Communication through chaotic modeling of languages

Murilo S. Baptista,[1,2,*] Epaminondas Rosa, Jr.,[3] and Celso Grebogi[1,2,4]
[1]*Institute for Plasma Research, University of Maryland, College Park, Maryland 20742*
[2]*Institute for Physical Science and Technology, University of Maryland, College Park, Maryland 20742*
[3]*Nonlinear Dynamics Laboratory, Department of Physics, University of Miami, Coral Gables, Florida 33146*
[4]*Department of Mathematics, University of Maryland, College Park, Maryland 20742*

(Received 2 March 1999)

We propose a communication technique that uses modeling of language in the encoding-decoding process of message transmission. A temporal partition (time-delay coarse graining of the phase space based on the symbol sequence statistics) is introduced with little if any intervention required for the targeting of the trajectory. Message transmission is performed by means of codeword, i.e., specific targeting instructions are sent to the receiver rather than the explicit message. This approach yields (i) error correction availability for transmission in the presence of noise or dropouts, (ii) transmission in a compressed format, (iii) a high level of security against undesirable detection, and (iv) language recognition.

PACS number(s): 05.45.Vx, 05.45.Gg

I. INTRODUCTION

Recent developments in communicating with chaos [1–4] have produced a wealth of potential practical applications including synchronization [5–7], encoding-decoding techniques [1–4,8–11], noise filtering [12], and signal masking and recovery [13,14]. This is so because chaotic systems have peculiar properties that make them natural candidates to play a significant role in nonlinear communication systems. One of these properties, the sensitivity of the dynamics to small perturbations, is useful for targeting the trajectory in phase space to specific regions to which particular symbols have been assigned. This targeting feasibility provides chaotic systems with a natural type of dynamics to be used in communication. The symbol sequence to be followed by the chaotic trajectory corresponds then to the information to be transmitted [1–4,8,9,11,15]. Indeed, the ergodicity (or the eventual visit of the trajectory to all partitions without any targeting or control) of chaotic systems has been used recently [14] in a chaotic communication scheme.

Symbolization of a chaotic trajectory can be useful for extracting relevant information about the system under consideration. Correlation function computing [16,17], parameter estimation [18], and data compression [19] are examples of symbolic dynamics [20] application toward a better understanding of the system dynamics. Also, different signals generated by the same dynamics can be identified with the help of the conditional entropy [21] obtained from the symbolic dynamics of the chaotic process. Of course, the symbolic sequence generated by a chaotic trajectory depends on how the phase space is partitioned. It also depends on the time delay interval (sampling rate for symbol sequence construction), which has been used to measure correlation lengths from given symbolic sequences [19]. Much emphasis has been placed on the characterization of the complexity of symbol sequences based on patterns and transmission rules estimated from symbolic time series [22].

In this work we present a language approach for a chaotic communication system. The text message to be transmitted is generated by a chaotic process that respects the grammar of a language. Symbols are assigned to judiciously chosen regions of phase space, and the chaotic trajectory is controlled to visit these regions generating a symbol sequence that corresponds to the desired message. The message itself is not transmitted. Rather, what is transmitted is a set of instructions, the codeword, that enables the receiver to decode the message. A temporal partition is introduced as a time-delay coarse graining [19] of the phase space. The phase space is divided into a number of cells to which different symbols are assigned [16,17]. As the chaotic trajectory visits these regions, symbols are generated, producing a symbol sequence that corresponds to a message to be transmitted. The partitions are chosen in such a way that the message is consistent with the grammar of a language. For the purpose of illustration we use an artificial language created as an approximation to a real language in terms of statistical structure. We assume a communication system consisting basically of a transmitter where the message is encoded, a communication channel that carries the message from one place to another, and the receiver where the message is decoded. Transmitter and receiver have complete knowledge about the dynamical system being used. The procedure involves a minimum of information transmission, is secure against unwanted detection, and is robust against noise and dropouts.

This paper is organized as follows. In Sec. II we introduce concepts and definitions related to languages, paying special attention to their statistical structure. In Sec. III, we show how this statistical structure is used in the construction of the dynamical model process. Section IV details how the communication system is built based on language modeling, and a technique for optimal transmission of information is presented in Sec. V. In Sec. VI, we introduce a language recognition scheme and explain how this proposed communication system is secure against undesired decoding. Section VII proposes an error correcting code that is able to recover information when the transmission is corrupted by noise or lost

*Permanent address: Instituto de Física, Universidade de São Paulo, C.P. 66318, 05315-970 São Paulo, SP, Brazil.

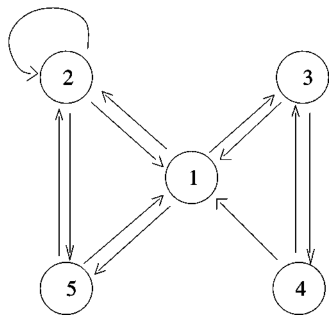

FIG. 1. Transition diagram for the artificial language. The arrows indicate the allowed unit transitions. Notice that some sequences of unit pairs are not allowed, as the sequences ''11'' or ''14,'' for example.

due to dropouts. Our conclusions are presented in Sec. VIII.

II. LANGUAGES

In general terms a language can be defined as any means of communicating. More restrictedly, it can be viewed as a body of words and methods of combining words used and understood by a considerable community. The alphabet of a language is the finite set S of all permitted basic elements or units S_i. The information is contained in the message M which is a sequence of units organized according to the rules of the language, or grammar. We refer to the message as a text because of its alphanumerical character. Suppose, for instance, that the unit S_1 corresponds to the ''blank space unit.'' In this case, a word is a sequence of units in between two units S_1. An example of a two-word message would be $M = S_2S_3S_2S_1S_2S_2S_4S_5$. Every unit along the message can be localized by attaching subscripts to the message units, such as M_i. In this example, $M_1 = S_2$, $M_2 = S_3$, and so on.

One of the basic language rules is related to the way successive units appear in a message. Each unit depends on the preceding units. A graphical visualization of the possible transitions in a language composed of a five-unit alphabet is given by the transition diagram of Fig. 1 (also known as a state machine). It describes the allowed pairs of adjacent units that can possibly appear in a message generated by this model of an artificial language.

Natural languages can be modeled by series of approximations where each approximation models a particular statistical property of the language. The more statistical properties are taken into account, the more approximations are considered, and the more reliable the model is [17]. In general, a zeroth-order approximation model generates messages with units independent of the preceding units: All units are equally probable. A first-order approximation generates messages with units appearing independently of the preceding units, but units have a different frequencies of appearance given by the *unit frequency*. A second-order approximation generates messages with units dependent on the preceding units, units have different unit frequencies, and pairs of units appear with different frequencies given by the *transition frequency*. An *n*th-order approximation generates messages with different frequencies, or probabilities of appearance for each *n*-unit combination. More sophisticated model approximations may require, for example, that every word depend on the preceding ones.

A message composed of units drawn from a finite set S can be viewed as a Markov process that might produce an *ergodic* message. By ergodic we mean that all representative (large enough) sequences of units have almost the same statistical properties (statistical homogeneity). Thus unit frequencies, transition frequencies, and so on, approach definite limits, as the length of the sequences approaches infinity. Two conditions have to be met in order to construct a message with ergodic characteristics. The first condition imposes that every unit has to be reachable from all other units by following some allowed path. The diagram of Fig. 1 shows that this reachability is satisfied by following the paths indicated by the arrows. The second condition requires that the greatest common divisor of the lengths of closed circuits in the diagram of Fig. 1 be equal to one, a condition also satisfied by this diagram.

The artificial language that we use here is simple, created by imposing statistical rules up to second-order approximation. This simple language serves well the purpose of demonstrating the implementation of our approach. We show that it can be modeled by a chaotic mapping, and that the communication procedure allows secure transmission of information in a compressed format, error correction capability and language recognition.

III. MODELING A LANGUAGE

A major characteristic of language is their inner organization. Languages produce messages with finite entropy [23], strongly suggesting that the generation of a message is due to a deterministic dynamical process. To model a language by means of a dynamical system, we look for a one-to-one correspondence between a trajectory and a message created by this model. The correspondence is found by partitioning the phase space where the trajectory evolves in a number of partitions P_i. Thereby, real number trajectory points are associated with a finite set of symbols. With this procedure the phase space is mapped into the symbol space. As the trajectory evolves, visiting the different symbol regions in which the phase space has been partitioned, a symbol sequence is generated. This symbol sequence ξ with components ξ_i is unique and characterizes the particular trajectory that generated it. The correspondence between the language and the dynamical system boils down to the equivalence between messages and symbol sequences. There must exist a one-to-one correspondence between the symbols and alphabet units given by a function W, such that $M_i = W(\xi_i)$. For simplicity we set $M_i = \xi_i$ for all i's.

So, with a message generated by the selected language model, say up to second-order approximations as chosen here, we look for a system (i) with finite positive entropy (typical of deterministic systems), (ii) capable of producing an ergodic symbol sequence, such that (iii) the symbol sequence statistics is similar to the message statistics (transition diagram, unit frequency, and transition frequency). In other words, we are looking for a chaotic system. So far, to the best of our knowledge, given a generic symbol sequence, there is no general technique for finding a system capable of generating such a sequence, or to define the real phase space partitioning. We propose here a new approach for finding the proper partitioning of the state space of a generic chaotic

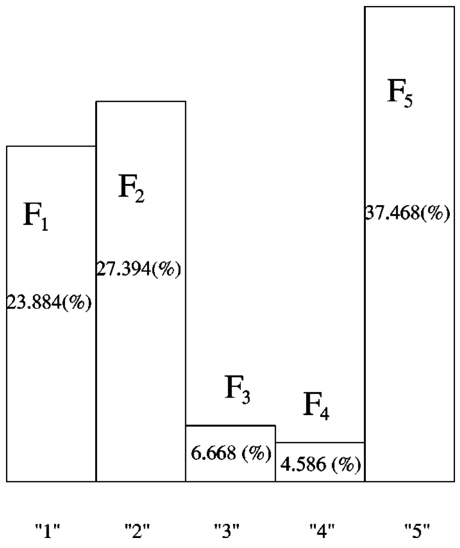

FIG. 2. Unit frequencies of the language to be modeled by the chaotic dynamics system.

system, such that the trajectory of this system is capable of generating the given symbol sequence.

In order to demonstrate how our ideas can be implemented, we create a simple artificial language. It is based on an alphabet composed of five units represented by the numerals 1, 2, 3, 4, and 5, with allowed sequences and groupings governed by the transition diagram of Fig. 1. Notice that some sequences of unit pairs are not allowed, such as the sequences 11 or 14, for example. Then we generate a message randomly choosing among the five units available, discarding sequences that are not allowed. If the random number generator has a uniform distribution, the frequency of units is proportional to the number of arrows with the tip pointing toward the corresponding unit. In natural languages, due to higher levels of organization when creating words and phrases, such a property is not verified. Thus, if we want to mimic a natural language, we need a random number generator with a nonuniform distribution. For this purpose we rescale the random number η, defined to lie within the interval [0,1], by the function $f(\eta)=\eta e^{1.0-\eta}$. This interval is divided in five equal size subintervals. The numbers that fall into the first interval are associated with the unit 1, into the second interval with unit 2, and so on. This rescaling makes the distribution nonuniform and generates numbers within the interval [0,1]. For each rescaled random number, we generate one unit of our message if the character respects the rules imposed by the transition diagram. The message used here contains 50 000 units.

The next step now is to model the artificial language that can generate a 50 000-unit message. Some of its statistical properties are readily available, such as the number of units and the transition diagram. Notice that when modeling a natural language this information must be obtained by inspecting the message itself. Here we need to compute the frequencies of the units F_i, and the transition frequencies, $F_{i,j}$, with $i,j=1,2,3,4,5$. The F_i's of our 50 000-unit message are shown in Fig. 2. Correspondingly, the statistics of the symbol sequence is quantified by the symbol frequency (the frequency of appearance of the *i*th symbol), G_i, and the symbol transition frequency (the frequency of appearance of the *ij* two-symbol sequence) $G_{i,j}$.

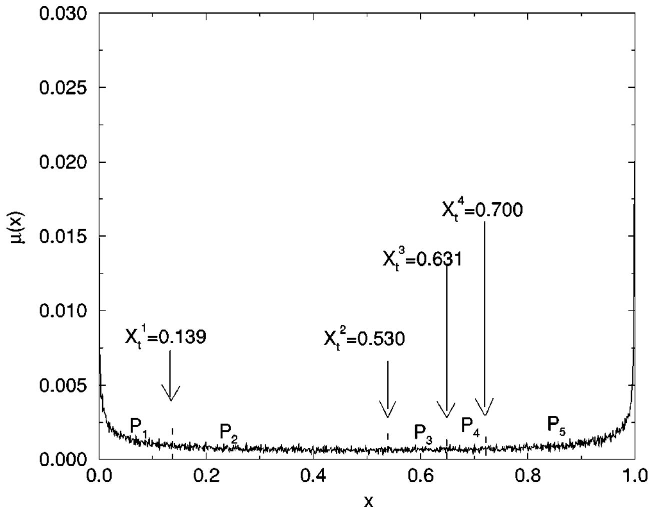

FIG. 3. Natural invariant density of Eq. (1) computed through a finite 50 000-sized trajectory. The limits X_b^i and X_t^i of the partition are also indicated in this figure. The invariant density is independent of the initial condition chosen.

Next, we need a chaotic system. Our choice is

$$X_{n+1}=T(X_n)\equiv bX_n(1-X_n), \tag{1}$$

where we use $b=4$, which yields a chaotic trajectory in [0,1]. The map (1) from now on will be referred to as the transformation *T*.

Recall that in order to have a symbolic sequence (generated by a nonlinear system) equivalent to the message (the unit sequence generated by the artificial language), the symbol frequency G_i, with which the chaotic trajectory visits the partition P_i, must be close to the unit frequency F_i. Given that a certain unit S_i appears with a frequency F_i, the partition whose symbolic description is associated with the character S_i, defined in the interval $]X_b^i,X_t^i]$, is found by validating

$$F_i=\int_{x_b^i}^{x_t^i}\mu(x)dx, \tag{2}$$

where x_b^i and x_t^i are the bottom and the top of the partition P_i, and $\mu(x)$ is the natural invariant measure of Eq. (1). This is shown in Fig. 3, obtained from a 50 000-long trajectory. We numerically calculate Eq. (2) breaking it in a sum computed over intervals of length 0.001. In this work, $x_b^1=0$ and $x_t^5=1.0$, corresponding to the valid range for Eq. (1), and $x_t^i=x_b^{i+1}$. The five partitions whose locations are indicated in Fig. 3 are defined to lie in the intervals: $]x_b^1,x_t^1]=]0.000,0.139]$, $]x_b^2,x_t^2]=]0.139,0.530]$, $]x_b^3,x_t^3]=]0.530,0.631]$, $]x_b^4,x_t^4]=]0.631,0.700]$, $]x_b^5,x_t^5[=]0.700,1.000[$. Defining these five partitions (see Fig. 3), the generated symbolic sequence of Eq. (1) for a typical trajectory, has symbols P_i with the same frequency as the frequency of the units S_i (see Fig. 2). Thus, $F_i=G_i$ (in practice, numerically we find $F_i\cong G_i$).

In our modeling process we now need to obtain a symbolic sequence with a symbolic transition frequency $G_{i,j}$ equal to the unit transition frequency $F_{i,j}$. This means $G_{i,j} = F_{i,j}$ (in practice, $G_{i,j} \cong F_{i,j}$). Even though the symbol sequence of Eq. (1), with partitions defined by Eq. (2), follows the same unit frequencies of the message, it is natural to expect that such a sequence does not respect the transition diagram, and also that the symbol transition frequencies $G_{i,j}$ are typically very different from the unit transition frequencies $F_{i,j}$. For example, a point in the partition P_1 does not go to the partition P_3 under the application of Eq. (1), as required for the existing equivalence between the symbol sequence and the message. However, we know that there are points in P_1 that go to P_3 after a certain number of iterations different from one. Thus, in order to obtain a symbol sequence consistent with the second level approximation of our artificial language, we define subpartitions $P_{i,j}$ such that a trajectory visiting them respects the transition diagram. Besides, these subpartitions must be spatially localized such that the trajectory through them generates a symbolic sequence for which $F_i = G_i$ and $F_{i,j} = G_{i,j}$.

Two conditions are in order at this point. First, $P_{i,j}$ is contained in the partition P_i, and

$$T^{n_{i,j}}(P_{i,j}) \rightarrow P_j, \tag{3}$$

meaning that the $n_{i,j}$th iteration of the region of the subpartition $P_{i,j}$ is mapped into the partition P_j. So, a point in $P_{i,j}$ generates a trajectory under the application of $T^{n_{i,j}}$ that represents the transition i j. Second,

$$\sum_i F_{i,j} = F_j; \quad \sum_i G_{i,j} = G_j, \tag{4}$$

meaning that the sum of all probabilities of transitions that go to the unit (or symbol) j must be equal (or close to) the unit (or symbol) frequency F_j (or G_j). In practice, Eqs. (4) are approximately satisfied. Then, the subpartitions are obtained by satisfying the following integral equation:

$$\int \mu(P_{i,j})dx = F_{i,j}, \tag{5}$$

which defines an interval with a probability of having a typical trajectory falling within it equal to the transition probability $F_{i,j}$. Even though different intervals in the attractor obey condition (3), we choose the subpartition $P_{i,j}$ to be the interval that belongs to the partition P_i.

To find the subpartition $P_{i,j}$, we compute the number of iterations $n_{i,j}$ for which a large collection of points go from the partition P_i to the partition P_j, following a trajectory that is 50 000 long. Once this number of iterations is found, we solve Eq. (5) numerically. We break the integral over a sum of terms that measure the probability density of Eq. (1) for small intervals of size 0.005. Notice that the subpartitions are defined in terms of number of iterations $n_{i,j}$ which explains their being named temporal. Two types of subpartitions are obtained depending on how we choose the numbers $n_{i,j}$. One type is obtained by choosing $n_{i,j}$ such that the

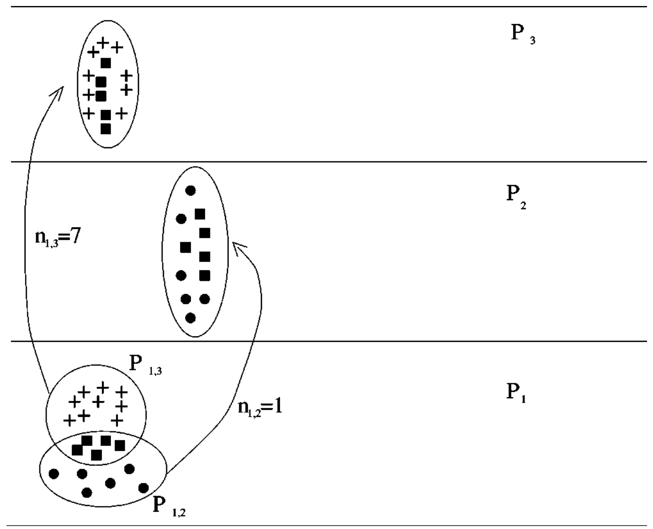

FIG. 4. Illustration of how points from the same partition can go to different partitions. A collection of points that are in P_1 go to P_2 in $n_{1,2} = 1$ iterations (full circles), and a collection of points that are in P_1, go to P_3 in $n_{1,3} = 7$ iterations (crosses). The squares represent points that go to either P_2 or P_3.

collection of points is maximum. Another type is obtained by choosing the same $n_{i,j}$ for subpartitions located in the same partition, and this $n_{i,j}$ must be such that we still find in that trajectory a large collection of points going from P_i to P_j. The subpartitions of the first type are composed of a union of a few closed intervals, while in the second type subpartitions are formed by only one closed interval.

An important aspect of the subpartitions refers to the successful recovering of the message at the receiver. This means that two subpartitions belonging to the same partition should not overlap, that is, $P_{i,j} \cap P_{i,j'} = \emptyset$. When obtaining the subpartitions, $P_{i,j}$ and $P_{i,j'}$, for $n_{i,j} \neq n_{i,j'}$, we might find subpartitions that overlap. For example, in Fig. 4, a collection of points that are in P_1 go to P_2 in $n_{1,2} = 1$ iterations (full circles), and a collection of points that are in P_1, go to P_3 in $n_{1,3} = 7$ iterations (crosses). The squares represent points that go to either P_2 to P_3. In such a case, the region that contains the square shaped points is considered to be part of either $P_{1,3}$ or $P_{1,2}$. If we restrict the choices for the values of $n_{i,j}$ such that $n_{i,j} = n_{i,j'}$, we avoid the overlapping of subpartitions. This can be seen more clearly by doing the following. We know that $T^{n_{i,j}}(P_{i,j})$ is mapped onto P_j. Now we want to find the subpartition $P_{i,j'}$ such that $T^{n_{i,j'}}(P_{i,j'})$ is mapped onto $P_{j'}$. If $n_{i,j} = n_{i,j'}$ and $P_{j'} \cap P_j = \emptyset$, thus $P_{i,j'} \cap P_{i,j} = \emptyset$. Even though we worked with subpartitions in the same partition constructed by considering either different or equal n's, from now on we present results obtained by constructing subpartitions in the same partition but for which $n_{i,j} = n_{i,j'}$. In this way, we avoid difficulties when communicating using a subpartition that is not connected.

These subpartitions provide the temporal characteristic of our partition. In fact, the phase space coarse graining is based on the statistical properties of a time-delay symbolic sequence, expected to be equivalent to the message. In our example here the subpartitions are given by the following intervals:

$$\begin{aligned}
P_{1,2}&=[9.110959390093726\text{E-}003,\ 4.091426526520488\text{E-}002],\\
P_{1,3}&=[4.099086504290043\text{E-}002,\ 5.170528142116289\text{E-}002],\\
P_{1,5}&=[6.015706173586438\text{E-}002,\ 0.138938311406073],\\
P_{2,1}&=[0.405160149850449,\ 0.529882153383280],\\
P_{2,2}&=[0.301807100421237,\ 0.404999984676375],\\
P_{2,5}&=[0.139035266963177,\ 0.262229143759187],\\
P_{3,1}&=[0.530069064408815,\ 0.547671906102650],\\
P_{3,4}&=[0.613756404263279,\ 0.622613674593072],\\
P_{4,1}&=[0.669097773904945,\ 0.699922529656951],\\
P_{4,3}&=[0.637208684765797,\ 0.643325562499964],\\
P_{5,1}&=[0.963978076071563,\ 0.999999995867970],\\
P_{5,2}&=[0.842835294422986,\ 0.963940926475781],
\end{aligned} \tag{6}$$

and the iteration numbers are

$$\begin{gathered}
n_{1,2}=n_{1,3}=n_{1,5}=2, \quad n_{2,1}=n_{2,2}=n_{2,5}=2,\\
n_{3,1}=n_{3,4}=3, \quad n_{4,1}=n_{4,3}=4, \quad n_{5,1}=n_{5,2}=1.
\end{gathered} \tag{7}$$

The temporal partition has now been outlined in detail. Our dynamical model for the artificial language is complete, except for the perturbations necessary to guide the trajectory along the partitions, in order to generate the corresponding symbol sequence. The next section takes care of this in the context of communication.

IV. COMMUNICATING WITH CHAOTIC DYNAMICS

As expected, the uncontrolled chaotic trajectory will not necessarily follow a symbol sequence that is consistent with the message. Also, our model does not include complex structures of words beyond pairs of units. So we introduce trajectory perturbations in order to make the due corrections for generating the desired message. Sometimes it is not convenient to apply trajectory perturbations, and as is usually done in chaos control, if the perturbation exceeds an upper bounded value, no perturbation is applied. Instead, to move the trajectory to some target location, the chaotic system is iterated a certain number of times until, due to ergodicity, the trajectory visits the target location. The upper bound δ_ϵ represents the maximum allowed value of the perturbation to be applied to the trajectory. The message can then be generated and transmitted from the transmitter to the receiver. However, in our approach, instead of the message, we send instructions that enable the receiver to generate the message. The instructions, or codeword, may contain information about the trajectory perturbations (numbers that $\in \mathbb{R}$) and/or about the number of iterations applied to the dynamical system (numbers that $\in Z$). Importantly, both receiver and transmitter know the dynamical equations, the parameter values, and the temporal partition. Another shared piece of information is the initial condition X_0 with which the codeword or the encoding is realized. The result is secure and it involves compressed information transmission.

In what follows we show how the transmitter encodes the message and how the receiver decodes it. For instance, if we want to transmit the two-character message $M=M_1M_2=52$, each component of the unit sequence M is identified by the subscript t of M_t.

To encode the message the *transmitter* enacts the following strategy.

(1) Set the counter t equal to 1 at the beginning of the encoding process, and incremented by one each time the process comes back to this step.

(2) Make j equal to the character to be transmitted, $j=M_t$ (for the first letter, $j=5$; for the second letter, $j=2$).

(3) Find the partition P_i where the point X_{t-1} is located.

(4) Compute the minimum distance ϵ_{t-1} between X_{t-1} and the subpartition $P_{i,j}$. If $X_{t-1}\in P_{i,j}$, then $\epsilon_{t-1}=0.0$.

(5) If $\epsilon_{t-1}\leqslant\delta_\epsilon$, send the integer number 0 and the real number ϵ_{t-1} to the receiver, and go to step 7.

(6) If $\epsilon_{t-1}>\delta_\epsilon$,

(a) iterate X_{t-1} under Eq. (1); set $X_{t-1}=T(X_{t-1})$ and update an integer variable N_{t-1} that registers the number of times step 6(a) is called;

(b) search for the subpartition $P_{i',j}$ ($P_{i'}$ might be different from P_i), such that the distance ϵ_{t-1} between X_{t-1} and $P_{i',j}$ is minimal; if $X_{t-1}\in P_{i',j}$ then $\epsilon_{t-1}=0.0$;

(c) if $\epsilon_{t-1}\leqslant\delta_\epsilon$, send N_{t-1} and ϵ_{t-1} to the receiver and go to step 7;

(d) if $\epsilon_{t-1}>\delta_\epsilon$, return to step 6(a).

(7) Set $X'_{t-1}=X_{t-1}$.

(8) Add the perturbation to X'_{t-1}: $X''_{t-1}=X'_{t-1}+\epsilon_{t-1}$.

(9) Iterate the point X''_{t-1} under T, $n_{i,j}$ times, obtaining $X_t=T^{n_{i,j}}(X''_{t-1})$.

(10) Go back to step 1.

Suppose that the codeword is "$N_0\ \ \epsilon_0\ \ N_1\ \ \epsilon_1$," where N_0 and N_1 are two integer numbers, and ϵ_0 and ϵ_1 are two real numbers. This codeword is composed of two 2D vectors, each vector with one component integer and another real.

To decode the message the *receiver* does the following.
(1) Set the counter t equal to 1 at the beginning of the decoding process, and incremented by 1 each time the process comes back to this step.
(2) Get the integer codeword element N_{t-1}.
(3) Iterate the point X_{t-1} under T, N_{t-1} times. Thus, $X'_{t-1}=T^{N_{t-1}}(X_{t-1})$.

TABLE I. Codeword for the message ''1525.'' $X_0 = 0.700030753590814$.

Codeword
$N_0 = 2$
$\epsilon_0 = 0.0$
$N_1 = 0$
$\epsilon_1 = 0.0$
$N_2 = 0$
$\epsilon_2 = 0.0$
$N_3 = 2$
$\epsilon_3 = 1.662759113164498 \times 10^{-3}$

(4) Get the real codeword element ϵ_{t-1}.
(5) Perform an orbit correction by $X''_{t-1} = X'_{t-1} + \epsilon_{t-1}$.
(6) Verify the subpartition $P_{i,j}$ that the point X''_{t-1} belongs to.
(7) The number j represents the first character, $M_t = j$.
(8) Iterate X''_{t-1}, under T, $n_{i,j}$ times, and obtain $X_t = T^{n_{i,j}}(X''_{t-1})$ (which belongs to P_j).
(9) Go back to step 1.

An implementation of this encoding/decoding scheme is presented in Tables I and II. Table I shows the codeword for the message ''1525,'' and Table II shows how the receiver decodes the message. The receiver gets the initial condition X_0, which is located in the partition P_5, and transfers it to the subpartition P_{31}, by doing $X''_0 = T^{N_0}(X_0) + \epsilon_0$. Then, the point X_1, which is the n_{31}th iteration of X''_0 under T, falls within the partition P_1, which decodes for the unit 1. Next, the point X_1 is iterated N_1 times under T, and then an orbit correction ϵ_1 is applied to it. This transfers the point to the subpartition P_{25}, which guides point X''_1 to point X_2, by doing $X_2 = T^{n_{25}}(X''_1)$, located in partition P_5, which decodes for the unit 5. The remainder of the message is obtained by repeating this process.

One important feature of our modeling/encoding/decoding technique is that few trajectory alterations are necessary to guide the trajectory through the desired symbol sequence of the temporal partitions. These dynamic alterations are produced either by applying a number N of iterations, different from 0, or by applying an orbit correction ϵ, also different from 0.0.

In Figs. 5 and 6, a codeword frequency analysis is shown for two values of δ_ϵ, $\delta_\epsilon = 0.0$ and $\delta_\epsilon = 0.1$. Notice that in both cases, most of the codeword is composed of either null integers or null floating points. When not null, the element of the codeword is in most cases very small, which strongly indicates that the proposed modeling is reliable.

In Tables I and II we show the components of the codeword that belong to $\mathbb{R}$, and the trajectory with double-precision. The locations of the subpartitions are also defined in double precision [see Eq. (6)]. This might lead the reader to the impression that the method is numerically dependent. This is not the case, as is demonstrated in the next section,

TABLE II. Decoding of the message ''1525.'' The value of the variables are $X_0 = 0.700030753590814$, $X''_0 = 0.537733840231034$, $X_1 = X'_1 = X''_1 = 8.85545324873836E\text{-}2$, $X_2 = X'_2 = X''_2 = 0.874472231429296$, $X_3 = 0.439082191553455$, $X'_3 = 5.849430262269988E\text{-}2$, $X''_3 = 6.015706173586438E\text{-}2$.

Trajectory	X_n belongs to	Decoded message
$X'_0 = T^{N_0}(X_0)$		
$X''_0 = X'_0 + \epsilon_0$	P_{31}	
$X_1 = T^{n_{31}}(X_0)$		''1''
$X'_1 = T^{N_1}(X_1)$		
$X''_1 = X_1 + \epsilon_1$	P_{15}	
$X_2 = T^{n_{15}}(X_1)$		''5''
$X'_2 = T^{N_2}(X_2)$		
$X''_2 = X_2 + \epsilon_2$	P_{52}	
$X_3 = T^{n_{52}}(X_2)$		''2''
$X'_3 = T^{N_3}(X_3)$		
$X''_3 = X_3 + \epsilon_3$	P_{15}	
$X_4 = T^{n_{15}}(X_3)$		''5''

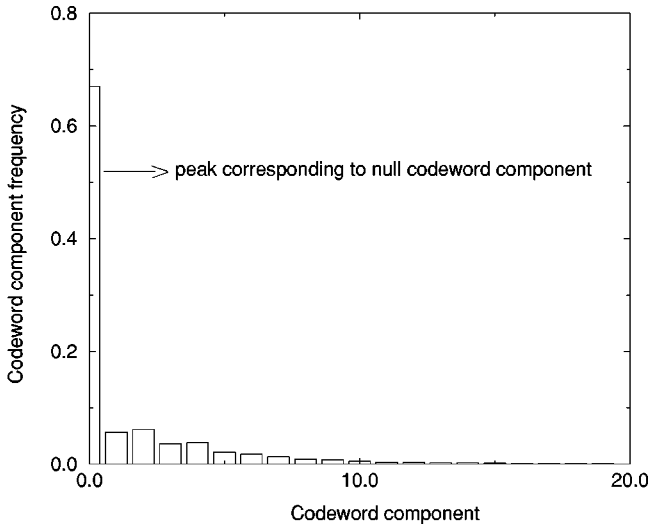

FIG. 5. Normalized number of appearances for each component of the codeword. $\delta_\epsilon = 0.0$. Notice that most of the codeword is composed of either null integers or null floating points.

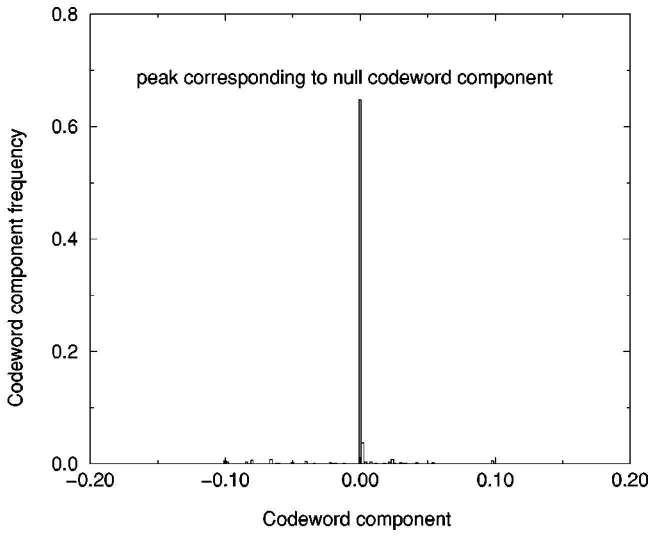

FIG. 6. Normalized number of appearances for each component of the codeword. $\delta_\epsilon = 0.1$. Notice that most of the codeword is composed of either null integers or null floating points.

where the components of the codeword that belong to $\mathbb{R}$ are encoded in a set of integer numbers. The codeword components represent dynamical alterations that the receiver must apply to the dynamical system in order to target the trajectory (and recover the message). Therefore, the receiver dynamical trajectory is independent of the codeword floating number precision. The subpartitioning is known by the receiver and to the transmitter, and does not depend on the precision of the transmitted trajectory. Finally, receiver and transmitter should be synchronized, which is possible even if receiver and transmitter possess slightly different architectures [6,15].

V. OPTIMIZING THE TRANSMISSION

The manner in which the codeword is transmitted depends partially on the value of δ_ϵ. We consider three cases: $\delta_\epsilon = 0.0$, $0.0 < \delta_\epsilon < 0.2680$, and $\delta_\epsilon \geqslant 0.2680$. For $\delta_\epsilon = 0.0$, no perturbation is applied. So, all ϵ are null and there is no need to transmit the real part of the codeword. The opposite happens for $\delta_\epsilon \geqslant 0.2680$. In this case, it is always possible to find a perturbation in order to transfer the orbit to the desired subpartition. As a result, all integers are null and there is no need to transmit the integer numbers. For $0.0 < \delta_\epsilon < 0.2680$, both the integer and the real elements of the codeword are transmitted. In addition, the transmission of each element takes no more than Δ arbitrary units of time. Therefore, independently of the value of δ_ϵ, an important consequence of the criterion used to construct the subpartitions is that, during the encoding of the message, it is often the case that moving from one point X_0 to the next X_1 does not require any perturbation or any extra iterations. When this happens, the communication channel is free and certainly can be used for transmission of other messages. So, time division multiplexing may be another simple and attractive possibility in this approach.

A schematic view of the three methods of transmitting the codeword ξ is as follows.

If $\delta_\epsilon = 0.0$, $\xi_i \in Z$

$$\frac{N=0 \quad \text{channel is free during } \Delta}{N\neq 0 \qquad\qquad \text{send } N}. \tag{8}$$

If $\delta_\epsilon \geqslant 0.2680$, $\xi_i \in \mathbb{R}$

$$\frac{\epsilon=0.0 \quad \text{channel is free during } \Delta}{\epsilon\neq 0.0 \qquad\qquad \text{send } \epsilon}. \tag{9}$$

If $0.0 < \delta_\epsilon < 0.2680$, $\{\xi_i^0, \xi_i^1\} = \{N_i, \epsilon_i\}$, with $N_i \in Z$ and $\epsilon_i \in \mathbb{R}$.

$N=0$	$\epsilon=0.0$	channel is free during 2Δ
	$\epsilon\neq 0$	channel is free during Δ and send ϵ
$N\neq 0$	$\epsilon=0.0$	send N and channel is free during Δ
	$\epsilon\neq 0.0$	send N and ϵ

(10)

The information being sent is composed only of the non-null codeword components. Even with this consideration, the

TABLE III. Code for ϵ_n

ϵ_n lead to	code	ϵ_n lead to	code
P_{12}	-1	P_{31}	-1
P_{13}	-2	P_{34}	-2
P_{15}	-3	P_{41}	-1
P_{21}	-1	P_{43}	-2
P_{22}	-2	P_{51}	-1
P_{25}	-3	P_{52}	-2

transmission can still be improved. Notice that due to the way the transmitter computes the perturbations ϵ_n [given by steps 4 and 6(b) of Sec. IV], no matter what the value of ϵ_n is, the number of different ϵ_n's that lead the trajectory to some particular subpartition is finite. Moreover, the transmission of one codeword element that belongs to $\mathbb{R}$ demands 8 bytes (with double-precision real number), making the codeword larger than the message itself. This difficulty can be easily circumvented by simply applying the encoding procedure again, this time to the real components of the codeword. This is done to some different set of numbers that we choose to be the set of negative integers Z^-, to differ from the set of positive integer codeword components.

This second encoding does not depend on the value of the codeword component. It depends on the way it was obtained. More specifically, it depends on the subpartition where the perturbation sends the trajectory in. Table III shows the chosen code for these perturbations. The code is chosen such that the codeword has a minimum number of different integer numbers. In the next section we show that this condition enhances security and transmission efficiency. In order to better evaluate the method of information transfer, we define the compression rate R as the ratio between the amount of bytes needed to transmit the codeword and the amount of bytes needed to transmit the message. The number of bits B needed to transfer one component of the codeword (message) depends on how many different components H the codeword (message) has. So,

$$B = \text{Int}[\log_2(H)] + 1. \tag{11}$$

If the message is composed of five units, then one transmitted unit is counted to contribute three bits [as given by Eq. (11)] to the size of the full transmitted message. Similarly, depending on the number of different components of the codeword, each component contributes with an amount given by Eq. (11). When the perturbation is not bounded, there are only three different numbers being transmitted: -1, -2, and -3. Therefore, 2 bits are required to transfer one codeword component. We want to emphasize that the null codeword component does not contribute to the size of the full codeword. The best compression rate is obtained for $\delta_\epsilon \geqslant 0.26800$, which is $R = 0.45827$, meaning that our technique successfully transmit information in a compressed format, in addition to the security features discussed in the next section. Figure 7 shows that $R < 1.0$ for most values of δ_ϵ. The best performance of this model happens when both compression rate and perturbation δ are minima. This is the case for $\delta_\epsilon \approx 0.0015$ and $R = 0.6872$ as seen in Fig. 7. There is no change in R for $\delta_\epsilon > 0.2679$ as expected, since $\delta_\epsilon = 0.2679$ is

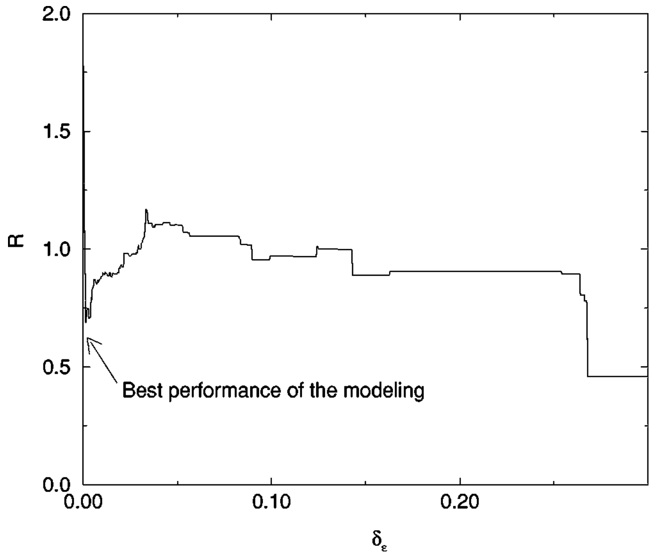

FIG. 7. Compression rate for the codeword with respect to the bound on the size of the perturbation δ_ϵ. $R<1.0$ for most values of δ_ϵ. There is no change in R for $\delta_\epsilon>0.2679$ as expected, since $\delta_\epsilon=0.2679$ is the largest perturbation needed to move a point to some subpartition.

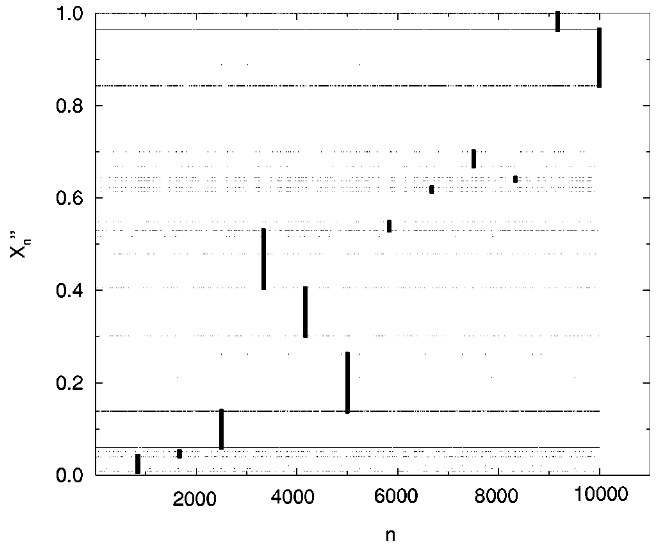

FIG. 8. Evolution of the variable X_n'' with no bound on δ_ϵ. The black vertical bar represents the subpartitions given by Eq. (6). The trajectory is dense in the edges of the subpartitions, identified by the thin vertical black lines. This occurs because ϵ_n must be minimal (see step 4, Sec. IV).

the largest perturbation needed to move a point to some subpartition. For this case it is easy to construct a codeword. Suppose for example that $X_0\in P_1$ and that the message is $M=$ ''5 2 1.'' To construct the codeword we look at Table III, and find perturbations that make the point X_n'' go to the following subpartitions: P_{15}, P_{52}, P_{21}. Thus, $\xi=$ ''-3 -2 -1.''

VI. LANGUAGE RECOGNITION AND SECURITY

The language approach for encoding-decoding technique presented here can be applied to different types of language. Each language, according to its own grammar, requires a different class of temporal partition function. In terms of language recognition, our method can be implemented in two distinct ways. One is by just sending to the receiver the temporal partition function as a header file. The other is by recording on a data base specific partitions for different languages, and then picking the particular partition that corresponds to the language used at the encoder. To pick the correct decoding temporal partition, the receiver has only to check if the decoded message is consistent with the dynamics of the language for which that particular temporal partition is generated. If there is consistency, the temporal partition under consideration has revealed the language that the transmitter has used for encoding. Figure 7 shows that there are intervals of δ_ϵ for which R does not change. These intervals are caused by the natural tendency of the modeling to look for a minimum set of trajectories that encodes the particular message when a large perturbation is allowed. In this case, the method often forces the trajectory to go to some of the subpartition edges, breaking down the random behavior of the chaotic trajectory. If we plot the evolution of the variable X_n'' during the encoding when $\delta_\epsilon=0.2679$, we see in Fig. 8 that the trajectory is dense in some specific regions. These regions happen to be the edges of the subpartitions, identified by the thin vertical black lines. This occurs because ϵ_n must be minimal (see step 4, Sec. IV), and when that happens one might say that the secrecy of the codeword is not achieved. But, in fact, this is not a problem. For example, if the message to be transmitted is $M=$ ''3 1 3 1,'' a suitable codeword is $\xi=$ ''-2 $-1-2$ -1'' (ξ_i codes for M_i). However, different messages such as $M=$ ''4 1 3 1'' and $M=$ ''2 1 3 1,'' generate the same codeword $\xi=$ ''-2 -1 -2 -1.'' So the codeword is noninvertible, which is a necessary condition for safe coding. The codeword is invertible only for the receiver that knows (a) the correct initial condition, (b) the temporal partition, (c) the dynamical system, and (d) the type of transmission determined by the value of δ_ϵ. These are the keys for decoding the message. Notice that when $\epsilon_n=0.0$ no information is transmitted during a time interval Δ. Assuming that the eavesdropper knows the encoding-decoding technique but has no access to the keys, the only plausible information then is that during the time interval Δ the trajectory needs no correction. This is not of much help for the eavesdropper. For instance, if $\xi=$ ''-2 0 -2 -1,'' the number of messages that can produce this codeword increases to seven: ''3 1 3 1,'' ''3 4 3 1,'' ''4 1 3 1,'' ''4 3 4 1,'' ''2 1 3 1,'' ''2 2 2 1,'' and ''2 5 2 1.'' Furthermore, secrecy is improved when the perturbation is bounded. In order to demonstrate the safety of the codeword, we make the very natural assumption that the eavesdropper knows the language grammar, in particular, the transition diagram of Fig. 1. We also assume that the dynamical process is known but not its parameter. Under these circumstances, a small error in either the initial condition or in the system parameter or in the temporal partition, precludes the decoding beyond a one-unit message. For example, suppose that the temporal subpartition locations and the system parameter are known, but not the initial condition. Let the codeword (with components belonging to Z^+) be $\xi=$ ''20 0.'' We know that due to the sensitive dependence on initial conditions, small errors am-

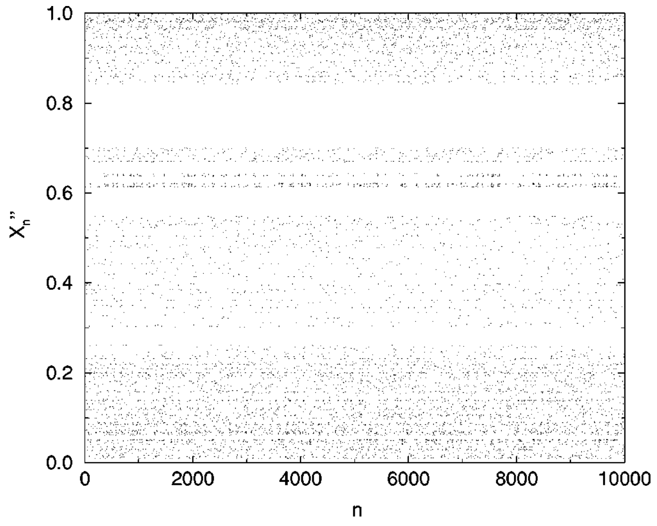

FIG. 9. Evolution of the variable X''_n, for $\delta_\epsilon = 0.005$. For small δ_ϵ, the trajectory is no longer limited to the edges of the subpartitions. That results in a codeword composed mostly of null components restoring the random behavior of the chaotic equation, thus improving the performance of the modeling procedure and security.

plified exponentially after a few iterations of the chaotic process. To decode the message one needs to iterate the initial condition 20 times under the process. Thus, for slightly different initial conditions and reasonable values of the largest Lyapunov exponent, ''20 0'' can in fact be the code word for any arbitrary one-unit message. For small δ_ϵ, the trajectory is no longer limited to being on the edges of the subpartitions most of the time as shown in Fig. 9, where the evolution of X''_n is plotted. That results in a codeword composed mostly of null components restoring the random behavior of the chaotic equation, thus improving the performance of the modeling procedure and security.

VII. ERROR CORRECTING CODE

Another relevant feature of our language-based communication system is its capability of recovering corrupted pieces of the codeword. Several causes can contribute to loss of information during the transmission, including noise and transmission interruptions, or dropouts. Here we model noise and gaps in the transmission as total absence of signal, or dropouts. As a matter of fact, dropouts can improve security in the system.

In Fig. 10, we show the partition transition diagram indicating how the chaotic trajectory visits different partitions during the process of generating the message. The negative numbers, as indicated by the arrows, are the codes of the perturbations that take the orbit through those partitions. If the message to be transmitted is $M=$''2 1 5,'' we need the trajectory to fall in the following partitions: P_2, P_1, and P_5. Suppose that the initial condition is located in P_5 (or any other partition that transfers orbits to P_2, namely either P_1 or P_2). With the help of Fig. 10, we construct a path oriented by the arrows connecting the partition P_5 (where the initial condition is) with the partitions P_2, P_1, and P_5. The path describes the trajectory and also indicates the codeword. In Fig. 10, it occurs if the perturbations $\xi=$''-2 -1 -3'' are applied. This set of perturbations is in fact the

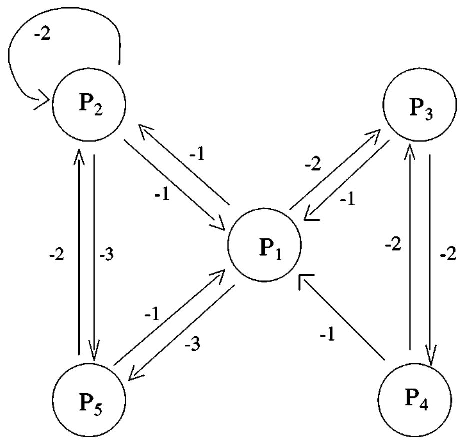

FIG. 10. Partition transition diagram, for perturbations encoded as shown in Table III, indicating how the chaotic trajectory visits different partitions during the process of generating the message.

codeword, and the length of the path is equal to the number of arrows guiding it.

We use n to represent the length of the dropout. It is the number of sequential components of the codeword that need to be checked to guarantee that the receiver establishes a robust decoding against dropouts. In the case of a codeword corrupted with dropouts of length 1, a number of conditions must be satisfied to successfully recover the message.

(1) The code for any ϵ_n ($\in \mathbb{R}$) that takes the trajectory to some subpartition must be different from the one that takes the trajectory to some other subpartition (so, Table III) cannot be used.

(2) When $\epsilon_n = 0.0$, the transmitter has to send to the receiver the same negative integer that encodes perturbations which take points to the subpartition containing X'_n.

(3) There must be no positive integer numbers in the codeword—a condition that is satisfied if there is no bound on the perturbation.

For large dropouts, complete reconstruction is possible if the codeword satisfies some extra conditions. Given that the codeword has a length n, the first codeword component, ξ_1 (that is not part of the dropout), encodes for a perturbation that puts the trajectory in P_q ($q=1,2,3,4,5$), and the last codeword component, ξ_n (that is not part of the dropout), encodes for a perturbation that puts the trajectory in P_w ($w=1,2,3,4,5$), and the dropout has size $n-2$. The following additional conditions must be satisfied for a successful reconstruction of a dropout of size greater than or equal to 2.

(4) The path with minimum length that connects the partition P_q with P_w must have a length equal to n.

(5) This path must be unique.

To understand why the first three conditions imply successful recovery of dropout of length one, let us assume initially that condition (1) is not satisfied, and that the code for the perturbations are given by Table III. Suppose that the message to be sent is $M=$''2 1 5'' (with initial condition in P_5), and the codeword $\xi_2=\{-1\}$, which is the code for $M_2=$''1,'' is not transmitted. In such a case, it is left to the receiver to analyze all possible messages that could have been transmitted. The receiver knows that $M_1=$''2,'' and

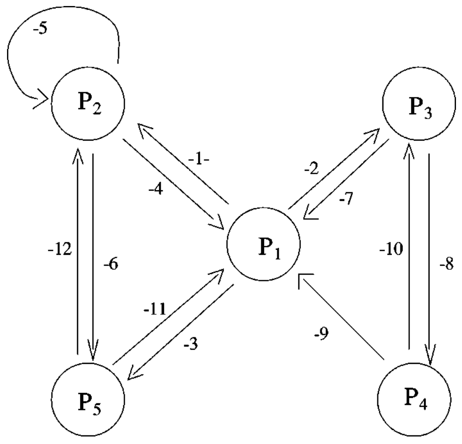

FIG. 11. Partition transition diagram, for perturbations encoded by respecting condition (6).

has to figure out what the next two letters of the message are. For that, it has to use what is already known, that is, $\xi_3 = \{-3\}$. It looks at the diagram of Fig. 10 and reconstructs all possible paths of trajectories that generates a valid message. So, the receiver comes up with two choices for the message: $M' =$ ''2 1 5'' and $M'' =$ ''2 2 5.'' These two choices are coded by $\xi' = \{-2\ \ -1\ \ -3\}$ and $\xi'' = \{-2\ \ -2\ \ -3\}$. Comparing the message known by the receiver $\xi_3 = \{-3\}$ with ξ_3' and ξ_3'', there is no way of discerning whether the message transmitted was M' or M''. This problem arises because the transitions from P_2 to P_5 and from P_1 to P_5 happen through perturbations that are coded equally. Therefore, we conclude that there must be only one code for each transition shown in Fig. 10. Thus, condition (1) must be satisfied. Hence, the perturbations are coded according to the negative numbers shown by the arrows in the transition diagram of Fig. 11. Looking at this diagram, and assuming that the initial condition belongs to P_5, the codeword for the message $M =$ ''2 1 5'' is $\xi = \{-12\ \ -4\ \ -3\}$. If $\xi_2 = \{-4\}$ is not sent to the receiver, looking at Fig. 11, it identifies that the only trajectory that generates a codeword with $\xi_3 = \{-3\}$ is the one who leaves P_2, and goes to P_5, passing through P_1.

The reason for condition (2) is to make a distinction between a dropout and a null ϵ_n. Otherwise the absence of signal would be ambiguous.

Condition (3) is required because if a bound in the perturbation is imposed, each codeword component is a subset of the set Z^2, composed of a pair of integer numbers, one set of positive integers, N^+, and another set of negative integers, N^-. When decoding the message, the receiver knows that N^- has only one value for each partition transition considered [condition (1)]. However, the same does not apply for N^+ which can be any number. Therefore, if N^+ is part of the codeword, the receiver faces the task of finding a number with endless choices. Thus, condition (3) results in a codeword composed of only numbers of the set N^-. If condition (3) is not satisfied, and the codeword is composed of both positive and negative integers, there is no way of correcting a transmission error. However, we still can identify errors and be certain about the inaccuracy of the transmitted codeword. This is done by checking if the decoded message respects the transition diagram in Fig. 1. When the decoded message is halted with respect to such transitions, there is an error in the transmission.

By considering conditions (4) and (5), we determine that 3 is the maximum length for the dropout which still allows message reconstruction, if the message is generated by the diagram of Fig. 1. When conditions (4) and (5) are not satisfied, the receiver is faced with more than one option for the possible message, and reconstruction becomes unfeasible. However, the receiver can make its choice based on the statistics of the language and pick the option with a sequence of letter frequency consistent with the transition frequency. This will considerably increase the chance of picking the right option.

When the code obeys all five conditions, dropout reconstruction offers no difficulty. But if the first three conditions are satisfied, there is no codeword secrecy. If Fig. 11 is used to encode the perturbations that belong to $\mathbb{R}$, we are coding the possible transitions of pairs of letters in the message. An eavesdropper can, after some work, find out which codeword component encodes for which pair of letters by simply making a frequency analysis in the codeword.

So, the method presented here has a tradeoff involving secrecy, dropout reconstruction, and compression rate. When secrecy and high compression rate are not fundamental, the method guarantees that dropouts of at least length one can be reconstructed. The compression rate is not the best permitted by the method, but still reaches a reasonable rate of $R = 4/3$. In the case of dropout reconstruction being vital (common when the transmission is carried out only once as in the case of spatial probes and satellites), and there is no need for codeword secrecy, the scheme proposed here works fine. Adding security requires compliance with condition (2) which brings back lack of dropout reconstruction capability. Outing of this loop is possible by redefining condition (2) as

(6) null perturbations that belong to $\mathbb{R}$ are coded into an integer that we consider to be the null integer.

Condition (6), instead of condition (2), makes the transitions of Fig. 11 identifiable by either the negative number shown by the arrows, or by the new code for the null perturbation. The presence of a null number in the codeword prevents the eavesdropper from discovering the meaning of each codeword component by doing a frequency analysis. In addition, it restores to the receiver, and only to the receiver, the capability of correcting errors due to dropouts during the transmission. However, depending on the value of the initial condition, a dropout might not be allowed reconstruction. More specifically, if the initial condition produces a codeword with a null component, and this component is part of the dropout, the receiver is not able to tell the dropout from the null element. This situation is taken care of by condition (1).

The numerical procedure dropout reconstruction is realized by making a tree of possible paths through the partition transition diagram, as shown in Fig. 12. The message to be sent is ''1 5 2,'' coded as ''$\epsilon_0\ \ \epsilon_1\ \ \epsilon_2$.'' These three real numbers are then coded again by ''$-9\ -3\ -12$'' [if condition (2) is respected] or ''$0\ -3\ -12$'' [if condition (6) is respected instead of (2)], (with $\epsilon_1 = 0.0$, $\epsilon_2 = 3.832339823599391$ E-2, $\epsilon_3 = 0.142804540832172$). If a dropout occurs and ϵ_2 is not transmitted, we find its value by

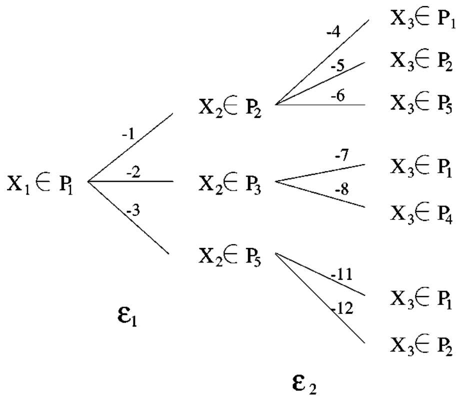

FIG. 12. Tree that allows the receiver to recover a gap in the codeword. The message to be sent is ''1 5 2,'' codes as ''ϵ_0 ϵ_1 ϵ_2.''

checking that there is only one path for which the value of ϵ_2 corresponds to the value transmitted.

VIII. CONCLUSIONS

We propose a method of communicating using deterministic dynamics based on modeling of language. Our procedure provides an efficient way of implementing a variety of different features required for a good communication system such as security, compression, language recognition, and error correcting code. We create a temporal partition of the phase space of a dynamical system. This allows the statistics of the message, constructed from an artificial language, to be approximately equal to the symbolic sequence statistics of a chaotic trajectory along this partition. This temporal partition is the cornerstone of the chaotic modeling of the artificial language. Equations other than Eq. (1) can be used to generate the chaotic trajectory, provided that Eqs. (2) and (5) are satisfied. Most of the time there is no need for external intervention for the message generation, which makes our temporal partition approach very attractive indeed.

We foresee the usage of our temporal partition as a way of modeling other discrete processes that can be represented by a symbol sequence. In more general terms, given any experimental symbol sequence, which could be originated by some unknown dynamical process, we can find an approximate model of the dynamics by wise partitioning of the phase space with some function T that generates the chaotic trajectory.

ACKNOWLEDGMENTS

Murilo S. Baptista is supported by Fundação do Estado de São Paulo (FAPESP). This work was also supported by a joint CNPq/NSF-INT grant. We would like to thank H. C. M. Andrade and R. Ferreira for helpful discussions.

[1] S. Hayes, C. Grebogi, and E. Ott, Phys. Rev. Lett. **70**, 3039 (1993).
[2] Y.-C. Lai and C. Grebogi, Proc. SPIE **2038**, 91 (1993).
[3] S. Hayes and C. Grebogi, Proc. SPIE **2038**, 153 (1993).
[4] S. Hayes, C. Grebogi, E. Ott, and A. Mark, Phys. Rev. Lett. **73**, 1781 (1994).
[5] L. M. Pecora and T. L. Carroll, Phys. Rev. Lett. **64**, 821 (1990); K. Cuomo and A. V. Oppenheim, *ibid.* **71**, 65 (1993); L. Kocarev and U. Parlitz, *ibid.* **74**, 5028 (1995); M. Hasler, Int. J. Bifurcation Chaos Appl. Sci. Eng. **8**, 647 (1998).
[6] Y. C. Lai and C. Grebogi, Phys. Rev. E **47**, 2357 (1993).
[7] M. G. Rosenblum, A. S. Pikovsky, and J. Kurths, Phys. Rev. Lett. **76**, 1804 (1996); E. Rosa, Jr., E. Ott, and M. H. Hess, *ibid.* **80**, 1642 (1998).
[8] E. Bollt, Y.-C. Lai, and C. Grebogi, Phys. Rev. Lett. **79**, 3787 (1997).
[9] E. Bollt and Y.-C. Lai, Phys. Rev. E **58**, 1724 (1998).
[10] M. Dolnik and E. M. Bollt, Chaos **8**, 702 (1998).
[11] D. Gligoriski, D. Dimovski, L. Kocarev, V. Urumov, and L. O. Chua, Int. J. Bifurcation Chaos Appl. Sci. Eng. **6**, 2119 (1996).
[12] E. Rosa, Jr., S. Hayes, and C. Grebogi, Phys. Rev. Lett. **78**, 1247 (1997).
[13] K. M. Short, Int. J. Bifurcation Chaos Appl. Sci. Eng. **7**, 1579 (1997).
[14] M. S. Baptista, Phys. Lett. A **240**, 50 (1998).
[15] J. Schweizer and M. P. Kennedy, Phys. Rev. E **52**, 4865 (1995).
[16] C. Grebogi and A. N. Kaufman, Phys. Rev. A **24**, 2829 (1981).
[17] A. B. Rechester and R. B. White, Phys. Lett. A **156**, 419 (1991); A. B. Rechester and R. B. White, *ibid.* **158**, 51 (1991).
[18] X. Z. Tang, E. R. Tracy, A. D. Boozer, A. deBrauw, and R. Brown, Phys. Lett. A **190**, 393 (1994).
[19] X. Z. Tang and E. R. Tracy, Chaos **8**, 688 (1998).
[20] R. Badii and A. Politi, *Complexity: Hierarchical Structures and Scaling in Physics* (Cambridge University Press, New York, 1997); B.-L. Hao and W.-M. Zheng, *Applied Symbolic Dynamics and Chaos* (World Scientific, Singapore, 1998).
[21] M. Lehrman, A. B. Rechester, and R. B. White, Phys. Rev. Lett. **78**, 54 (1997).
[22] J. P. Crutchfield and K. Young, Phys. Rev. Lett. **63**, 105 (1989); J. Kurths, A. Voss, A. Witt, P. Saparin, H. J. Kleiner, and N. Wessel, Chaos **5**, 88 (1995).
[23] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication* (The University of Illinois Press, Chicago, 1964).