grid point y_k' nearest $f_{\lambda_0}(y_k)$.

Alternatively, if the node corresponding to y_j is not message bearing, y_j must be stabilized to the grid point y_j' closest to the natural image, without bearing the desired bit. Nonbearing transitions through the directed graph, must be made whenever a nonbearing node is visited. A nonbearing node is defined as a node which has only one leaving arrow, corresponding to no choice, and therefore no information carrying capacity. The larger the number of nonbearing nodes, the slower the transmission rate, often called a small bandwidth. The transmission rate, or information carrying capacity of the directed graph, is measured by the topological entropy function.

Each grid point y_j has one fixed code, mapped to the n-bit code $r(y_j)$, and therefore there is a closest y_i such that $r(f_{\lambda_0}(y_i))$ has a ''0'' as the least significant digit, if shifting in a ''0'' from node $r(y_j)$ is permitted by the grammar. Likewise, there is a closest y_k such that $r(f_{\lambda_0}(y_k))$ has a ''1'' as the least significant digit, if permitted by the grammar. Thus, it is possible to preprocess the grid. Before any control is done experimentally, we precalculate and record the closest point to grid point y_j which shifts in a ''0,'' and the closest point which shifts in a ''1.'' Thus the grammar at each grid point can be prerecorded, and all the best targets of all the grid points can be stored in a $2\times N$ array, for an N point grid. A huge advantage of prerecording is the speed and simplicity for ''on-the-fly control'' of an experimental orbit of x_0; this only requires association of x_t, at time t, to its nearest grid point y_j, and then targeting y_j's prerecorded target on the grid which transmits the desired bit, at a minimal energy cost. All pertinent data can be stored in a $3\times N$ array, where the entries of the array can be arranged as follows.

(i) The first column, $y_{j,1}$, records the code and/or itinerary of the grid point, $r(y_j)$. Note that forbidden n-bit words will never appear on this list.

(ii) The second column, $y_{j,2}$, contains the grid number of the closest grid point y_i' to the natural image $f_{\lambda_0}(y_j)$, which shifts in a zero. y_i' represents the target whose code $r(y_i')$ has a ''0'' as the least significant digit and it is the grid point which shifts in the ''0,'' while requiring the smallest parameter perturbation. If shifting a ''0'' into the n-bit window is not allowed by the grammar, we record a -1 in this position. It is necessary to transmit a nonmessage bearing buffer bit ''1.''

(iii) The third column, $y_{j,3}$, contains the grid number of y_k', the closest grid point to the natural image $f_{\lambda_0}(y_j)$ which shifts in a 1. If shifting a ''1'' into the n-bit window is not allowed, we record a -1 in this position. It is necessary to transmit a nonmessage bearing ''0.''

Whenever both $y_{j,2}>0$ and $y_{j,3}>0$, indicating $r(y_j)$ is a message bearing node, the desired bit is transmitted, and on the next iterate, the next bit of the message is considered. If $y_{j,2}<0$ or $y_{j,3}<0$, then a nonmessage bearing bit (a ''buffer bit'') is sent, and on the next iterate, the same message bit is again attempted for transmission. It never happens that both

$y_{j,2}<0$ and $y_{j,3}<0$; this indicates a forbidden n-bit word and as mentioned above such a word never appears on the list of *observed* orbits.

B. Decoding

There are two alternative methods by which the controlled time-series signal can be translated back into the original message. Both techniques require translation of the times series into the sequence of n-bit nodes visited, that is, the path through the directed graph. The first method requires that the receiver also has access to all the tools used by the encoder and/or controller. Given the full grid of points y_j and their corresponding symbolic code $r(y_j)$, the experimental time series can be translated to the sequence of n-bit nodes visited, by associating the nearest grid points y_j to the time series x_t at each time t.

The second method is related to the first in that we interpret transitions through the directed graph, except we do not require that the decoder has access to the original N point grid approximating the function $r(x)$. The alternative and more realistic idea is to read the itinerary directly from the controlled time series x_t, by comparison of x_t to the decision point d at each time t, according to Eq. (6). This information is easily translated into the sequence of n-bit nodes visited by considering a sliding n-bit block through the long itinerary sequence corresponding to x_t.

At this point, interpretation of the path through the directed graph into the digital message is the same for both techniques above. We must strip off the ''buffer bits'' which were added during encryption. Given knowledge of permissible transitions through the 2^n node directed graph, the sequence of nodes actually visited can be translated into the encoded message by reading the transitions *only from the information bearing nodes*. Just as was the case during encryption, the decoder reads the choice of shifting in a ''0'' or a ''1'' in terms of the actual transitions away from information-bearing nodes. However, the transitions away from nonbearing nodes must be ignored, effectively stripping the buffer bits that the encoder was required to send.

Encoding a message requires the insertion of a buffer ''1'' every time after two ''0's'' have occurred. So to send the message ''0000,'' it is necessary to send the bits ''001001'' to avoid the illegal three bit word ''000'' ever occurring in the 4-bit sliding block window. Encryption with buffer bits, for an arbitrary n-bit grammar, occurs automatically when encoding by use of the directed graph method. The interpretation of the path through the directed graph decryption described above automatically strips the buffer bits. The method has the advantage that interpreting the grammar is all done automatically on the computer. This saves a large part of the difficulty found in [9] of interpreting the grammar of the dynamical system; it is never necessary to try to ''manually'' deduce any fundamental word restrictions (such as forbidding ''000'') and then to manually insert and then strip buffer bits, on a case by case basis, to prevent the forbidden words. The list of n-bit nodes and their allowed transitions is sufficient for encoding *and* decoding.

C. Noise

The encoding method can be made robust to noise and modeling errors. Errors to the time-series x_t, due to noise, can cause misidentification of location of x_t relative to d when $|x_t-d|$ is small [11]. A ''0'' when $x_t<d$ may be interpreted as ''1'' when a small error causes the reading $x_t>d$. If noise volume δx_n is possible, then it is necessary to avoid targeting the interval $I=[d-\delta x_n, d+\delta x_n]$ and all preiterates of I, $\{I, f^{-1}(I), f^{-2}(I), \ldots\}$, which defines the holes of a Cantor set that are subtracted from the attractor Λ. Note that as the noise gap is increased, the measure of the Cantor set decreases. Also, the restrictions on the grammar increase, and therefore the bandwidth decreases [14]. We did not formally remove such a Cantor set in this work.

We use a pragmatic technique to prevent errors in control from effecting the message. Interpreting the three-dimensional flow as a one-dimensional map causes a small error in predicted control response, which we discuss in the next section. If a small error causes x_t, which was targeted at $y_{i'}$, whose code is $r(y_{i'})$, to land at y, and $r(y)\neq r(y_i')$, then there is a coding error. If y_i' is near the boundary $\partial S_i'$ of the connected set $S_i'=\{z:r(z)=r(y_i')\}$, the set of all points with the same code as the desired target, then even a small targeting error will be prone to coding errors. The simple fix to this problem is to avoid targeting points near the boundaries of code regions. Simply put, we define a buffering size of m-grid points, and we avoid targeting a point which is less than m points from the boundary of a code region. That is, we add the following caveat when forming the prerecorded $3\times N$ list of targets: $y_{j,2}$ lists the grid number of the optimal energy target, from y_j, to transmit a ''0.'' If $y_{j,2}$ is more than m grid points from the boundary of its code region, then it is already noise error resistant. But if $y_{j,2}$ is less than m grid points from the boundary, then we store instead the noise-resistant and nearby point which is exactly m grid points from the boundary, in the array location $y_{j,2}$. We similarly alter the third column of the array, $y_{j,3}$, of optimal energy targets which shift in a ''1,'' by adding an m-grid buffer region and therefore error resistance.

V. TARGETING

Knowledge of the symbol dynamics of a dynamical systems, to an n-bit accuracy, is equivalent to a complete course grained knowledge of all possible orbits. Controlling the symbol dynamics can also be used to quickly steer orbits to desired periodic states while using only small perturbations. Targeting is surprisingly straightforward, given complete knowledge of the n-bit grammar.

Suppose we wish to target a fixed point. This can be achieved by identifying a legal repeating sequence of bits. For example, we continue with the 3-bit example in which ''000'' is forbidden. In this case, a legal repeating sequence ''111111. . . '' corresponds to the period-1 orbit on the attractor. In this example, it is not possible to target a periodic point ''00000 . . . '' by restriction that three ''0's'' in a row are forbidden. Figure 1 shows that the function $f_\lambda(x)$ intersects the diagonal $x_i=x_{i+1}$ only at $x>d$ and there is no intersection for $x<d$.

Similarly, if we wish to target a period-2 state, it is sufficient to feed any legal alternating two bit sequence, i.e., ''01010101 . . . .'' Legal period-3 three sequences include ''001001001 . . . ,'' ''110110110 . . . ,'' etc.

Note that we never hit a periodic state exactly using

n-bit word accuracy. Rather, we produce an orbit which is stabilized around the periodic orbit by confining it to the bins whose sizes are specified by the n-bit word accuracy. This can be made as accurate as justified by the minimum possible experimental parameter perturbations. This limitation is not unique to the symbol dynamics method of targeting, but limits all methods of course-grained control. An analogy is a person's ability to stabilize a stick in the upwards vertical position. The limitation here is a person's hand-eye reaction time, and minimal hand movement tolerances.

VI. CONTROL PERTURBATIONS

In this section, we discuss the calculation of parameter variations based purely on the assumption of a one-dimensional map. Then we include the complications which arise due to the true three-dimensional nature of the phase space.

In terms of the map based description, given a desired displacement δx_{want}, the parameter variation $\delta\lambda$ can be calculated by linearization of the map, f_λ. Suppose the current, on-the-fly experimental value for phase point on the surface of sections is x. We have determined by above considerations that we need to stabilize the iterate of x to the grid point y_j. The uncontrolled iteration of x is

$$x' = f_{\lambda_0}(x). \tag{14}$$

Therefore, the desired displacement is

$$\delta x_{\mathrm{want}} = y_j - x', \tag{15}$$

from which we can make the following linearization of the map:

$$\delta x_{\mathrm{want}} \approx \left.\frac{\partial f_\lambda}{\partial\lambda}\right|_{(\lambda_0,x)} \delta\lambda. \tag{16}$$

Solving for the required parameter perturbation yields

$$\delta\lambda = \frac{\delta x_{\mathrm{want}}}{\left.\frac{\partial f_\lambda}{\partial\lambda}\right|_{(\lambda_0,x)}}. \tag{17}$$

Parameter control requires knowledge of how the map varies with respect to parameter variations. To calculate the one-dimensional map changes for a slightly perturbed parameter value at least two separate data sets are required: one at the nominal parameter value λ_0, and one for a slightly perturbed parameter value $\lambda = \lambda_0 + \epsilon$. Using the difference quotient approximation,

$$\left.\frac{\partial f_\lambda}{\partial\lambda}\right|_{(\lambda_0,x)} \approx \frac{f_{\lambda_0+\epsilon}(x) - f_{\lambda_0}(x)}{\epsilon}, \tag{18}$$

the derivative can be calculated by comparison of the two spline fit maps (see Fig. 5). We employ several both negative and positive values of ϵ and we use the least squares method at every grid point y_i to approximate the derivative $\partial f_\lambda/\partial\lambda$. Squares in Fig. 6 correspond to the calculated values of $f_{\lambda+\epsilon}$ at the grid point $x = 8.006\,489\,87\times10^{-6}$ and the solid line represents the least squares fit.

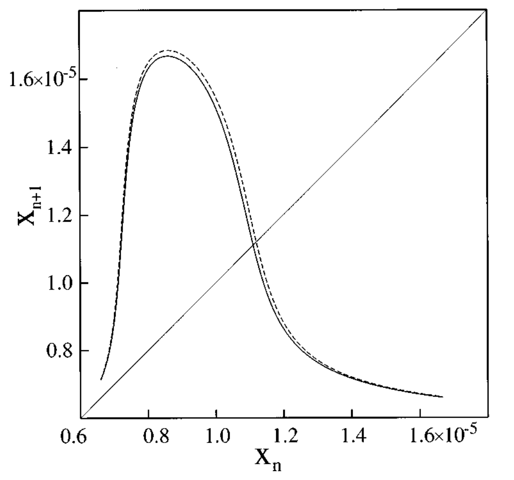

FIG. 5. A spline fit model of the map for the nominal parameter value $f_{\lambda_0}(x)$ as in Fig. 1, and a second spline fit of the map $f_{\lambda_0+\epsilon}(x)$ collected for the parameter variation $\epsilon = 1.4\times10^{-6}$ s-1, $\lambda_0 = 3.5\times10^{-4}$ s-1.

Unfortunately, for our system, the derivative from Eq. (18) cannot be used directly in Eq. (17) to estimate the required parameter perturbation. Equation (17) requires that the response to parameter variations occurs completely along the surface of the section. In fact, the set of three ODE's [Eqs. (1)–(4)], reduced to a one-dimensional map by the Poincaré method, displays transient effects of parameter perturbations. Integration reveals that the very next intersection with the Poincaré surface after a parameter perturbation does not correspond to a perturbed one-dimensional map, but all the following intersections settle onto the perturbed map.

To calculate the true response, on the Poincaré surface, we need to calculate the partial derivatives $\partial f^1/\partial\lambda$ at every point of the nominal one-dimensional map by integrating Eqs. (1)–(4) to find the very next Poincaré section for the slightly perturbed parameter values $\lambda_0+\epsilon$. Let $f^1_{\lambda_0+\epsilon}(x)$ be

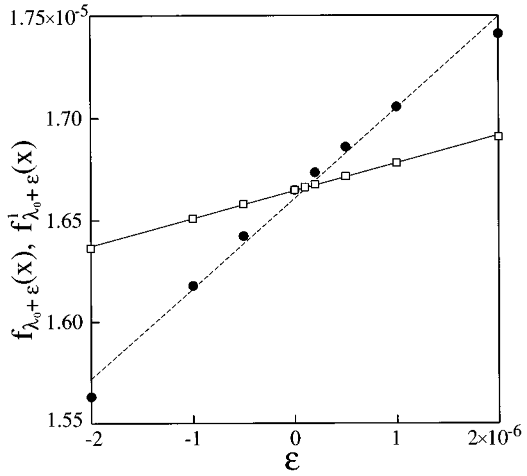

FIG. 6. Dependence of $f_{\lambda+\epsilon}$ and $f^1_{\lambda+\epsilon}$ on ϵ at grid point $x = 8.006\,489\,87\times10^{-6}$ M. Squares and the solid line represent $f_{\lambda+\epsilon}$; dots and the dashed line represent $f^1_{\lambda+\epsilon}$.

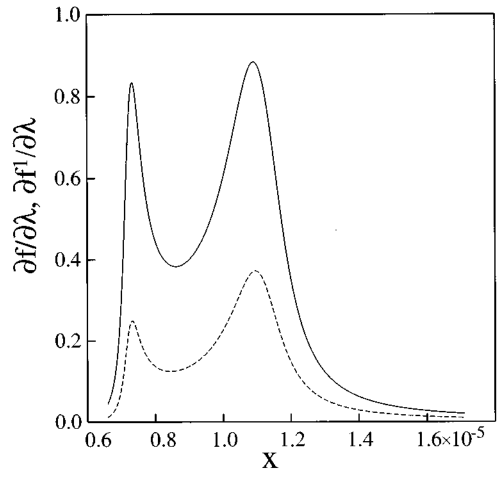

FIG. 7. Comparison between $\partial f_\lambda/\partial\lambda$ and $\partial f^1_\lambda/\partial\lambda$. The approximate partial derivative $\partial f_\lambda/\partial\lambda$ calculated by Eq. (18), under the assumption of a truly one-dimensional map, and responses to parameter variations occur entirely on the Poincaré surface. The true response $\partial f^1_\lambda/\partial\lambda$ variation is calculated by direct integration of the differential equations, with a slightly perturbed parameter, for initial conditions on the original map until the next surface piercing. The nominal parameter value $\lambda_0=3.5\times10^{-4}\mathrm{s}^{-1}$.

the very next Poincaré section after a parameter variation by ϵ. Then the approximation of $\partial f^1/\partial\lambda(x)$ is

$$\left.\frac{\partial f^1_\lambda}{\partial\lambda}\right|_{(\lambda_0,x)}\approx\frac{f^1_{\lambda_0+\epsilon}(x)-f_{\lambda_0}(x)}{\epsilon}. \tag{19}$$

We also employ several negative and positive values of ϵ and we use a least squares fit at each grid point y_i to approximate the derivative $\partial f^1_\lambda/\partial\lambda$. Figure 6 also displays the dependence of $f^1_{\lambda+\epsilon}$ on ϵ at the grid point $x=8.006\,489\,87\times10^{-6}$ and Fig. 7 contrasts the difference between the partial derivatives $\partial f/\partial\lambda$ and $\partial f^1/\partial\lambda$.

The required parameter perturbation is therefore equal to

$$\delta\lambda=\frac{\delta x_{\text{want}}}{\dfrac{\partial f^1_\lambda}{\partial\lambda}(x)}. \tag{20}$$

Another complication, as noted in [13], arises from the false assumption that system dependence on the parameter is given by $x_{n+1}=f(x_n,\lambda_n)$, which for a fixed parameter value gives rise to the one-dimensional map. In fact, the dependence we observe is $x_{n+1}=f(x_n,\lambda_n,\lambda_{n-1})$, due to rapid contraction to a new perturbed one-dimensional map, but not fast enough during frequent parameter variations. Due to the three-dimensional nature of the flow, Eqs. (14), (15), and (20) give the correct estimate for the required parameter perturbation only when the adjustable parameter is changed from λ_0 to $\lambda_0+\delta\lambda$.

The control of symbol dynamics requires slight changes to the adjustable parameter at *every* iteration and therefore there is no time to settle back onto the one-dimensional map of the nominal parameter value. Therefore, we cannot use Eq. (14) to correctly predict the next uncontrolled iteration of x when the current value of the adjustable parameter is not λ_0 but $\lambda_n=\lambda_0+\delta\lambda_n$. We utilize here the fact that after a parameter variation all iterations except the first one settle onto the perturbed map. Assuming that we do not change λ_n during the next iteration, then the uncontrolled iteration of x_n is described by the perturbed one-dimensional map,

$$x'=f_{\lambda_n}(x_n). \tag{21}$$

We estimate $f_{\lambda_n}(x_n)$ by linearization of the map around the nominal value with the derivative from Eq. (18),

$$f_{\lambda_n}(x_n)=f_{\lambda_0}(x_n)+\delta\lambda_n\frac{\partial f_\lambda}{\partial\lambda}(x_n). \tag{22}$$

We employ Eqs. (15), (21), and (22) to determine the desired displacement and Eqs. (19) and (20) to calculate the required perturbation at every iteration.

TABLE III. Encoding of the letter C.

	$Z_{\max}\times10^6$		Relative	Required	Code	Bit	Bit out	Bit	Bit in
Initial	Desired	Actual	error (%)	$k_0\times10^4$		out	type	in	type
13.3982	7.4385	7.4449	0.086	3.5000	0101	1	Arbitrary	1	C-1st
7.4449	13.8439	13.8442	0.003	3.4883	1010	0	Arbitrary	0	C-2nd
13.8442	7.6594	7.7581	1.289	3.4885	0100	1	Arbitrary	0	C-3rd
7.7581	15.9896	16.0028	0.083	3.5423	1001	0	Arbitrary	1	Nonbear
16.0028	6.7654	6.7653	−0.001	3.5295	0010	1	C-1st	0	C-4th
6.7653	7.6594	7.6599	0.007	3.5189	0100	0	C-2nd	0	C-5th
7.6599	15.5794	15.5873	0.051	3.5180	1001	0	C-3rd	1	Nonbear
15.5873	6.6076	6.6437	0.546	3.5126	0011	1	Nonbear	1	C-6th
6.6437	7.2387	7.2381	−0.008	3.4347	0111	0	C-4th	1	C-7th
7.2381	11.1829	11.2101	0.244	3.4517	1111	0	C-5th	1	h-1st
11.2101	11.1829	11.2078	0.222	3.4610	1111	1	Nonbear	1	h-2nd
11.2078	11.4143	11.4311	0.147	3.4749	1110	1	C-6th	0	h-3rd
11.4311	10.5203	10.5085	−0.112	3.4857	1101	1	C-7th	1	h-4th

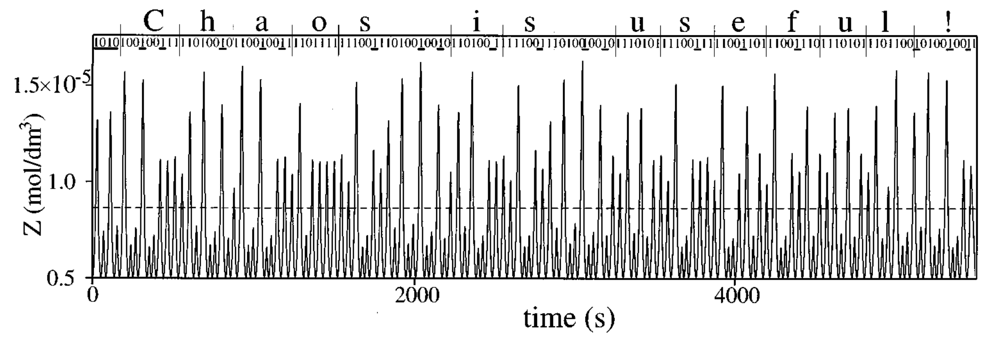

FIG. 8. Controlled ''chaotic'' oscillations of BZ system [Eqs. (1)–(4)] containing the message from Table II. The dashed line indicates the decision point. An oscillation maximum above this line corresponds to a ''1'' bit, and below corresponds to a ''0'' bit. The first four bits are not part of the message, but of the symbolic code of the randomly chosen initial condition. In accordance with the grammar, after every ''00'' sequence, a noninformation bearing ''1'' bit has been inserted. The noninformation bearing bits are underlined.

VII. NUMERICAL EXPERIMENTS

A. Encoding and decoding of a message

The initial parts of controlling chemical oscillations to encode the message in Table II is listed in Table III. The first letter of the message, a ''*C*,'' has the binary representation 1000011. A randomly chosen initial point on the Poincaré section is shown as the first number in the first column of Table III. Because this number is larger than d ($d = 8.637\,56\ldots\times10^{-6}$) the ''Bit out'' column displays bit ''1.'' The first bit of the message is ''1,'' and shifting in a ''1'' is permitted by the grammar, since it does not cause the illegal word, ''000.'' To encode the ''1,'' we need to control the trajectory of the initial condition to the vicinity of the value shown in the column ''Desired,'' on the Poincaré section. The required parameter perturbation is estimated using Eq. (20). The resulting value of the parameter is shown in the column ''Required.'' The next intersection of the perturbed orbit, on the Poincaré surface, is shown in the column ''Actual,'' and the relative error in targeting the desired point is in the ''Relative error'' column. The 4-bit code corresponding to the ''Actual'' point is shown in ''Code'' column, a ''0101.'' The least significant bit of this code contains the first bit of the ASCII message—the ''1'' bit on the far right. After the third iteration, the code is 0100 and the grammar requires that we shift in a ''1'' to prevent the forbidden sequence. Therefore, the mandatory nonbearing bit ''1'' is targeted on the next iteration. The code 1001 results, from which there are no grammatical restrictions. The grammar allows that, from the code 1001, either a ''0'' or a ''1'' can be shifted in and therefore the fourth ASCII bit of the letter ''*C*,'' a ''0'' bit, is encoded next.

To read the message we have to remember that the first four bits belong to the arbitrary initial state and we ignore them. The first encoded bit can be read after it moves from the least significant position to the most significant position—i.e., there is a delay connected with the buffer length. To shift all the binary bits of the letter ''*C*'' through the buffer, we have to start with encoding of the next letter, the ''*h*'' (see the bottom of Table III).

Figure 8 shows the controlled time dependence of the variable Z, which after decoding reveals the full message, ''Chaos is useful!'' The dashed line marks the value of the critical decision point d. According to Eq. (6), maxima above the dashed line represent ''1'' bits, and maxima below represent ''0'' bits. Note that the first four bits do not belong to a message. To move the last bits of the message to the most significant position, four arbitrary bits were added to the end of message.

B. Targeting of periodic orbits

In demonstration of how easily we can target and then stabilize periodic orbits through control of symbol dynamics, we target a fixed point by encoding the message ''111111 . . . ,'' corresponding to the period-1 orbit on the attractor. To target a period-2 state, we feed the register with an alternating two-bit sequence ''01010101 . . . '' and we also target two possible period-3 sequences, i.e., ''011011011 . . . '' and ''001001001 . . . .'' In the last case we actually feed the register with the sequence ''00000 . . . '' and the algorithm automatically adds one bit ''1'' after every two ''0'' bits. Figure 9 shows the orbits and time series for these four examples of the targeting. A period-4 orbit results

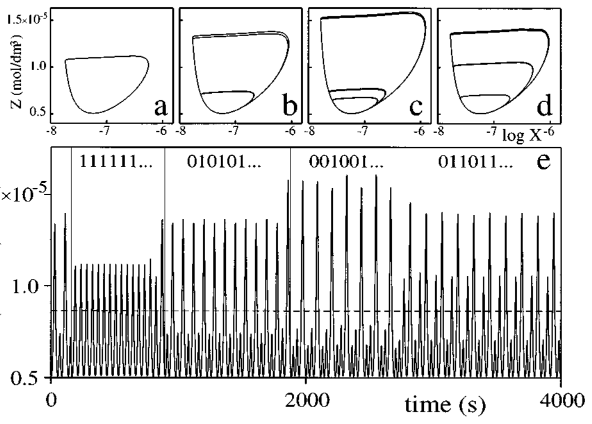

FIG. 9. Stabilization of the periodic orbits by encoding periodic sequences of bits—phase portraits and time series. Phase portrait of targeting (a) period-one oscillations—encoded sequence ''111111 . . . ,'' (b) period-two oscillations—encoded sequence ''010101 . . . ,'' (c) period-three oscillations—encoded sequence ''001001001001 . . . ,'' (d) period-three oscillations—encoded sequence ''110110110110 . . . ,'' (e) time series of the stabilized periodic orbits; the dashed line indicates the decision point.

from ''110011001100...'' or from ''111011101110....'' This is quite general and allows any periodic orbit to be targeted.

VIII. CONCLUSION

We demonstrate that chemical oscillations can be steered, through *small perturbations*, to contain information. The analogy of the BZ reaction to biological systems implies the possibility that living systems might store information in chaotic oscillations. In terms of practical control issues, we demonstrate that complete control over symbol dynamics implies complete control over all permissible orbits; targeting and stabilization of all periodic orbits and arbitrary aperiodic orbits are straightforward.

In general, to encode information in the dynamical system using only small perturbations requires that we respect the grammar of the subshift produced by the free-running chaotic system. A grammar of infinite type is impractical to learn in general. We have demonstrated that due to the correspondence between the physical dynamics and the symbol dynamics, a course-grained parameter control threshold $\delta\lambda_{\max}>0$ only requires that we learn an n-bit approximation of the grammar.

We describe a method for learning an n-bit approximation of the grammar in a format which is well suited for encoding and decoding information. Due to the course-grained control nature of any real experiment, it is only necessary to observe the grammar for a grid of initial conditions. Any initial condition not on the grid can be stabilized to the grid within control tolerance. We have presented a format to precalculate and record a grid of information-bearing targets for compact and fast access during the on-the-fly control procedure.

The grammar learning and control techniques we have developed could be performed for an experimental data set whose dynamics are known only through time-delay embedding, or successive maxima. We have described our technique to observe short orbit segments for a grid of initial conditions on the attractor. However, we have just implemented the more experimentally accessible and equivalent technique of observing a *single* long trajectory which ergodically wanders through attractor, eventually occupying all the bins which the grammar permits. Our current research is pointed towards demonstrating our techniques in the laboratory experiment to control the BZ reaction, and to encode a message.

Our grammar learning techniques are more general than the setting in which we have developed them. Unlike the work of Hayes *et al.* [9,10], which was based in only one dimension, our technique requires no restrictions on the dimension of the oscillator, requiring minimal modification to our computer programs to learn an arbitrary grammar. We are currently exploring a wide variety of intriguing engineering and interdisciplinary applications of controlling symbol dynamics.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation. We would like to thank S. Hayes for many enlightening discussions concerning his pioneering work on encoding information into the chaotic double-scroll oscillator, including noise issues. We would also like to thank C. Grebogi and E. Ott and Y.-C. Lai for related discussions concerning hyperbolicity and ergodicity.

[1] E. Ott, C. Grebogi, and J.A. Yorke, Phys. Rev. Lett. **64**, 1196 (1990).
[2] T. Shinbrot, E. Ott, C. Grebogi, and J.A. Yorke, Phys. Rev. A **45**, 4165 (1992).
[3] V. Petrov, V. Gaspar, J. Masere, and K. Showalter, Nature **361**, 240 (1993).
[4] V. Petrov, B. Peng, and K. Showalter, J. Chem. Phys. **96**, 7506 (1992).
[5] S. Smale, Bull. Am. Math. Soc. **73**, 747 (1967).
[6] J.P. Crutchfield and N.H. Packard, Int. J. Theor. Phys. **21**, 433 (1982).
[7] P. Cvitanovic, G. Gunaratne, and I. Procaccia, Phys. Rev. A **38**, 3 (1988); **38**, 1503 (1988).
[8] C. Robinson, *Dynamical Systems: Stability, Symbol Dynamics, and Chaos* (CRC Press, Ann Arbor, 1995).
[9] S. Hayes, C. Grebogi, and E. Ott, Phys. Rev. Lett. **70**, 3031 (1993).
[10] S. Hayes, C. Grebogi, E. Ott, and A. Mark, Phys. Rev. Lett. **73**, 1781 (1994).
[11] S. Hayes, Ph.D. thesis, University of Maryland, 1995.
[12] L. Györgyi and R. J. Field, Nature **355**, 808 (1992).
[13] T. Shinbrot, W. Ditto, C. Grebogi, E. Ott, M. Spano, and J.A. Yorke, Phys. Rev. Lett. **68**, 2863 (1992).
[14] E. Bollt and Y.-C. Lai (unpublished).