Chaos computing in terms of periodic orbits

Behnam Kia,[1,2] Mark L. Spano,[1] and William L. Ditto[1]
[1]*School of Biological and Health Systems Engineering, Arizona State University, Tempe, Arizona 85287-9709, USA*
[2]*School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, Arizona 85287-5706, USA*
(Received 12 October 2010; revised manuscript received 19 January 2011; published 12 September 2011)

The complex dynamics of chaotic systems can perform computations. The parameters and/or the initial conditions of a dynamical system are the data inputs and the resulting system state is the output of the computation. By controlling how inputs are mapped to outputs, a specific function can be performed. Previously no clear connection has been drawn between the structure of the dynamics and the computation. In this paper we demonstrate how chaos computation can be explained, modeled, and even predicted in terms of the dynamics of the underlying chaotic system, specifically the periodic orbit structure of the system. Knowing the dynamical equations of the system, we compute the system's periodic orbits as well as its stability in terms of its eigenvalues, thereby demonstrating how, how well, and what the chaotic system can compute.

DOI: 10.1103/PhysRevE.84.036207 PACS number(s): 05.45.Gg, 05.10.−a, 02.70.Wz

I. INTRODUCTION

Chaos computing, a candidate for replacing conventional computing technology, has been around for more than one decade [1–20]. So far, research in the area has focused on the observation and manipulation of computation within the rich chaotic dynamics of highly nonlinear systems. In essence, data inputs and control inputs are encoded as either the initial conditions of a given chaotic system, the parameters of the chaotic system, or the parameters of a chaos controller or chaos synchronizer. The final state, or the stabilized state of the chaotic system in the case of control or synchronization, is the computation's output. By observing how the chaotic system maps inputs to outputs, an instruction set is generated that the chaotic system can perform. Each function of the instruction set is chosen by using appropriate control inputs [3–20].

Until now, to the best of our knowledge, no *direct* technique has been introduced to determine the possible functions that a given chaotic system can implement or the control inputs that select these instructions. Rather, the evolution of chaotic computing model under different inputs is observed and monitored to determine its instruction set.

In this paper the relationship between the computational capabilities and properties of a chaos-based computer and the dynamical properties of the underlying chaotic system is explained. Specifically we demonstrate the instructions that a chaotic system can implement. We also examine the stability of those instructions against noise; this stability is *directly determined* from the periodic orbit structure and the dynamics of the system.

Periodic orbit theory is an efficient approach to study a chaotic dynamical system in terms of the fundamental orbits of its attractor [21]. In this paper we explain, model, and study chaos computing in terms of these basic periodic orbits.

Periodic orbits were introduced into the theory of dynamical systems by Poincaré [22], and they have played a main role in the mathematical work on dynamical systems ever since [21–24]. Periodic orbits provide a detailed, invariant characterization for deterministic low-dimensional dynamical systems [21,24]. As a result, explaining chaos computing in terms of these periodic orbits has profound theoretical consequences. On the other hand, low-period periodic orbits are experimentally extractable from time series [25]. This contributes practical importance to our ability to explain chaos computing in terms of basic periodic orbits; e.g., it enables us to predict and determine the instruction set that a chaotic system can implement and the stability of those instructions against noise just by having access to a time series from the chaotic system. This latter will be explicated in a future paper.

The organization of the paper is as follows: in Sec. II, we present a very brief review of chaos computing; in Sec. III, we demonstrate how to derive the computational functionality; in Sec. IV, we show how to estimate the computational robustness of the chaotic system from its dynamical equations; in Sec. V, two examples are presented; and in Sec. VI, we present our conclusions.

II. CHAOS COMPUTING

The main idea of chaos computing is to harness the library of orbits and patterns inherent in chaotic systems to select out logic operations and to utilize the sensitivity to initial conditions of such systems to perform rapid switching (morphing) between all of these logic functions [3–7]. These features are sufficient to perform reconfigurable logic operations using the chaotic system.

Data and control inputs to a chaotic system (either continuous or discrete) may be encoded as either the initial conditions of the chaotic system or the parameters of the system. Here we focus on the former technique. After applying the inputs, the system is allowed to evolve for a predefined time, after which time this "final state" of the chaotic system is decoded as the computation's output.

To be more precise, consider the m digital data inputs $X^1_{\text{Data}}, X^2_{\text{Data}}, \ldots, X^m_{\text{Data}}$ to a computing engine and the n digital control inputs $X^1_{\text{Control}}, X^2_{\text{Control}}, \ldots, X^n_{\text{Control}}$. Computation with this system consists of three steps:

Step 1. Each set of data and control inputs is mapped to a point on the unstable manifold of the chaotic system. This point will be used as the initial condition for the chaotic system. Let T map (encode) the m data and n control inputs onto the space of the initial conditions. If L is a binary set $\{0,1\}$, then $L^{(n+m)}$ represents the domain of T, which consists of all the possible combinations of digital data and control inputs. We

let β be the unstable manifold of the chaotic system, R^s the general state space of the chaotic system, and Y the output of the encoding map on the unstable manifold. In this case the general form of the encoding map T is as follows:

$$\begin{aligned} &T : L^{(n+m)} \to \beta, \quad \beta \subset R^s, \quad L = \{0,1\}, \\ &Y = T\left(X^1_{\text{Data}}, X^2_{\text{Data}}, \ldots, X^m_{\text{Data}}, X^1_{\text{Control}}, \ldots, X^n_{\text{Control}}\right). \end{aligned} \tag{1}$$

Step 2. Starting from the initial conditions produced by the encoding map, the chaotic system evolves for a fixed time (or for a fixed iteration number, if the chaotic system is discrete).

Step 3. After the evolution time, the system stops working and its state at the end of the evolution time is sampled and decoded to the outputs using a decoding map.

The encoding map maps different sets of the inputs to different points on unstable manifold of the chaotic system and these points are used as initial conditions for the chaotic system. Since the system is on the unstable manifold, the orbits of the chaotic system are very sensitive to the inputs and the orbits dramatically change with just a one-bit change in the control input. Thus control inputs can select a chaotic logic function. To evaluate which digital function is selected with a particular control input, one notes the association of this control input with the logic function and then enumerates all possible combinations of data inputs to construct the truth table of the function.

By changing the control input and repeating this procedure (of constructing the truth table of the digital function), one may observe a second digital function different (with high probability) from the first one. This is the meaning of the *reconfigurability* of chaos computing. By using all possible control inputs and finding the type of function that the chaotic system implements, we obtain the full instruction set of the chaotic system [4–7]. Different implementations have been introduced for chaos computing [10–14]. Furthermore, a company named ChaoLogix$^{\text{TM}}$ has been founded to commercialize chaos-based logic gates [15]. Also recently another method for computation based on nonlinearity of a dynamical system is introduced, which is named logical stochastic resonance (LSR) [26–31].

In what follows we address the important remaining questions: Why do we observe a specific form of logic function from a chaotic dynamic system? What are all the possible logic functions that we can obtain from any given chaotic system? How can we connect computation to the dynamics of chaos computing? In the next part these questions are addressed by connecting the chaos computing model to the dynamics of the chaotic system.

III. DYNAMICS AND COMPUTATION

Let **x** be the dynamic state of a chaotic system and let the chaotic discrete evolution of the system be governed by the dynamical equation

$$x_{p+1} = f(x_p). \tag{2}$$

The aim is to compute directly the spectrum of functions that a given chaotic system can implement and the robustness of these functions against noise from the dynamical equation (2).

The description of a low-dimensional chaotic system in terms of unstable periodic orbits, which is known as *periodic orbit theory*, is a powerful tool for the analysis of chaotic systems [21,23–25]. A chaotic system is composed of an infinite number of unstable periodic orbits (UPOs) [32]. It is known that a collection of short-period UPOs is enough to obtain a very precise approximation of a sufficiently low-dimensional chaotic system [21,23,24]. Here we approximate our chaotic system with an appropriate collection of short period orbits to estimate the computational functionality and robustness of the chaotic system. As was mentioned in Sec. II, during step 2 of computation, the chaotic system undergoes a specific number of iterations, which we denote as p. We claim that, in one-dimensional (1D) unimodal chaotic maps where the critical point x_c is mapped to unity and $f(0) = f(1) = 0$, for the p iterations that the chaotic system undergoes, approximating the chaotic system by all of its UPOs of length $p + 1$ is enough to determine the function set of the chaotic system and to approximate the robustness of these functions against noise. This method works for any other chaotic system where all symbolic sequences are admissible and therefore the topological entropy is ln(2). But in other chaotic systems we might need to use slightly higher length UPOs to model the system. This case will be studied in the Gaussian map example.

In a unimodal map, where the height of the map is unity and where $f(0) = f(1) = 0$, there are 2^p different unstable periodic points of order p, including repetition of periodic points of lower order [33]. For example, there are 2^4 unstable periodic points of period 4, which includes two unstable fixed points and two unstable periodic points (one unstable periodic orbit) of period 2. Thus in a unimodal map of height unity, there are exactly 2^4 possible symbolic sequences of length 4, and for each symbolic sequence there is a neighborhood of initial conditions where all the initial conditions have the same four-symbol iterates. Therefore, there is a one-to-one relationship between UPOs and the neighborhood of similarly behaved initial conditions. The same argument is correct for any other chaotic system that has no forbidden symbolic sequence or, equivalently, whose topological entropy is ln(2) [33]. Figure 1(a) shows how all periodic orbits of length 2 produce a polygonal approximation of the unimodal map. The unimodal map has two period-1 unstable fixed points, which are at the intersection of the map and the identity line, and two period-2 unstable fixed points. Two repetitions of the period-1 unstable fixed points are considered as periodic orbits of length two as well.

Because the behavior of the dynamical system in the neighborhood of each of these points may be approximated linearly, the unstable fixed point and nearby points lying on a straight line are a good approximation of the dynamics near that unstable fixed point. If we have sufficient numbers of these linear approximations, we can approximate the map in its entirety. Therefore, each fixed point neighborhood is one of the four faces of the polygonal (piecewise) approximation for the map, as illustrated by red (dark gray) tangent lines for period-1 fixed points and yellow (light gray) tangent lines for period-2 fixed points in Fig. 1(a).

As explained above, each face of the approximation is composed of an unstable fixed point or periodic point, plus

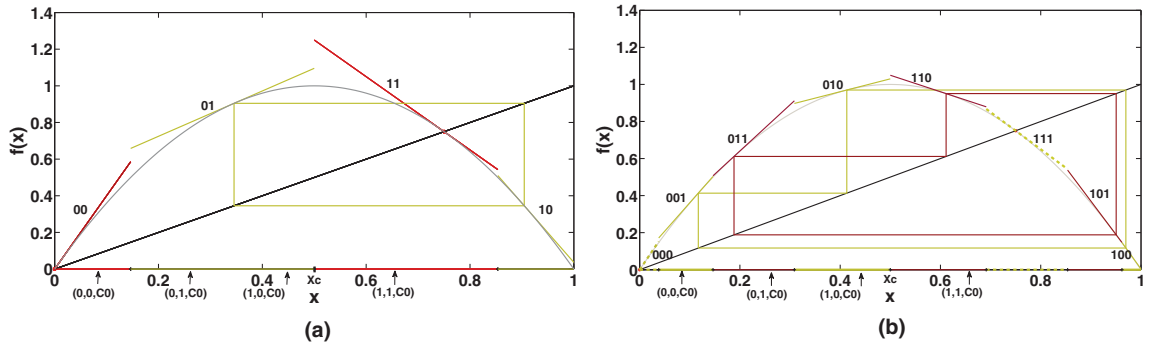

FIG. 1. (Color online) The figure on the left-hand side (right-hand side) shows how UPOs of length 2 (3) and the neighborhoods around them can be used to determine the functionality of the chaotic system when it undergoes one (two) iteration(s). Consider the left-hand graph and recall that we have chosen $X_i = 0$ if $f^{(i)}(x_0) \leqslant x_c$ and $X_i = 1$ if $x_c < f^{(i)}(x_0)$. Any set of initial conditions in the area denoted "00" will return to that area, generating the symbolic itinerary 00. An initial condition in the region denoted "01" will start below x_c, but the next iteration will take it above x_c, thus generating the symbolic itinerary 01. This is similar for the other regions. Note that some UPO neighborhoods are not used to implement the function of interest here.

all close-by points. These points are those whose Jacobian is qualitatively similar. Notice that projecting each face of the polygonal approximation on the x axis results in a neighborhood around each periodic point, where all the initial conditions within this neighborhood symbolically (in the *symbolic dynamics* sense) behave the same as the periodic orbit. In unimodal maps the critical point x_c can be used for partitioning of state space and assigning symbolic itineraries to initial conditions. As an example, the *symbolic itinerary* for $x = x_0$ is $X_0X_1X_2\cdots X_k$, where each succeeding digit in the itinerary denotes the next iteration of the map. We (arbitrarily) choose $X_i = 0$ if $f^{(i)}(x_0) \leqslant x_c$ and $X_i = 1$ if $x_c < f^{(i)}(x_0)$. Therefore, each periodic point locally explains the symbolic behavior of the initial conditions around itself during a specific number of iterations of the map. More specifically, UPOs of length $p+1$ represent the symbolic behavior of nearby orbits during the first p iterations of the chaotic map. Furthermore, the measure of the robustness against noise of each periodic orbit in terms of eigenvalues is a good approximation for the robustness of orbits around it until the pth iteration of the chaotic map. In Fig. 1(a) on the x axis, the neighborhoods of initial conditions that symbolically behave the same as the UPOs are denoted in the same colors. As an example, the first neighborhood on the x axis, which is illustrated by the red color, contains the initial conditions that symbolically behave the same as the UPO at 0 and all of which produce the symbolic itinerary 00 when they evolve under the chaotic map, i.e., for any initial condition in this neighborhood, $x_0, x_0 < x_c$ and $f(x_0) < x_c$. In Fig. 1(b) a different polygonal approximation for the chaotic map using period-3 UPOs is illustrated. This approximation is composed of two unstable fixed points and two new UPOs of period-3, resulting in an eight-faced polygonal approximation. Two faces of the polygonal approximation are delineated by the two unstable fixed points of the chaotic system, and the remaining six faces are related to two unstable periodic orbits of period-3, each unstable fixed point of the periodic orbit centering a face. In these two approximations, the boundaries between neighborhoods on the x axis are the preimages of the critical point x_c.

As described above, chaos computation encodes the data as well as the control inputs to form the initial conditions, next evolving the chaotic system from these initial conditions for some number of iterations, and lastly decoding the final state to obtain the output of the computation. The technique for obtaining the instruction set of a chaotic system for use in computation is as follows: When the chaotic system is iterated p times, approximate the chaotic system by its UPOs of length $p+1$. Determine in which UPO neighborhood the encoding map places each initial condition and the characteristic itinerary [e.g., (0,1) in Fig. 1(a)] for that neighborhood. The last symbol of this itinerary represents the output of the computation for those specific data and control inputs. By applying this technique for all combinations of data and control inputs, the instruction set of the chaotic system can be directly obtained.

Consider as an example a one-humped map, as shown in Fig. 1(a). This might be the logistic map or any other similar map. Now let us consider that this chaotic system undergoes one iteration and that, as explained above, we will be using period-2 UPOs. If we denote the selected control inputs from Eq. (1), $X^1_{\text{Control}}, X^2_{\text{Control}}, \ldots, X^n_{\text{Control}}$, collectively as C_0, all four possible initial conditions produced by the encoding map are illustrated on the x axis. Here, the aim is to implement a two-input function; therefore, we have four different combinations of initial conditions. In this example, the encoding map encodes the data inputs (0,0) to the point $(0,0,C_0)$, (0,1) to $(0,1,C_0)$, (1,0) to $(1,0,C_0)$, and (1,1) to $(1,1,C_0)$. The initial condition $(0,0,C_0)$ falls in the first neighborhood on the x axis, which produces 0 after one iteration. Therefore, the output of the computation for the (0,0) input is 0. The second and third initial conditions $(0,1,C_0)$ and $(1,0,C_0)$ fall in the second neighborhood, which is represented by the periodic orbit 01, and so the output of the computation for these two inputs is 1. The last initial condition $(1,1,C_0)$ settles in the third neighborhood, which corresponds to the 11 periodic orbit. Therefore, the output of the computation will be 1. We observe that the set of control inputs C_0 thus constructs an OR gate. This procedure can be repeated for other control inputs to obtain the instruction set of any given chaotic system.

The instruction set of the chaotic system for other iteration numbers can be obtained in a similar way. As a further example, we show this for iteration number 2 in Fig. 1(b). From the figure it is clear that the set of control inputs C_0 constructs a function that produces the output 1 when the inputs are (0,0) and (0,1), but produces the output 0 when inputs are (1,0), and (1,1).

IV. ROBUSTNESS AGAINST NOISE

UPOs can also help us in approximating the robustness against noise of the chaotic system while doing computation. The robustness of each UPO against noise can be measured by evaluating its Jacobian matrix. In our 1D case, the measure of the robustness of each UPO is simply the product of the slopes of all the tangent lines at each UPO. For example, for the dynamical system $x_{n+1} = f(x_n)$, the robustness against noise for a UPO of length $p+1$, $x_0^{\mathrm{UPO}}, x_1^{\mathrm{UPO}}, \ldots, x_p^{\mathrm{UPO}}, x_{p+1}^{\mathrm{UPO}} = x_0^{\mathrm{UPO}}$, is $\lambda_0 \times \lambda_1 \times \cdots \times \lambda_p$, where $\lambda_i = f'(x_i^{\mathrm{UPO}})$.

This robustness measure for each UPO can be used as an approximation for the robustness of orbits that start in the neighborhood of the UPO. To construct a specific function, the chaotic system maps the initial conditions produced by the encoding map to the final states. Therefore, to evaluate the robustness of each function in doing computation, the robustness for each orbit needs to be obtained, and the overall robustness of the function is the robustness measure of the least robust orbit, i.e., the worst case.

We assume the noise to the system is additive, $x_{n+1} = f(x_n) + D\varepsilon(t)$, where D is the intensity of the noise and $\varepsilon(t)$ is the white noise. We also assume the noise is approximately Gaussian white noise with zero mean and unit variance $\varepsilon(t) = N(0,1)$. Earlier we claimed that, when the chaotic system iterates p times, approximating the chaotic system by its UPOs of length $p+1$ is sufficient to determine the robustness against noise of the functions implemented by the chaotic system. To demonstrate this, let the chaotic system f iterate p times from a given initial condition x_0, producing the noisy orbit

$$\begin{aligned} &x_0 + \varepsilon(0), f(x_0 + \varepsilon(0)) + \varepsilon(1), \\ &\quad f(f(x_0 + \varepsilon(0)) + \varepsilon(1)) + \varepsilon(2), \ldots, \\ &\quad f(\cdots(f(f(x_0 + \varepsilon(0)) + \varepsilon(1))\cdots) + \varepsilon(p). \end{aligned} \tag{3}$$

By use of the polygonal approximation by UPOs of length $p+1$, the orbit can be approximated by:

$$\begin{aligned} &x_0 + \varepsilon(0), f(x_0) + D\lambda_1\varepsilon(0) + \varepsilon(1), \\ &\quad f^2(x_0) + D\lambda_1\lambda_2\varepsilon(0) + D\lambda_2\varepsilon(1) + \varepsilon(2), \ldots, \\ &\quad f^p(x_0) + D\lambda_1\lambda_2\ldots\lambda_p\varepsilon(0) \\ &\quad + D\lambda_2\ldots\lambda_p\varepsilon(1) + \cdots + \varepsilon(p), \end{aligned} \tag{4}$$

where $\lambda_i = f'(x_i^{\mathrm{UPO}})$ and x_i^{UPO} is an iterate of the UPO into whose neighborhood $f^{(i)}(x_0)$ places the iterate of the initial condition x_0.

Since $\varepsilon(t)$ is a normal Gaussian random variable, $\varepsilon(t) = N(0,1)$, the deviation of the final state in the noisy case from the original final state will be a Gaussian random process:

$$\begin{aligned} &D\lambda_1\lambda_2\cdots\lambda_p\varepsilon(0) + D\lambda_2\cdots\lambda_p\varepsilon(1) + \cdots + \varepsilon(p) \\ &\quad = N\big(0, D^2\lambda_1^2\lambda_2^2\lambda_3^2\cdots\lambda_n^2 + D^2\lambda_2^2\lambda_3^2\cdots\lambda_n^2 \\ &\qquad + \cdots + D^2\lambda_n^2 + D^2\big). \end{aligned} \tag{5}$$

Let y be the minimum distance of the noiseless final state $f^{(p)}(x_0)$ from the boundaries of the neighborhood in which it resides. If the deviation introduced by the noise exceeds this value, the orbit will enter another neighborhood, and it may result in an incorrect (undesired) output symbol. Therefore, the output symbol is robust to noise only if the noise cannot move the final state out of the neighborhood where it settles. If z is a Gaussian random variable, $z = N(0,\sigma)$, the probabilities that z is less than σ, 2σ, and 3σ are 84.2%, 97.8%, and 99.9%, respectively. We observe that the probability of $3\sigma < z$ is just 0.1%. This fact suggests that, if $3\sigma < y$, where σ is the standard deviation of ε, then the outcome will be robust against this noise 99.9% of the time. Since y is the minimum distance of final state $f^{(p)}(x_0)$ from the boundary of the neighborhood, it can be easily computed. Therefore, the noise intensity should be limited by

$$D < D_{\max} \equiv \frac{y}{3\times\sqrt{\lambda_1^2\lambda_2^2\lambda_3^2\cdots\lambda_n^2 + \lambda_2^2\lambda_3^2\cdots\lambda_n^2 + \cdots + \lambda_n^2 + 1}}. \tag{6}$$

Therefore, the symbol of the final state is robust against noise when the signal-to-noise ratio (SNR) is greater than $20\log\frac{A_{\mathrm{rms}}}{D_{\max}}$, where A_{rms} is the root mean square of $f(x)$ over all x. Notice that because of the ergodicity of the chaotic map f, A_{rms} *does not depend* on the selection of the initial condition.

Here we derived a measure of the robustness of an orbit against noise. To compute a robustness measure for a function, we apply the procedure to all orbits produced by the encoding map and set the lowest allowed SNR (highest D), as determined over all the orbits.

V. EXAMPLES

A. Logistic map

As an example, consider the functionality of the logistic map for doing computation and approximate the robustness of the resulting functions against noise. In this example we assume that an additive noise perturbs the dynamics as follows:

$$x_{n+1} = 4x_n(1 - x_n) + D\varepsilon(t). \tag{7}$$

A simple digital-to-analog converter with ten binary digital inputs will be used as the encoding map. Two inputs are allocated for data, which enables us to construct two-input functions, and the eight remaining inputs are used as controls to reconfigure the chaotic system by morphing between different functions. As the first step of the three-step computing algorithm, the two binary data inputs and eight binary control inputs are each encoded to either 0 or 1, yielding a combined initial value in [0,1), as follows:

$$x_0 = (0.I_1I_2C_1C_2\cdots C_8)_{\text{base 2}}, \tag{8}$$

where I_1, I_2 are the two binary data inputs, and $C_1, C_2, \ldots, C_8$ are the eight control inputs.

TABLE I. Instruction set of the logistic map for different iteration numbers.

p	Instruction set		
1	{(6,129,39.39 dB,39.9 dB),	(7,255,26.74 dB,26.11 dB),	(14,0,26.67 dB,26.31 dB)}
2	{(5,255,32.95 dB,34.2 dB), (11,52,42.72 dB,48.79 dB),	(9,123,30.73 dB,31.8 dB), (13,207,44.89 dB,46.3998 dB)}	(10,0,35.48 dB,33.7 dB),
3	{(2,89,50.46 dB,57.09 dB), (5,255,45.29 dB,46.89 dB), (11,20,46.80 dB,47.79 dB),	(3,53,38.59 dB,38.79 dB), (6133,36.32 dB,42.2 dB), (12,211,37.62 dB,42 dB) ,	(4,165,52.28 dB,52.59 dB), (10,0,47.52 dB,45.79 dB), (13,233,45.92 dB,51.89 dB)}
4	{(1,228,50.83 dB,50.39 dB), (6,126,44.13 dB,46.79 dB), (9,20,48.56 dB,59.89 dB), (12,59,50.89 dB,62.2 dB),	(3,200,52.73 dB,58.89 dB), (7,114,49.45 dB,59.49 dB), (10,17,40.40 dB,42.59 dB), (13,62,49.98 dB,61.89 dB),	(5,93,40.41 dB,45.19 dB), (8,24,49.45 dB,59.89 dB), (11,8,47.01 dB,51.29 dB), (14,144,53.41 dB,54.89 dB)}
5	{(1,177,54.97 dB,59.69 dB), (4,43,50.83 dB,53.09 dB), (7,29,62.30 dB,67.79 dB), (11,195,47.40 dB,48.69 dB), (14,228,63.97 dB,70.39 dB),	(2,98,50.07 dB,57.39 dB), (5,170,57.96 dB,63.39 dB), (8,80,54.69 dB,57.79 dB), (12,146,48.20 dB,47.79 dB), (15,128,55.34 dB,58.19 dB)}	(3,106,47.72 dB,51.89 dB), (6,35,54.85 dB,60.99 dB), (10,85,59.64 dB,64.49 dB), (13,58,47.30 dB,48.19 dB),
6	{(0,110,60.66 dB,62.69 dB), (3,140,58.56 dB,69.69 dB), (6,35,66.87 dB,69.19 dB), (9,53,53.76 dB,61.29 dB), (12,118,60.55 dB,65.19 dB), (15,126,54.21 dB,62.39 dB)}	(1,106,58.20 dB,58.99 dB), (4,66,59.67 dB,71.59 dB), (7,173,55.50 dB,57.89 dB), (10,92,57.01 dB,63.09 dB), (13,210,59.97 dB,60.49 dB),	(2,232,61.11 dB,62.59 dB), (5,68,55.76 dB,62.19 dB), (8,56,56.42 dB,63.49 dB), (11,46,55.98 dB,60.39 dB), (14,123,55.1 dB,64.69 dB),

At the second step of the algorithm, we allow the logistic map to undergo different numbers of iterations in order to determine the instruction set for each of those different numbers of iterations.

As the last stage of the computing model, the final state of the system is decoded to obtain the output of the computation as follows:

$$\text{output} = \begin{cases} 0 & \text{if} \quad x \leqslant 0.5, \\ 1 & \text{if} \quad x > 0.5, \end{cases} \tag{9}$$

where x has obviously been converted to base 10. We have computed period 2, 3, 4, 5, 6, and 7 UPOs for the logistic map. Then we have found the aforementioned neighborhoods around these UPOs and have computed the robustness of these UPOs against noise. Then for each iteration, e.g., $p-1$, we approximate and model the chaotic logistic map with period p UPOs. By use of this model we directly compute the instruction set of the chaotic logistic map when it undergoes p iterations. The results are listed in Table I for different values of p, $1 \leqslant p \leqslant 6$.

In Table I each instruction set consists of 4-tuples, the first element being the type of function that the logistic map constructs. The format that we use for identifying each of these functions is as follows: Table II presents the truth table of a sample function. We denote this function by a function number defined as $2^3O_3 + 2^2O_2 + 2^1O_1 + 2^0O_0$. Based on this definition, a chaotic system would present a two-input AND gate (with outputs 1000) as function number 8 and a two-input OR gate (with outputs 1110) as function number 14.

TABLE II. Truth table of a typical two input, one output function.

Data inputs	Output
00	O_0
01	O_1
10	O_2
11	O_3

The second element of the 4-tuple is the control inputs that construct this sample function. There are eight binary digital control inputs to the system, so the control inputs are numbered from 0 to 255. [Note that this is equivalent to expressing the control inputs as in Eq. (8), setting the data inputs to 0, and then multiplying the resulting number by 210 or 1024.] To evaluate the accuracy of our method in obtaining the functionality obtainable from a chaotic map, we have applied the computed control inputs to the logistic map, and in practice we have observed computationally that they construct the same functions that were predicted based on the periodic orbit approximation.

The third element of each 4-tuple is the computed SNR using the UPO approximation. To examine the precision of these SNRs, we experimentally compute the SNR (called SNR_e) for all functions and report it as the fourth element of each 4-tuple. To compute these experimental SNRs, we statistically compute the probabilities that the desired functions are constructed when the noise intensity is changed. For this example, we choose the noise intensity such that a given threshold value for noise intensity results in 99.9% success in constructing the desired function. We then use this same noise intensity to compute SNR_p, based on the formula $20 \log \frac{A_{\text{rms}}}{D_{\text{Threshold}}}$. In order to facilitate understanding of the last two elements of the 4-tuples, the estimated SNR and the experimental SNRs, we compute the statistical mean and variance of the differences between these two SNRs, defined as $r = \text{SNR}_p - \text{SNR}_e$, for different iteration numbers, $p-1$. As explained above, SNR_p is the predicted SNR based

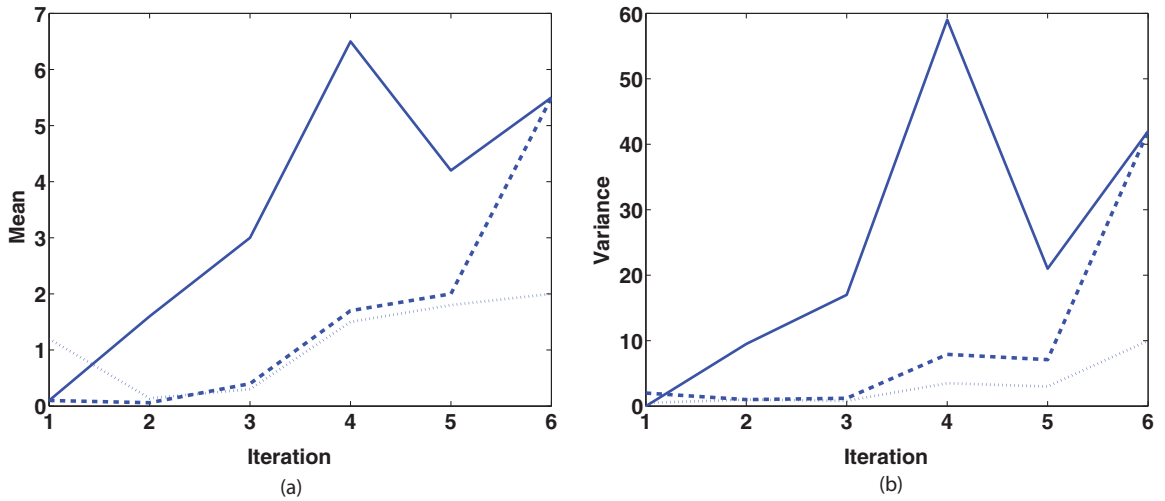

FIG. 2. (Color online) Statistical measures, mean and variance, of the error in estimating robustness of different instructions against noise are reported. The error is the difference between the estimated SNR and the experimental SNR for each instruction. The mean of these errors at each iteration is reported in the left-hand panel, and the variance of the error at each iteration is presented in the right-hand panel. The solid lines denote cases where, for $(p-1)$ iterations of the map, period p UPOs are used for modeling. Dashed lines denote the means and variances when period-7 UPOs are used for predicting the SNR. Dotted lines show the means and variances of the difference r, where a direct slope technique is used.

on UPOs of order p and SNR_e is the experimental SNR. The results are plotted as solid lines in Fig. 2. The overall trend is that with increasing iteration number, the mean and variance of the error signal grow. The predicted SNRs are not very accurate, since we approximate a large portion of the map f or the iterated map $f^{(p-1)}$ with a straight line. To obtain more accurate SNR predictions, we need more precise modeling and approximations. In this example we have computed all the UPOs up to period-7, so an alternative, more precise (and no additional cost) approximation would use these already computed period-7 UPOs for a better calculation of the SNRs for lower iteration numbers as well. The mean and variance of the error $r = \mathrm{SNR}_7 - \mathrm{SNR}_e$, where SNR_7 is the predicted SNR based on UPOs of order 7, is computed for different iteration numbers. The results are presented in Fig. 2 by dotted-dashed lines. We observe that when the iteration number is less than 6, these predicted SNRs are considerably more precise than the previously predicted SNRs, because of more accurate modeling and approximations. Based on Fig. 2, we observe that modeling the chaotic orbits by their nearby UPOs results in a very good approximation of the symbolic behavior of the orbits during limited iteration of the chaotic map. This observation follows the main claim of periodic orbit theory: A collection of short-period UPOs is enough to obtain a very precise approximation of a sufficiently low-dimensional chaotic system [21].

Finally, to examine the accuracy of the approximated SNRs by use of UPOs, we approximate SNRs directly based on the slopes of the orbits, starting from the chosen initial conditions. Thus, instead of finding a nearby UPO and using its robustness measure, we compute the slope of the main orbit at various points on the orbit and use these slopes directly in the formula $D_{\max} = \frac{y}{3\times\sqrt{\lambda_1^2\lambda_2^2\lambda_3^2\cdots\lambda_n^2+\lambda_2^2\lambda_3^2\cdots\lambda_n^2+\cdots+\lambda_n^2+1}}$, where $\lambda_i = f^{(i)'}(x_0)$ and x_0 is the initial condition produced by the encoding map. The mean and variance of the error is plotted in Fig. 2 by dashed lines. We see that using UPOs of order 7 for predicting SNRs is as precise as using direct slopes, when the iteration number is less than 6.

B. Gaussian map

As a second example, we determine the functionality of the Gaussian map for doing computations and estimate the robustness of the resulting functions against noise. The Gaussian map is studied in detail in Ref. [34]. Again, in this example we assume that an additive noise perturbs the dynamics as follows:

$$x_{n+1} = e^{-bx_n} + c + D\varepsilon(t). \tag{10}$$

The phenomenon of chaos is observed in this map at some parameter values [34]. In this paper we set $b = 6.5$ and $c = -0.54$ in order to make the Gaussian map chaotic. The chaotic attractor of the Gaussian map lies in $[-0.28,0.5]$. Similar to the logistic map example, a simple digital-to-analog converter with ten binary digital inputs, two inputs for data and eight inputs for control, will be used as the encoding map. As the first step of the three-step computing method, the combination of two binary data inputs and eight binary control inputs are, with an initial value in $[-0.28,0.5)$, as follows:

$$X_0 = -0.28 + 0.78(0.I_1I_2C_1C_2\cdots C_8)_{\text{base }2}, \tag{11}$$

where I_1, I_2 are the two binary data inputs, and $C_1, C_2, \ldots, C_8$ are the eight control inputs. Notice that the coefficient value, -0.28, and the additive value, 0.78, are inserted to insure that the initial condition is situated inside the attractor.

At the second step of the algorithm, we let the Gaussian map undergo different numbers of iterations in order to determine the instruction set at each iteration number.

TABLE III. Instruction set of the Gaussian map for different iteration numbers.

p	Instruction set		
1	{(7,133,18.90 dB,19.7 dB),	(15,0,45.79 dB,45.89 dB)}	
2	{(9,0,23.62 dB,25.50 dB),	(12,192,33.15 dB,34.20 dB),	(13,118,28.42 dB,28.80 dB)}
3	{(6,0,26.47 dB,26.40 dB), (15,147,36.4 dB,36.5 dB)}	(7,88,36.24 dB,37.2 dB),	(11,246,25.50 dB,25.80 dB),
4	{(6,191,34.45 dB,35.5 dB), (11,88,29.73 dB,33.60 dB),	(7,246,40.37 dB,40.89 dB), (14,138,35.51 dB,36.4 dB),	(10,118,55.71 dB,56.69 dB), (15,0,51.85 dB,51.79 dB)}
5	{(4,89,37.62 dB,39.89 dB), (12,47,35.42 dB,43.19 dB),	(5,106,36.81 dB,38.6 dB), (13,30,40.96 dB,40.39 dB),	(9,187,41.931B,42.29 dB), (15,0,36.88 dB,37.3 dB)}
6	{(2,6,45.40 dB,44.99 dB), (8,244,44.09 dB,47.19 dB), (14,213,47.67 dB,48.79 dB),	(3,22,45.96 dB,48.69 dB), (9,255,62.52 dB,62.39 dB), (15,79,39.77 dB,36.1 dB)}	(7,45,38.81 dB,49.39 dB), (10,223,41.67 dB,51.49 dB),

As the third and last stage of the computing model, the final state of the Gaussian map is decoded to obtain the output

$$\text{output} = \begin{cases} 0 & \text{if} \quad x \leqslant 0, \\ 1 & \text{if} \quad x > 0. \end{cases} \tag{12}$$

There is an important difference between the logistic map and the Gaussian map examples. When the bifurcation value of the logistic map is 4, for any symbolic sequence $X_0, X_1, \ldots, X_p$, there is a unique UPO of length $p + 1$ that has the same symbolic itinerary. This one-to-one relationship between any possible symbolic sequence and a unique UPO describes any other one-humped map, where the attractor is between $[0,b]$ and the critical point x_c is mapped to b [33]. Therefore, the collection of all UPOs of length $p + 1$ can model the behavior of the chaotic map over the next p iterations. But the Gaussian map does not have this property and there are some neighborhoods of initial conditions with admissible symbolic itineraries of length p for which there is no UPO of length $p + 1$ with the same symbolic itinerary. But we know that, since the UPOs are dense over the chaotic attractor, there is therefore at least one UPO that comes inside the neighborhood and which can model this portion of the attractor during the next p iterations. Therefore, we can easily overcome the problem by using higher-order UPOs, such as $p + 2$ or $p + 3$, to model the next p iterations of the map. All we need to do is to compute the preimages of the critical map to find the neighborhood of initial conditions that symbolically behave the same during limited iterations of the map. Then we compute the UPOs until we can find at least one UPO in any neighborhood. This collection of UPOs can be used to model the chaotic map over a limited number of iterations. In the Gaussian map example, we observe that UPOs of length 8 are enough to model the attractor during any iteration up to six iterations. By use of this model we directly compute the instruction set of the chaotic logistic map when it undergoes p iterations. The results are listed in Table III for different values of p, $1 \leqslant p \leqslant 6$. The format of data in Table III is the same as the format in Table I. We observe that, in a noise-free simulation, this technique determines the instruction set of the chaotic system precisely. Also simulation results illustrate that after modeling the Gaussian map by period-8 UPOs, the robustness of the instructions against noise are predicted with very high precision.

VI. CONCLUSIONS

In this paper we have demonstrated how chaotic computation could be explained, modeled, and predicted in terms of the dynamics of the underlying chaotic systems. Unstable periodic orbits of the chaotic system were used first to model it and then to approximate it. These periodic orbits and the polygonal approximations based on them can be used for obtaining the computational functionality (the instruction set) of the system. In this way we have elucidated the deep connection between the structure of the system dynamics and the system's ability to perform computation. This connection intimately depends on the periodic orbit structure of the system.

In a noise-free simulation this technique determined precisely the instruction set of the chaotic system. Then, using UPOs, we examined the robustness against noise of the chaotic system while performing computations. Simulation results illustrate that, by using enough UPOs to model the chaotic system, the predicted SNRs are in close agreement with the experimental SNRs. More specifically, by modeling the logistic attractor with period-7 UPOs, and a Gaussian map by period-8 UPOs, the robustness of the instruction set against noise was predicted with very high precision.

In sum, chaotic dynamical systems provide a fertile base for the construction of computing devices. The intrinsic complexity of a chaotic system, along with its intrinsic controllability (due to its sensitivity to perturbations), allows one to envision a computing device that is simultaneously compact and flexible, capable of highly specialized computation but generic in its ability to morph into different devices as needed.

ACKNOWLEDGMENTS

We gratefully acknowledge support from Dr. Michael Shlesinger of the Office of Naval Research under Grants No. N00014-09-1-0963 (WLD and BK) and No. N00014-11-1-0586 (MLS and BK).

[1] S. Sinha and W. L. Ditto, Phys. Rev. Lett. **81**, 2156 (1998).
[2] S. Sinha and W. L. Ditto, Phys. Rev. E **60**, 363 (1999).
[3] S. Sinha, T. Munakata, and W. L. Ditto, Phys. Rev. E **65**, 036216 (2002).
[4] M.R. Jahed-Motlagh, B. Kia, W. L. Ditto, and S. Sinha, Int. J. Bifurcation Chaos Appl. Sci. Eng. **17**, 1955 (2007).
[5] K. Murali and S. Sinha, Phys. Rev. E **75**, 025201 (2007).
[6] A. Miliotis, K. Murali, S. Sinha, W. L. Ditto, and M. L. Spano, Chaos, Solitons Fractals **30**, 809 (2009).
[7] K. Murali, A. Miliotis, W. L. Ditto, and S. Sinha, Phys. Lett. A **373**, 1346 (2009).
[8] D. Cafagna and G. Grassi, Int. J. Bifurcation Chaos Appl. Sci. Eng. **16**, 1521 (2006).
[9] T. Munakata, S. Sinha, and W. L. Ditto, IEEE Trans. Circuit Syst. **49**, 1629 (2002).
[10] K. Murali, S. Sinha, and W. L. Ditto, Phys. Rev. E **68**, 016205 (2003).
[11] K. Murali, S. Sinha, and W. L. Ditto, Int. J. Bifurcation Chaos Appl. Sci. Eng. **13**, 1 (2003).
[12] K. Murali, S. Sinha, and W. L. Ditto, Pramana J. Phys. **64**, 433 (2005).
[13] H. R. Pourshaghaghi, B. Kia, W. Ditto, and M. R. Jahed-Motlagh, Chaos, Solitons Fractals **41**, 233 (2008).
[14] H. R. Pourshaghaghi, R. Ahmadi, M.R. Jahed-Motlagh, and B. Kia, Int. J. Bifurcation Chaos Appl. Sci. Eng. **20**, 715 (2010).
[15] [http://www.chaologix.com]
[16] W. L. Ditto and S. Sinha, Philos. Trans. R. Soc. A **364**, 2483 (2006).
[17] W. L. Ditto, K. Murali, and S. Sinha, Philos. Trans. R. Soc. A **366**, 653 (2008).
[18] A. Miliotis, K. Murali, S. Sinha, W. L. Ditto, and M. L. Spano, Chaos, Solitons Fractals **42**, 809 (2009).
[19] J. P. Crutchfield, W. L. Ditto, and S. Sinha, Chaos **20**, 037101 (2010).
[20] W. L. Ditto, A. Miliotis, K. Murali, S. Sinha, and M. Spano, Chaos **20**, 037107 (2010).
[21] P. Cvitanovic, Phys. Rev. Lett. **61**, 2729 (1988).
[22] P. Holmes, Phys. Rep. **193**, 137 (1990).
[23] P. Cvitanovic, Physica D **51**, 138 (1991).
[24] R. Artuso, E. Aurell, and P. Cvitanovic, Nonlinearity **3**, 325 (1990).
[25] D. Auerbach, P. Cvitanović, J.-P. Eckamnn, G. H. Gunaratne, and I. Procaccia, Phys. Rev. Lett. **58**, 2387 (1987).
[26] K. Murali, S. Sinha, W. L. Ditto, and A. R. Bulsara, Phys. Rev. Lett. **102**, 104101 (2009).
[27] K. Murali, I. Raja Mohamed, S. Sinha, W. L. Ditto, and A. R. Bulsara, Appl. Phys. Lett. **95**, 194102 (2009).
[28] D. N. Guerra, A. R. Bulsara, W. L. Ditto, S. Sinha, K. Murali, and P. Mohanty, Nano Lett. **10**, 1168 (2010).
[29] A. R. Bulsara, A. Dari, W. L. Ditto, K. Murali, and S. Sinha, Chem. Phys. **375**, 424 (2010).
[30] A. Dari, B. Kia, A. R. Bulsara, and W. Ditto, Europhys. Lett. **93**, 18001 (2011).
[31] H. Ando, S. Sinha, R. Storni, and K. Aihara, Europhys. Lett. **93**, 50001 (2011).
[32] R. Badii, E. Brun, M. Finardi, L. Flepp, R. Holzner, J. Parisi, C. Reyl, and J. Simonet, Rev. Mod. Phys. **66**, 1389 (1994).
[33] R. Gilmore, *The Topology of Chaos: Alice in Stretch and Squeeze Land* (Wiley, Hoboken, NJ, 2002).
[34] V. Patidar, Electron. J.Theor. Physi. **3**, 29 (2006).