ARTICLES

Data-based control trajectory planning for nonlinear systems

Carl Rhodes
Chemical Engineering 210-41, California Institute of Technology, Pasadena, California 91125

Manfred Morari
ETH–Institut für Automatik, ETH-Z/ETL, CH-8092 Zürich, Switzerland

Lev S. Tsimring and Nikolai F. Rulkov
Institute for Nonlinear Science, University of California, San Diego, La Jolla, California 92093-0402
(Received 12 November 1996; revised manuscript received 11 April 1997)

An open-loop trajectory planning algorithm is presented for computing an input sequence that drives an input-output system such that a reference trajectory is tracked. The algorithm utilizes only input-output data from the system to determine the proper control sequence, and does not require a mathematical or identified description of the system dynamics. From the input-output data, the controlled input trajectory is calculated in a ''one-step-ahead'' fashion using local modeling. Since the algorithm is calculated in this fashion, the output trajectories to be tracked can be nonperiodic. The algorithm is applied to a driven Lorenz system, and an experimental electrical circuit and the results are analyzed. Issues of stability associated with the implementation of this open-loop scheme are also examined using an analytic example of a driven Hénon map, problems associated with inverse controllers are illustrated, and solutions to these problems are proposed.
[S1063-651X(97)02709-8]

PACS number(s): 05.45.+b, 47.52.+j, 07.05.Kf, 07.05.Dz

I. INTRODUCTION

Numerous methods for the control of nonlinear systems have been developed recently. In the control community, some of the more popular methods include geometric control methods based on methods from differential geometry (see Ref. [1] for an introduction), nonlinear model predictive control [2], and control based on neural networks [3]. In order to use these methods for control, it is necessary to have an accurate description of the system dynamics. This model can be the result of physical knowledge of the system dynamics or the result of system identification. While these methods are popular in the literature of the control community, different methods of control have been pursued for the control of chaotic systems.

Recently published methods of control for chaotic systems also build controllers based on a knowledge of the system dynamics. However, most of these methods rely on a knowledge of the underlying dynamics of the undriven, autonomous system [4–6]. The chaotic system is then stabilized around an unstable periodic orbit or fixed point using proportional linear feedback control. In the method of Ott, Grebogi, and Yorke (OGY) [4], and a number of later modifications, a scalar-controlled input is changed at discrete times such that a periodic orbit or fixed point of the system becomes stable. The implementation of the OGY algorithm requires knowledge of the linearized dynamics of the periodic orbit to be stabilized (a fixed point on the Poincaré section) and the linearized dynamics which result from small perturbations to the controlled input about some nominal value. This information can be found by linearizing the uncontrolled system dynamics and making small perturbations in the controlled input. Since the goal trajectory in the state space coincides with an existing unstable trajectory of the uncontrolled system, stabilization is achieved by infinitesimal perturbations of the input.

Other feedback methods (occasional proportional feedback control [5], continuous control [6]) also stabilize existing trajectories of the unforced system by making small perturbations in the controlled input. While these methods (with finite driving forces) in principle can be used to drive a system toward an orbit which is not a solution of the unperturbed system, they do not provide an algorithm for finding a driving signal necessary for producing and stabilization of a predefined orbit.

Another class of chaotic control schemes attempts to drive a system such that an arbitrary goal trajectory is tracked. To this end, open-loop (''entrainment'') control schemes have been suggested [7–12]. Originally, entrainment control was utilized on known dynamical systems where the controlled inputs directly affect each state variable of the system [7]. Later this method was generalized for reconstructed dynamical systems [9] and an arbitrary combination of inputs. However, once again it was assumed that the inputs are able to entirely specify the state of the dynamical system [10]. Clearly, the number of controlled inputs cannot be less than the state dimension of the underlying dynamical system when using this control scheme. Additionally, the stability of this open-loop control scheme cannot be guaranteed unless certain conditions are fulfilled. In particular, goal trajectories

must be contained within ''convergent'' regions [8,10] of the state space. A major problem with this method of control is that the problem definition is somewhat artificial. In most physical systems ''full-state control'' (control which directly affects all the states) is not possible since some of the states of the system may not be directly affected by the input.

In the present study, the entrainment control approach will be modified and extended in the following ways. First, it is assumed that the state-space system to be controlled is single input and single output, and equations describing the state-space dynamics are unknown. Since only a single input to the system is assumed, ''full-state control'' is impossible. By choosing the proper input trajectory, the output of the system should track a desired output trajectory. Finally, as the system to be controlled is assumed to be unknown, the proper input will be found using only an input-output time series from the system. This approach will be particularly useful for chaotic systems, where it can be difficult to determine a state-space model which accurately describes the global behavior of the system using standard methods of identification.

Since the method described below utilizes time-series data from the system to compute the proper controlled input, this approach is called data-based control trajectory planning. In order to accomplish this task, input-output identification data which characterize the dynamics of the *driven* nonlinear system are needed. The identification data consist of a time series collected from the driven system with random variations in the driving input. The set of goal trajectories which can be tracked by this control scheme should consist of the set of all trajectories which are possible for the driven system, which is larger than the set of trajectories of the undriven system. In this paper, input trajectories will be calculated off-line in an open-loop fashion. However, this same method could be used for closed-loop control with only minor modifications which will also be described.

Since the trajectory to be tracked may be unstable, it is possible that the computed open-loop control trajectory may not stabilize the system. This is because it is difficult to exactly cancel the instability present in the trajectory of the open-loop system. In this case, additional closed-loop feedback control may be necessary to stabilize the system. It is also possible that the ''inverse'' mapping which produces the open-loop control law may be unstable. This would lead us to believe that the dynamic system contains nonminimum phase behavior, and the system may exhibit problems very similar to internal instability problems which can be found in linear systems when inverse controllers are used [13]. Both of these problems will be illustrated and examined in more detail in the examples.

II. METHOD

Consider the following nonlinear dynamical systems:

$$\dot{\mathbf{x}} = f(\mathbf{x},u), \quad y = h(\mathbf{x}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^d$ is a d-dimensional vector of state variables, u is a scalar controlled input, and $h:\mathbb{R}^d \rightarrow \mathbb{R}$ is a measurement function. The goal of the trajectory planning algorithm is to find a time series $u_0(t)$ which generates a specified output series $y_0(t)$ when applied to the system.

In control theory, a system is called ''output controllable'' when it is possible to use a controlled input to produce any desired output [14]. While theorems exist which allow us to determine the controllability of nonlinear systems [1], these theorems are dependent on a state-space model of the system. In situations where no state-space model (1) exists and only an input-output time series is available, determining whether a system is globally controllable is a problem which has not been solved to the authors' knowledge. However, preliminary results in this area do exist. In a recent paper [15], a computational algorithm for computing controllable sets directly from time-series data is outlined. In this work, it is simply assumed that the goal trajectories can be produced by an input trajectory in the examples.

In recent papers [16,17], the Takens embedding theorem is extended to deal with input-output systems. Specifically, for system (1) future outputs can be generically represented as a function of time-delayed versions of the input and output (assuming the input remains constant between sampling times) as follows:

$$y(t) = P[y(t-T), y(t-2T), \ldots, y(t-lT), u(t-T), \ldots, u(t-mT)] \tag{2}$$

where T is an appropriate time delay (in theory, the choice of T is arbitrary), and $l,m \geqslant d+1$.

While this model is guaranteed to exist, for most physical systems only identification is available and the exact form of the state-space dynamics (1) is unknown *a priori*. Additionally since $l,m \geqslant d+1$ is only a sufficient (and not a necessary) condition for a model of the form (2) to exist, there may be l and m smaller than $d+1$ such that Eq. (2) exists. For these reasons, a way of determining the smallest values of l and m from input-output time-series data has been developed using an extended version of the false-nearest-neighbors (FNN) algorithm [18]. Once the proper number of embedded terms on the right-hand side of Eq. (2) has been determined, the function P can be described locally for predictive purposes using nonlinear modeling techniques based on local polynomial predictors [16,19,20]. Given a known input series, the system output could be predicted by repeated ''one-step ahead'' prediction.

For purposes of open-loop trajectory planning, the input sequence $u(t)$ should be determined as a function of a desired output sequence $y(t)$. Using the implicit function theorem, Eq. (2) can be locally inverted as

$$u(t) = Q(y(t+T), y(t), y(t-T), y(t-2T), \ldots, y[t-(l-1)T], u(t-T), \ldots, u[t-(m-1)T]). \tag{3}$$

Given the y terms (the values of the goal trajectory), this equation represents an m-dimensional nonautonomous mapping for the desired control u. Just as in the case of modeling the output dynamics (2), this inverse map can be recovered from the data by using local polynomial models in the space of delayed versions of y and u. Once this map is determined

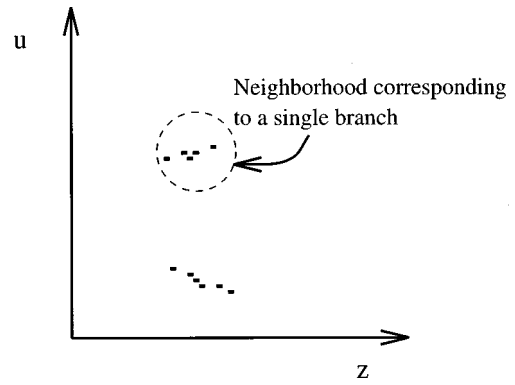

FIG. 1. Example of a nonunique inverse mapping for the input u. Only a set of points which are close with regard to the input u can be used here for the local linear approximation.

locally, it can be used to calculate the ''one-step-ahead'' control move $u(t)$ that will give the desired reference output $y(t+T)$.

Two problems may be encountered utilizing this local inversion process. First, it is possible that the control needed to produce the desired output is not contained in the data set. In this case, further identification with an input signal which has either a larger magnitude range or a wider frequency range may be needed. Second, the mapping Q from Eq. (3) is not guaranteed to be unique. For nonunique inverse mappings only the data corresponding to one branch of the inverse map can be used for inverse modeling purposes (see Fig. 1). If the data from both branches are used for building a model, the model will ''average'' the data from the two branches, and the resulting computed future control move u will lie somewhere between the two branches. In this case, the computed control move will not produce the desired action.

III. COMPUTATIONAL ALGORITHM

The goal of the computational algorithm presented here is to determine an input trajectory which will produce a desired output trajectory when applied to the system. It will be shown that the proper input trajectory can be computed directly in an open-loop fashion from an input-output time series of the system. Traditionally two distinct steps are completed for controller design. First, input-output identification data are analyzed, and a model of the dynamics is formed. Then a controller is designed using the identified model and the controller is implemented on the actual system. Here, a controller will be designed which determines the input control trajectory *directly* from input-output time-series data of the system. This method of control is more computationally intensive than open-loop control schemes which utilize fixed control laws; however, it may give better results for systems with complicated dynamics where accurate identification is difficult.

While the computational algorithm does not need a global description of the dynamics, the number of delayed terms needed to recreate the dynamics (l and m) is needed. From this information, a local model of the dynamics in the neighborhood around the desired trajectory is built utilizing time-series data from the training set. By using data which are ''near'' to the desired dynamics [in the sense of distance in the regression space defined by the right-hand side of Eq. (3)], a locally valid linear model of the dynamics is identified and the proper control move is computed. Once the proper control move is calculated, the process is repeated. This method is different from the OGY method [4], since a local linear model is formed for each sampling time of the system. In the OGY method, the control move is implemented periodically and the trajectories to be tracked must also be periodic. The method proposed here is not limited to tracking periodic signals.

Here is an outline of the computational algorithm for open-loop control trajectory planning.

(1) The data set is presorted using the method of Grassberger [21] in order to reduce the search time needed by the algorithm. The training data are sorted into a two-dimensional grid to save time when searching for neighbors within distance δ of a given point in the space $\mathbb{R}^{l+m}$. The data are presorted into two-dimensional bins in the regression space of the mapping in Eq. (3) $[(y(t+T),\ y(t),\ \ldots,\ \ y[t-(l-1)T],\ u(t-T),\ \ldots,u[t-(m-1)T])]$.

(2) For the first step, the inverse mapping is initialized with an input sequence. Since the desired output trajectory $y_0(t)$ is known, the delay coordinate vector $\mathbf{z}_0(t) = (y_0(t+T), y_0(t), \ldots, y_0[t-(l-1)T], u_0(t-T), \ldots, u_0[t-(m-1)T])$ is needed to determine the first control move, where $u_0(t-T), \ldots, u_0[t-(m-1)T]$ are the initialized input terms. After the first step, the vector $\mathbf{z}_0$ is formed from the goal output trajectory and past inputs computed by the algorithm.

(3) The training time series is searched for points such that $\|\mathbf{z}_{\text{train}}(k)-\mathbf{z}_0(t)\|_\infty \leq \delta$, where $\mathbf{z}_{\text{train}}(k)$ consists of the time-delay embedded data from the training set. This search is facilitated by the fact that the data are presorted. Points from the time series which are neighbors are then arranged into a matrix containing the time-delay-embedded terms $\mathbf{z}_{\text{train}}(\text{NN})$ and a vector of inputs $u_{\text{train}}(\text{NN})$ in the following manner,

$$\mathbf{X} = \begin{bmatrix} \mathbf{z}(\text{NN}_1) \\ \mathbf{z}(\text{NN}_2) \\ \vdots \\ \mathbf{z}(\text{NN}_p) \end{bmatrix} = \begin{bmatrix} y(\text{NN}_1+T) & y(\text{NN}_1) & \cdots & u(\text{NN}_1-(m-1)T) \\ y(\text{NN}_2+T) & y(\text{NN}_2) & \cdots & u(\text{NN}_2-(m-1)T) \\ \vdots & \vdots & \ddots & \vdots \\ y(\text{NN}_p+T) & y(\text{NN}_p) & \cdots & u(\text{NN}_p-(m-1)T) \end{bmatrix}, \tag{4}$$

$$\mathbf{u} = \begin{bmatrix} u(\text{NN}_1) \\ u(\text{NN}_2) \\ \vdots \\ u(\text{NN}_n) \end{bmatrix}, \tag{5}$$

where $\mathbf{z}$ and u are deviation variables about the point $\mathbf{z}_0$ which we are interested in. To solve for the unknown param-

eters in the linear model, a weighted least-squares problem is solved. Specifically, the least-squares problem

$$\mathbf{WX}\theta = \mathbf{Wu} \tag{6}$$

is solved for θ, where $\mathbf{W}$ is a diagonal matrix of weights (the weightings consist of a radial-basis function which penalizes distance from $\mathbf{z}_0$). The desired input move $u_0(t)$ is then calculated from the following equation:

$$u_0(t) = \mathbf{z}_0 \theta. \tag{7}$$

Note that this is a local linear approximation to Eq. (3).

(4) t is increased and the previous two steps are repeated. By repeating this process, the proper input trajectory is determined one step at a time.

Since the algorithm builds a description of the dynamics locally about the trajectory to be tracked, a global description of the dynamics is not required.

While the algorithm above describes open-loop calculation of the control law, closed-loop control could be performed by modifying the delay coordinate vector $\mathbf{z}_0$. For closed-loop operation, $\mathbf{z}_0$ would take the form

$$\mathbf{z}_0 = (y_0(t+T), y_M(t), \ldots, y_M[t-(l-1)T], u_0(t - T), \ldots, u_0[t-(m-1)T]), \tag{8}$$

where the y_M terms are the measured outputs from the system. With this change, the control law could be calculated online in a ''one-step-ahead'' manner. The only limitation to this closed-loop method is that the sampling time must be larger than the computation time needed to determine the next control move $u_0(t)$.

Currently, the algorithm does not account for the possibility that the inverse mapping (3) may not be unique. Another problem which could be encountered is an unstable inverse mapping (3). This instability may result in an input which becomes unbounded. If the dynamics of the system do not exactly cancel this input, the output behavior will not track the desired trajectory. However, as we will see in Sec. IV, it is possible that inverse mappings which are unstable may lead to acceptable results for open-loop control purposes. In addition, the training set must cover the entire range of inputs needed for the proper control trajectory. There is currently no way to determine the proper range of inputs *a priori*. If the range of inputs is not large enough, there will be no ''near neighbors'' to the vector $\mathbf{z}_0$, and a local linear model cannot be built.

IV. APPLYING THE COMPUTATIONAL ALGORITHM

In this section, the computational algorithm will be applied to two examples. The first example is the simulated driven Lorenz equations. The second example describes the application of the computational algorithm to an experimental electronic circuit which exhibits chaotic dynamics.

A. Driven Lorenz model

In this section, the computational algorithm is applied to the following driven Lorenz system:

$$\dot{x} = \sigma(y-x), \quad \dot{y} = -xz + rx - y + \epsilon[u(t)-y], \quad \dot{z} = xy - bz. \tag{9}$$

The parameter values $r = 45.62$, $b = 4$, and $\sigma = 16.0$, which correspond to chaotic behavior of the undriven system, are used. System (9) has a controlled input $u(t)$ which appears only in the equation for $\dot{y}$. The output of the system is $x(t)$, and we would like to drive the system such that a desired periodic trajectory $x_0(t)$ is produced. For large values of ϵ, it is expected from studies on synchronization that open-loop implementation of the computed control trajectory will be stable [22].

A control signal which causes the output to track the desired trajectory will be constructed from identification data using the methods illustrated previously. First, the Lorenz system is subjected to driving by a random input signal $u_{\text{train}}(t)$ obtained by passing white noise through a low-pass filter [the cutoff frequency of the filter is taken to approximately correspond to the frequency range of intrinsic oscillations of $x(t)$]. $u_{\text{train}}(t)$ and $x_{\text{train}}(t)$ are recorded using a sampling time of 0.02, and the time series is of length 50 000. The time delay T is found using average mutual information analysis (see Ref. [20]) of the input-output data, and the appropriate embedding dimensions $l = 2$, $m = 2$ are calculated by applying the input-output false-nearest-neighbors algorithm to the data. The identification data are used by the trajectory planning algorithm to form local inverse maps of form (3). A threshold distance of $\delta = 1.0$ is used to determine if points from the time series are considered as neighbors for the local modeling.

The data-based entrainment algorithm is used to calculate the input for driving the system such that the output trajectory $x_0(t) = 20 \sin(0.523t) + 5$ is produced. The value of the parameter $\epsilon = 20.0$ (9) is used in this example, and the results are presented in Fig. 2. In Figs. 2(a) and 2(b) parts of training sets $u_{\text{train}}(t)$ and $x_{\text{train}}(t)$ are shown. Since the output trajectory to be tracked is periodic, the trajectory planning algorithm is run until the input signal converges to a periodic signal (the initial transient is discarded). Figure 2(c) shows the control signal $u_1(t)$ obtained after the output of the data based trajectory planning algorithm converges to a periodic signal. Figure 2(d) shows the desired output behavior $x_0(t)$ (dotted line) and the output $x_1(t)$ produced by the Lorenz system (9) when driven by the control signal $u_1(t)$ (solid line).

A possible reason that tracking is poor near the ''top'' of the sine wave is that the local linear mapping for the input is unstable for portions of the trajectory. It is expected that the data-based scheme will compute the exact inverse of the Lorenz system; however, if this inversion is not exact the system will not track the desired output exactly. A plot of the output and the location of the single pole of the local inverse mapping as a function of time are given in Fig. 3. Any time the pole is greater than 1, the inverse mapping is unstable. Notice that fairly long-term instability of the inverse mapping appears to correspond to poor tracking of the desired output (seen from time 60 to 70). However, the other region of instability where the unstable pole is quite large (near time

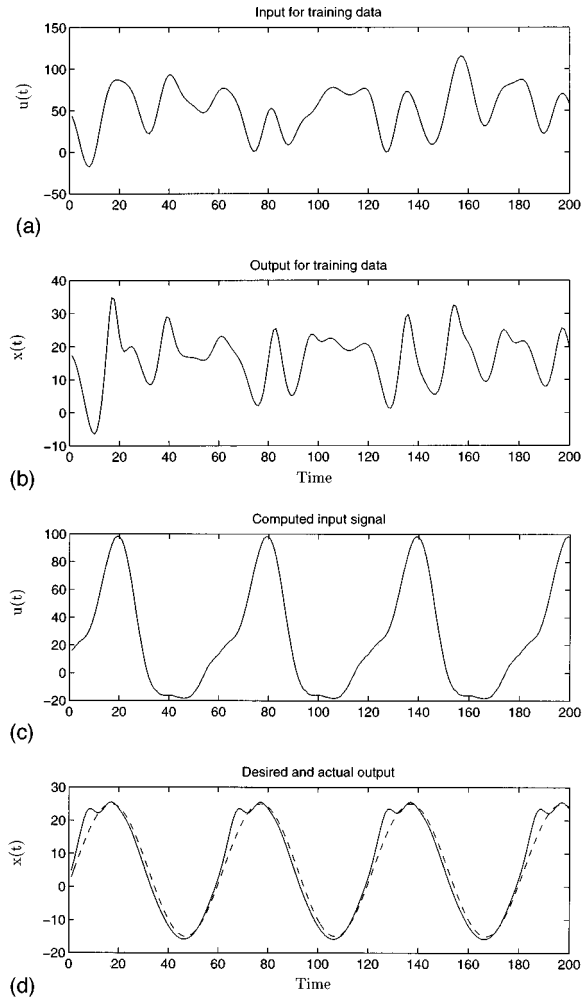

FIG. 2. Data-based trajectory planning for Lorenz system (9) with $\epsilon=20.0$. The goal trajectory of the output variable, $x_0(t)=20\sin(0.523t)+5$. The inverted map was reconstructed from test driving the Lorenz system by randomized input. A time-series of length 50 000 was used, and the parameters of the model were chosen: $l=2$ and $m=2$. (a) and (b) Simultaneous time series of the input $u(t)$ and output $x(t)$ from the training time-series. (c) Control sequence calculated using data-based trajectory planning. (d) Output signal resulting from the computed input signal.

100) does not seem to affect tracking of the output possibly because the cancellation of the systems dynamics is nearly exact in this region of the dynamics.

Figures 4(a) and 4(b) show input and output time series for tracking of a more complicated signal $x_2(t)=10\sin(t)+10\cos(0.5t)+5$. Again, a reasonably good reconstruction of the desired output trajectory is achieved using only this open-loop trajectory planning technique. In Fig. 5 the previous experiment is repeated for $\epsilon=5.0$ in system (9). Here, the computed control sequence $u_2(t)$ does not accurately track the desired output trajectory. Since the system does not converge to the desired periodic output trajectory, it appears that the linearized dynamics around the desired trajectory may be unstable for this system. A possible remedy for this situation is to use a closed-loop feedback stabilization technique (possibly the OGY method) in a periodic manner to stabilize the dynamics about the desired trajectory.

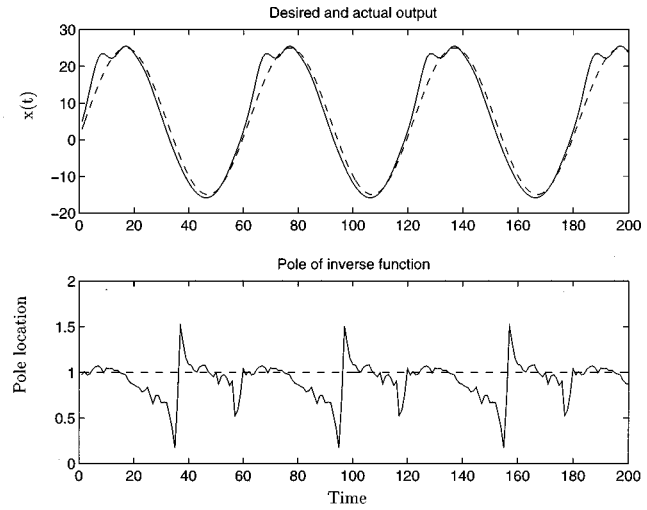

FIG. 3. Illustration of instability of the inverse mapping. The pole of the local linear inverse is plotted as a function of time.

B. Electronic circuit

In this section, the computational algorithm described in Sec. III will be applied to control the dynamics of a nonlinear electronic circuit in a physical experiment. In the experiment, a low-frequency (about 300 Hz) nonlinear circuit whose diagram is shown in Fig. 6 is used. The circuit consists of a nonlinear converter, linear feedback, and an input block. The nonlinear converter is implemented using operational amplifier U1A and the multiplier U2. The shape of the nonlinear function $F(w)$ generated by the converter was measured experimentally [see Fig. 7(a)]. The linear feedback contains three integrators (U3A, U3B, and U3C) and summers (U3D and U1B) which create three-dimensional phase space (x,y,z) of the nonlinear circuit. The input block is built using OP amplifiers U4A and U4B. This section takes signals $x(t)$ and the external input $u(t)$ to form the output $\epsilon[x(t)-u(t)]$, which is applied to terminal A of the switch SW1. The value of the parameter ϵ is determined by the resistor R_{cont}.

When the switch SW1 is in position B, the circuit operates without external inputs $[w(t)=x(t)]$ and generates cha-

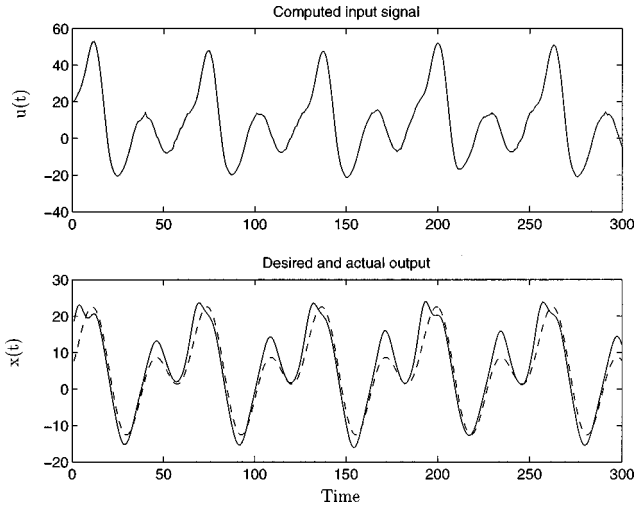

FIG. 4. Output tracking for goal dynamics consisting of two periodic components.

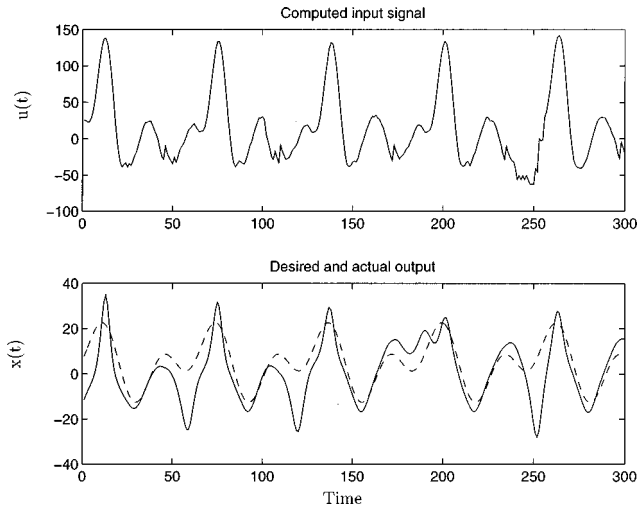

FIG. 5. Tracking of the output signal of Fig. 4, for system (9) with $\epsilon = 5.0$. In this case, the open-loop control cannot successfully track the output signal.

otic oscillations. The projection of this chaotic attractor onto the plane $[x(t), y(t)]$ is given in Fig. 7(b). When switch SW1 is in position A, the circuit is affected by the external input $u(t)$. In position A, the input to the nonlinear converter is $w(t) = x(t) - \epsilon[x(t) - u(t)]$. In this experiment, $\epsilon = 0.8$ is used.

A band-limited random training signal $u_{\text{train}}(t)$ is generated using a sampling rate of 1000 samples per second and this input is applied to the circuit. The input $u_{\text{train}}(t)$ and output $-y_{\text{train}}(t)$ of the system are recorded. This input-output time-series (30 000 points) is used by the algorithm proposed before to construct the input $u(t)$ which will cause

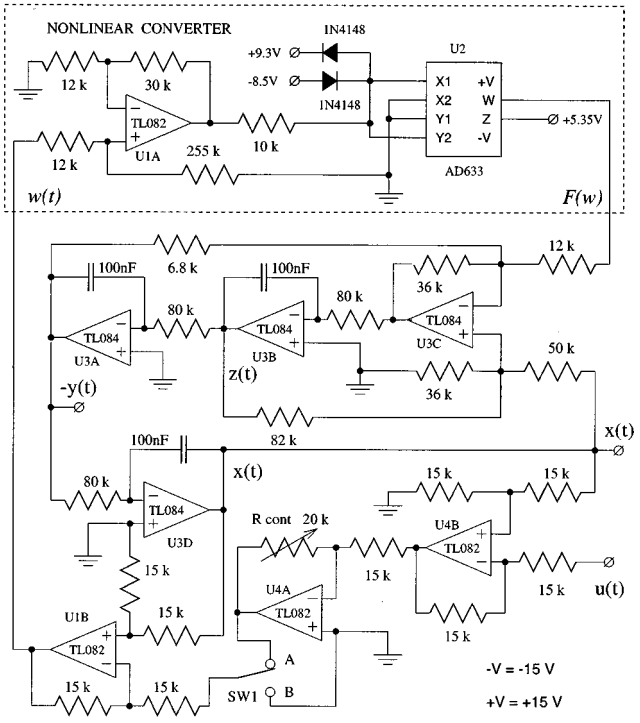

FIG. 6. The diagram of the nonlinear circuit used in the experimental studies.

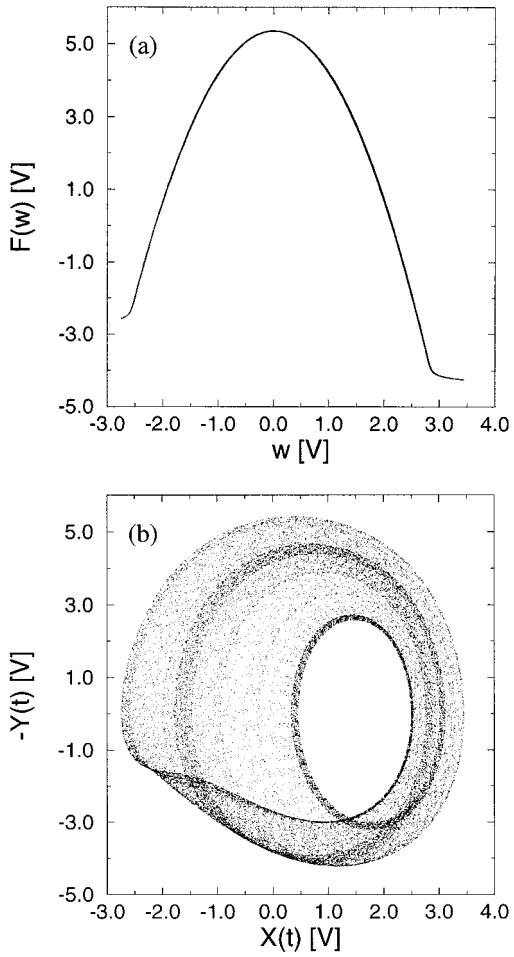

FIG. 7. (a) The shape of the nonlinear converter. (b) The projection of the chaotic attractor measured from the circuit with the sampling rate 1000 sample/s (SW1 in the position B).

the nonlinear circuit to track a desired output trajectory $-y_0(t)$. A portion of the training time series is given in Fig. 8. When the time series is analyzed by the input-output false-nearest-neighbors algorithm, it is found that the proper number of embedding terms is $l = 3$, $m = 2$.

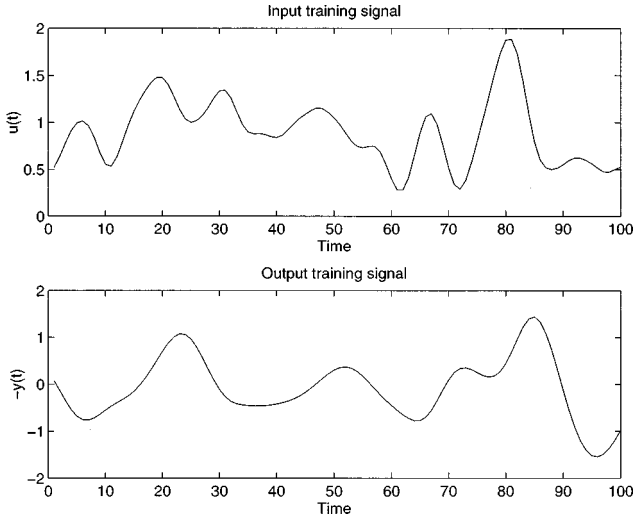

FIG. 8. Portion of the training time series for the nonlinear circuit.

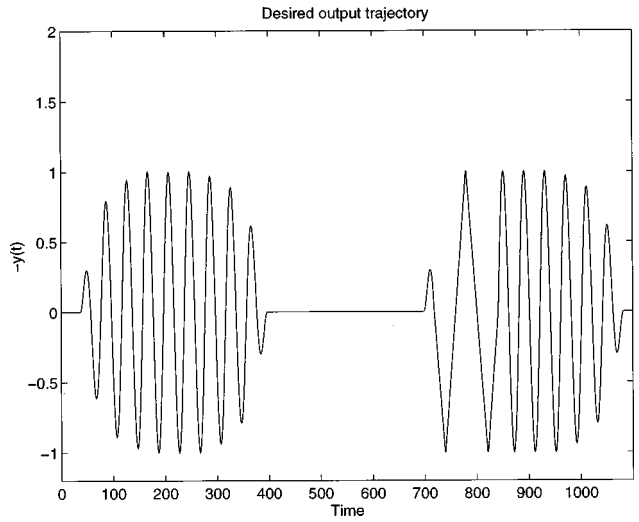

FIG. 9. The desired goal trajectory for the nonlinear circuit.

The trajectory to be tracked for this example given in Fig. 9 is nonperiodic. The goal output trajectory consists of a sine wave with varying amplitude (time 50–400 and 900–1100), a constant value (400–700), and a piecewise linear signal (700–900). When this goal trajectory and the training set are input to the trajectory planning algorithm using a neighbor threshold distance $\delta=0.25$, the result is the input sequence shown in Fig. 10. When this input sequence is used to drive the circuit, the measured output of the circuit tracks the goal trajectory quite well (see Fig. 11). The major differences between the goal trajectory and system outputs occur during the transition which starts and ends the piecewise linear signal at times 700 and 850. Since closed-loop control is not implemented in this example, the poor tracking at these times is not corrected online.

This example shows that this trajectory planning method can accurately track signals which are nonperiodic. The input signal is computed without any mathematical or identified description of the system dynamics. In addition, the output trajectory is not an existing trajectory of the chaotic attractor. This clearly demonstrates the difference between this proposed method and other methods of chaos control based on stabilization of periodic orbits of a nonlinear system.

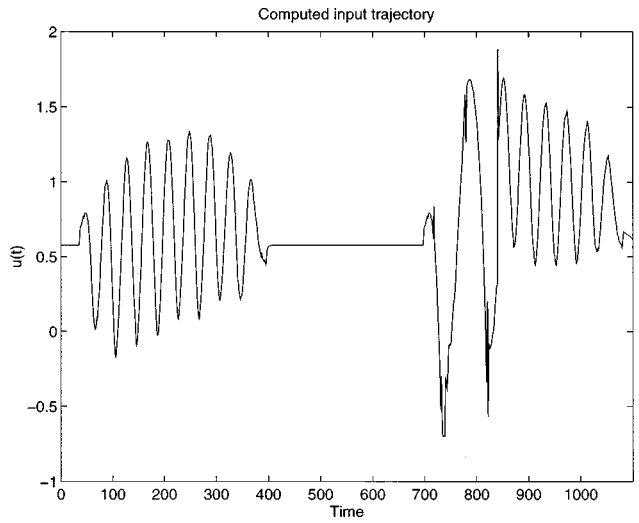

FIG. 10. The computed input from the trajectory planning algorithm.

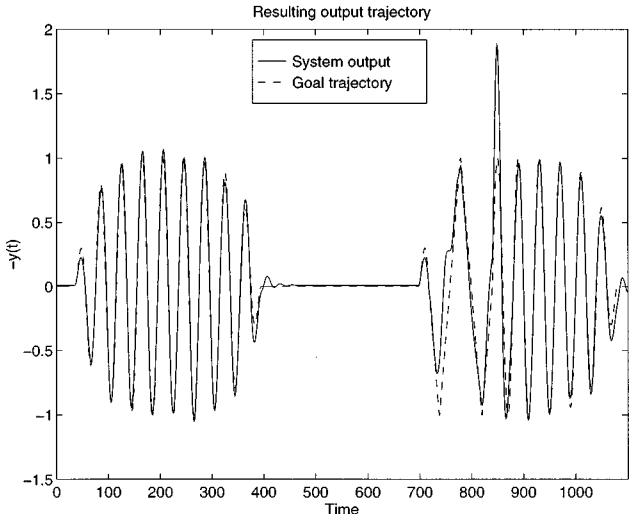

FIG. 11. A comparison of the goal trajectory and the output of the circuit when the computed input from the trajectory planning algorithm is applied.

V. INVERSE MODELING AND CONTROL OF THE HENON MAP

Issues of stability related to inverse modeling are better illustrated by examining a simple analytical example with discrete-time dynamics. Analytical inversion of the model system for control is a well-known method in the literature, so the main purpose of this section is to emphasize problems which could be encountered when using the data-based trajectory planning algorithm (which performs a local computed inversion of the system), and demonstrate methods for overcoming these problems. Consider the following driven Henon maps:

$$x_{n+1}=1-ax_n^2+y_n+u_n, \quad y_{n+1}=bx_n+cu_n, \tag{10}$$

where x_n is an observable output, and u_n is a controlled input. An exact ''reconstructed'' dynamical system in the embedding space $\{x_n,u_n\}$ takes the form

$$x_{n+1}=1-ax_n^2+bx_{n-1}+u_n+cu_{n-1}. \tag{11}$$

Suppose we would like to generate a period-2 output sequence $\{x_1,x_2,x_1,x_2,\ldots\}$ with prescribed values x_1 and x_2. Then we need to find a periodic sequence of inputs $\{u_1,u_2,u_1,u_2,\ldots\}$ such that $x_{2k}=x_1$, $x_{2k+1}=x_2$. By algebraic manipulation, the solution is

$$u_{1,2}=(1-c^2)^{-1}[x_{2,1}-1+ax_{1,2}^2-bx_{2,1} -c(x_{1,2}-1+ax_{2,1}^2-bx_{1,2})]. \tag{12}$$

Suppose we use the inputs $u_{1,2}$ from Eq. (12) and input them to the system in an open-loop fashion. With this choice of $u_{1,2}$ the periodic orbit of interest is a fixed point of the map

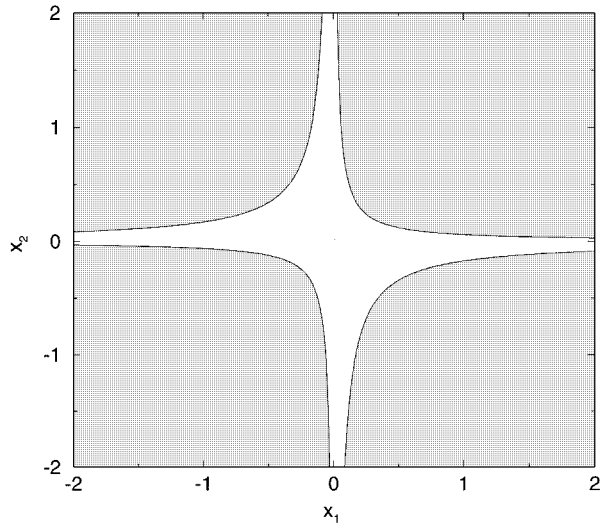

FIG. 12. Region of stability of the open-loop control of the Henon map (10) in the plane of parameters x_1 and x_2 of the goal trajectory for $a=1.4$ and $b=0.3$ (unshaded).

$$\mathbf{Y}_{2m+2}=\mathbf{P}_2(\mathbf{Y}_{2m},u_{2m+1},u_{2m},u_{2m-1}) \equiv\mathbf{P}[\mathbf{P}(\mathbf{Y}_n,u_{2m},u_{2m-1}),u_{2m+1},u_{2m}], \quad (13)$$

with $u_{2m\pm1}\equiv u_1$, $u_{2m}\equiv u_2$, $\mathbf{Y_n}\equiv\{x_n,x_{n-1}\}$. This solution can either be stable or unstable depending on the eigenvalues of the Jacobian of system (13), or equivalently Eq. (11), calculated over a periodic orbit,

$$\mathbf{DP_2}=\begin{pmatrix}-2ax_1 & b\\ 1 & 0\end{pmatrix}\begin{pmatrix}-2ax_2 & b\\ 1 & 0\end{pmatrix}. \quad (14)$$

The region of stability on the plane (x_1,x_2) is defined by two sets of hyperbolas (Fig. 12). Inside this region the map will be stable and the periodic control sequence (12) can be used in open-loop fashion. This region of stability is equivalent to the convergent regions of Ref. [8]. Outside the stability region, an additional closed-loop stabilization technique is needed [23].

A modification of the pole placement method can be used to stabilize this system (see, e.g., Ref. [24]). In addition to the two components of vector $\mathbf{Y}$, an additional linear equation for u_{2m} is added so the system to be analyzed (13) becomes

$$\mathbf{Y}_{2m+2}=\mathbf{P}(\mathbf{Y}_{2m},u_{2m+1},u_{2m},u_{2m-1}),$$
$$u_{2m+2}=u_1+\mathbf{K}\cdot\mathbf{Y}_{2m}+g(u_{2m}-u_1). \quad (15)$$

The control coefficients $[\mathbf{K}=(k_1,k_2)$ and $g]$ can be calculated by specifying the eigenvalues of the Jacobian of this system [24]. In this example, all three eigenvalues are placed at zero.

In Fig. 13, two examples of controlling a period-2 orbit in the driven Henon map are shown. Figure 13(a) demonstrates the application of analytic open-loop control for the periodic trajectory $x_1=0.3$, $x_2=0.2$. The eigenvalues of the open-loop system about the desired trajectory are 0.978 and 0.092, so the open-loop system is stable. Figure 13(b) shows that open-loop control fails for the period-2 signal $x_1=0.3$,

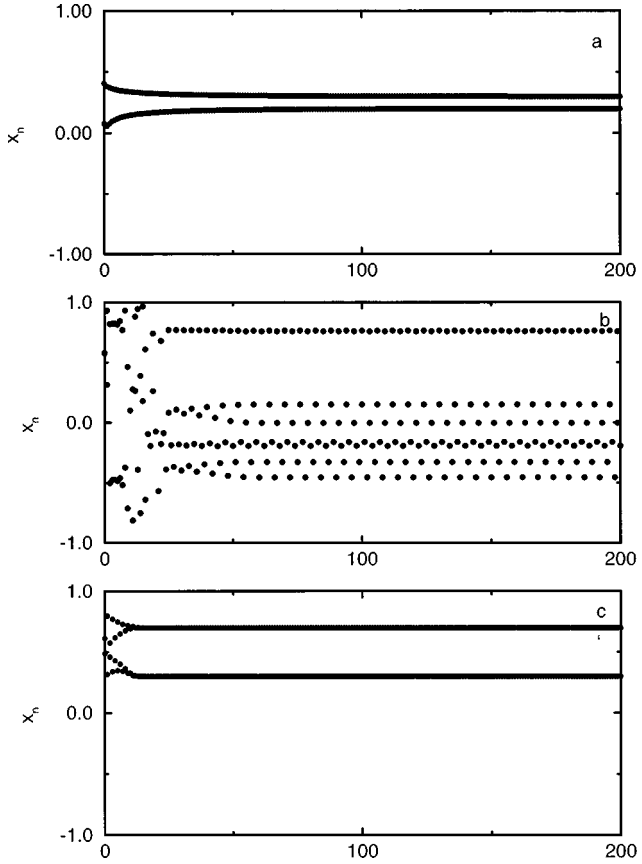

FIG. 13. (a) Successful open-loop control of a period-2 orbit of the Henon map with $x_1=0.3$ and $x_2=0.2$. (b) $x_1=0.3$ and $x_2=0.7$—open-loop control fails. (c) Successful closed-loop control for $x_1=0.3$ and $x_2=0.7$, with feedback corrections calculated by the pole placement method using Eq. (15) to stabilize the goal orbit.

$x_2=0.7$. The eigenvalues of the open-loop system are 2.206 and 0.041, so the open-loop system is unstable and will not produce the desired output trajectory. However, by using pole placement feedback (closed-loop) control, the desired trajectory can be stabilized, as seen in Fig. 13(c).

In the previous examples, initial conditions in the neighborhood of the trajectory to be tracked were utilized. Under this assumption, the presented linear stability analysis should be valid. If the Henon map is driven under open-loop control from arbitrary initial conditions, the system could settle on an undesired orbit, as happens in Fig. 13(b) where the map settles on a period-7 orbit. Simply waiting to start the control trajectory when the system comes close to the desired orbit may not be feasible, since the desired trajectory may not belong to the attractor of the undriven system. For these reasons, special care must be taken when bringing the system from arbitrary initial conditions to a desired trajectory.

The system could be brought from arbitrary initial conditions to the desired trajectory by computing the controllable sets of a point on the trajectory. The time n controllable set of a reference point in the embedded space consists of a set of points which can be controlled to that reference using exactly n input moves [15]. By finding the smallest time controllable set of a point on the reference trajectory, the inputs needed to drive the system to a point on the reference trajectory in the shortest time are found.

This analytic example was presented in some detail in

order to illustrate problems which may be encountered when the data-based trajectory planning algorithm is applied. First, the inverse mapping reconstructed from the training data may prove unstable, and then some other method may be needed to determine the proper input trajectory in an open-loop fashion. Second, the open-loop implementation of the computed control sequence may be unstable. In this case, some form of closed-loop tracking of the original control sequence will be required. The work on combining the closed-loop stabilization with data-based trajectory planning is in progress.

VI. CONCLUSIONS

In this paper an algorithm was presented which computes a control trajectory which will drive a nonlinear system such that a specified output trajectory is produced. This output trajectory does not need to coincide with a trajectory of the undriven system, and the algorithm which was presented for computing the trajectory relies only on time-series data. The main difference between our approach and other previously published ''entrainment control'' methods is that no *a priori* knowledge of the system is assumed. In addition, it is not assumed that the inputs directly affect all states of the system. We are interested in tracking a goal trajectory specified only in terms of a (scalar) output signal. Accordingly, instead of a full state control (which is rare in most physical systems of interest), it is assumed that only a single input to the system is available. When determining the proper input trajectory, a locally valid inverse model is constructed utilizing time-series data from a training set. The inverse model (3) is then used to determining the proper input trajectory in an open-loop one-step-ahead fashion.

Work for the future includes a closed-loop implementation of the data-based control algorithm, application of the algorithm to nonlinear systems which do not exhibit chaos, and extension of these data-based control schemes for robustness.

ACKNOWLEDGMENTS

The authors are indebted to Henry Abarbanel and Reggie Brown for useful discussions. Partial support from the Department of Energy (Grant No. DE-FG03-95ER14516) and Electric Power Research Institute (Grant) No. WO8015-11) are gratefully acknowledged.

[1] A. Isidori, *Nonlinear Control Systems* (Springer-Verlag, Berlin, 1989).
[2] E. S. Meadows and J. B. Rawlings, in *Nonlinear Process Control*, edited by M. A. Henson and D. E. Seborg (Prentice-Hall, Englewood Cliffs, NJ, 1997), p. 233.
[3] H. T. Su and T. J. McAvoy, in *Nonlinear Process Control* (Ref. [1]), p. 371.
[4] E. Ott, C. Grebogi, and J. Yorke, Phys. Rev. Lett. **64**, 1196 (1990).
[5] B. Peng, V. Petrov, and K. Showalter, Physica A **188**, 210 (1992).
[6] K. Pyragas, Phys. Lett. A **170**, 421 (1992).
[7] A. Hübler, Helv. Phys. Acta **62**, 343 (1989).
[8] E. A. Jackson, Phys. Lett. A **151**, 478 (1990).
[9] J. L. Breeden, Phys. Lett. A **190**, 264 (1994).
[10] R. Mettin, A. Hübler, A. Scheeline, and W. Lauterborn, Phys. Rev. E **51**, 4065 (1995).
[11] M.-C. Ho, J.-L. Chern, and D.-P. Wang, Phys. Lett. A **194**, 159 (1994).
[12] C.-C. Chen, Phys. Lett. A **213**, 148 (1995).
[13] M. Morari and E. Zafiriou, *Robust Process Control* (Prentice-Hall, Englewood Cliffs, NJ, 1989).
[14] R. Balasubramanian, Continuous Time Controller Design, 1989.
[15] C. Rhodes and M. Morari, in *Proceedings of the 1996 IFAC World Congress, San Francisco, 1996* (Pergamon, Oxford, 1996), Vol. M, p. 13.
[16] M. Casdagli, in *Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity*, edited by M. Casdagli and S. Eubank, Proceedings of Vol. XII (Addison-Wesley, Reading, MA, 1992), p. 265.
[17] A. Poncet, J. L. Poncet, and G. S. Moschytz, in *Proceedings of the 1995 IEEE Symposium on Circuits and Systems* (IEEE, New York, 1995), Vol. 2, p. 1500.
[18] C. Rhodes and M. Morari, in *Proceedings of the 1995 ACC Conference, Seattle, 1995* (Amer. Autom. Control Council, Evanston, IL, 1995), Vol. 3, p. 2190.
[19] N. F. Hunter, *Nonlinear Modeling and Forecasting, SFI Studies in the Sciences of Complexity* (Ref. [6]), p. 467.
[20] H. D. I. Abarbanel, R. Brown, J. J. Sidorowich, and L. S. Tsimring, Rev. Mod. Phys. **65**, 1331 (1993).
[21] P. Grassberger, Phys. Lett. A **148**, 63 (1990).
[22] L. Kocarev and U. Parlitz, Phys. Rev. Lett. **76**, 1816 (1996).
[23] Since we wish to change the parameter u only once every period from its nominal value u_1 (keeping u_2 constant), no generalizations of original OGY algorithms due to time-delay embedding [U. Dessler and G. Nitsche, Phys. Rev. Lett. **68**, 1 (1992); P. So and E. Ott, Phys. Rev. E **51**, 2955 (1995)] are needed.
[24] Z. Hong, Y. Jie, W. Jiao, and W. Yinghai, Phys. Rev. E **53**, 299 (1996).