PHYSICAL REVIEW E VOLUME 54, NUMBER 4 OCTOBER 1996

Neural network model to control an experimental chaotic pendulum

Rembrandt Bakker,[1,*] Jaap C. Schouten,[1] Floris Takens,[2] and Cor M. van den Bleek[1]
[1]*Department of Chemical Process Technology, Delft University of Technology, Julianalaan 136, 2628 BL Delft, The Netherlands*
[2]*Department of Mathematics, University of Groningen, P.O. Box 800, 9700 AV Groningen, The Netherlands*

(Received 9 May 1996)

A feedforward neural network was trained to predict the motion of an experimental, driven, and damped pendulum operating in a chaotic regime. The network learned the behavior of the pendulum from a time series of the pendulum's angle, the single measured variable. The validity of the neural network model was assessed by comparing Poincaré sections of measured and model-generated data. The model was used to find unstable periodic orbits (UPO's), up to period 7. Two selected orbits were stabilized using the semicontinuous control extension, as described by De Korte, Schouten, and van den Bleek [Phys. Rev. E **52**, 3358 (1995)], of the well-known Ott-Grebogi-Yorke chaos control scheme [Phys. Rev. Lett. **64**, 1196 (1990)]. The neural network was used as an alternative to local linear models. It has two advantages: (i) it requires much less data, and (ii) it can find many more UPO's than those found directly from the measured time series. [S1063-651X(96)09910-2]

PACS number(s): 05.45.+b, 84.35.+i

I. INTRODUCTION

In the last ten years, many physical systems that exhibit seemingly random behavior have been demonstrated to be low-dimensional chaotic. In practice, the presence of chaos is often undesirable, and insights from nonlinear dynamics have been used to account for chaos in process design. At the same time, Ott, Grebogi, and Yorke (OGY) [1] developed a control scheme that can be used to *exploit* chaotic behavior. The OGY method is based on the observation that the attractor of a chaotic system typically contains an infinite number of unstable periodic orbits (UPO's). All that is needed to change the system behavior from chaotic to periodic is to select one of these UPO's and stabilize it. Due to the sensitivity of chaotic systems for small perturbations, this can be achieved using only very small control actions. The possibility to select different kinds of periodic behavior by just selecting different UPO's makes this type of control very appealing.

Thus far, the OGY method has been applied to simple chaotic systems. To develop the chaos control methodology further, and make it applicable to more complex experimental systems, we have chosen to extend the method with a neural-network-based process model, and to first test this extension on a comprehensive experimental system, a driven and damped pendulum. For control, one needs (i) a method to find UPO's, and (ii) a model that can predict future states of the system from the present, measured state. Hübinger *et al.* [2] controlled a pendulum of which all the state variables were measured. They used the equations of motion to calculate UPO's and to make predictions, and they developed a semicontinuous control (SCC) extension of the OGY method to cope with the large unstable eigenvalues of the stabilized UPOs. De Korte, Schouten, and van den Bleek [3] controlled a different, less ideal pendulum of which only one state variable, its angle, was measured. Delayed values of the angle were used to obtain a complete representation of the state. UPO's were found by searching the measured data for close returning points, and the predictions were made by local linear models, fitted directly to the measured data.

A problem that we envisage when the control strategy is applied to more complex dynamical systems, i.e., with higher dimension D, is that the time needed to collect measurements for fitting local linear models becomes excessively long. This is because local models rely on the availability of measured data in the neighborhood of a selected UPO. The probability that the system visits this UPO within a certain distance decreases exponentially with D. The problem can be circumvented by using a global, nonlinear model, which uses *all* available data to fit its parameters. A second advantage of such a model is that it can be used to screen the system's attractor for periodic orbits. This is very important, because the more UPO's are found, the more different kinds of system behavior can be stabilized, and the more likely it is that the chaos control is useful in practice. Without a global model, one will only be able to find those UPO's that are visited at least several times during the measurement period.

For simple, well-defined systems, a basis for a global model can be obtained by careful analysis of the system and its principles. In practice, this is often not possible, and as an alternative, a general model structure can be adopted that ''learns'' its parameters from measured behavior. A common type of such a self-learning model is the multilayer feedforward neural network. We have applied this kind of modeling to the pendulum in order to further generalize the chaos control methodology. In the future we wish to control the chaotic hydrodynamics of a gas-solids fluidized bed reactor [4] to enhance chemical conversion and selectivity.

In this paper, we present the results of a neural-network-based model for the driven pendulum. First, the problem of modeling a chaotic system is defined and a description is given of the type of neural network we used (Sec. II). The pendulum and the measurements are described (Sec. III). Then it is shown how we selected inputs and outputs of the model (Sec. IV). The network model is trained (Sec. V),

*Electronic address: r.bakker@stm.tudelft.nl

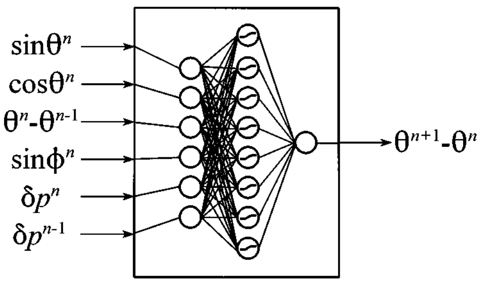

FIG. 1. Schematic representation of the MLP model for the chaotic pendulum with inputs and outputs taken from Eq. (13). Shown is an MLP network, having six input nodes, eight nonlinear nodes in a single hidden layer, and one output node. Linear nodes are represented by empty circles, sigmoidal nodes by circles filled with an *s* curve.

validated (Sec. VI), and then used to find UPO's (Sec. VII). Finally, it is applied to stabilize two different UPO's of the driven pendulum (Sec. VIII).

II. NEURAL NETWORK MODELING

The evolution of a deterministic chaotic system in discrete time is given by the map

$$\mathbf{Z}^{n+1}=\mathbf{F}(\mathbf{Z}^n), \tag{1}$$

where $\mathbf{Z}^n$ is the state of the system at time $t_n=t+n\Delta t$. The modeling of an experimental chaotic system consists of two steps: (i) find a representation of the state $\mathbf{Z}$ in terms of variables that can be measured; and (ii) approximate the nonlinear function $\mathbf{F}$, using information from measured time series only. For the first step, in principle only the measurement of a single variable is needed [5]. The system's state can be represented by a sequence of delayed values of this variable. This procedure, known as delay coordinate embedding, is outlined in Sec. III for the special case of the pendulum. For the second step, an appropriate nonlinear basis function $\mathbf{f}_\mathbf{w}(\mathbf{Z})$ must be chosen, where the parameters $\mathbf{w}$ can be varied to adjust the shape of $\mathbf{f}_\mathbf{w}$ in order to minimize the difference between $\mathbf{F}$ and $\mathbf{f}_\mathbf{w}$. In a large number of studies, e.g., [6–9], multilayer feedforward neural networks (FFN's) have been used to serve as a model structure $\mathbf{f}_\mathbf{w}$ for chaotic dynamical systems. FFN's generate very smooth functions that are not easily disturbed by noise. Other options for $\mathbf{f}_\mathbf{w}$ include polynomial NARMAX models [10] and radial basis functions [11,12]. Our choice of FFN's is based on the advantages of FFN's over polynomial models, in that their output is bounded so there is no danger that the model will be unstable. Radial basis functions suffer from the same problem as local linear models; they both rely on the availability of local information.

The structure of the most common FFN, the multilayer perceptron (MLP) [13], is as follows. The network consists of nodes that are arranged in layers. As a convention, the notation $\mathrm{MLP}_{I,H1,H2,\ldots,O}$ denotes a network with I inputs, $H1$ nodes in the first hidden layer, $H2$ in the second, etc., and O outputs. A schematic representation of, as an example, a $\mathrm{MLP}_{6,8,1}$ network is shown in Fig. 1. Each node in the secondand subsequent layers is assigned an activity a which is a linear combination of the outputs of the nodes in the previous layer:

$$a_q^l=\sum_{p=1}^{P}(w_{pq}^l o_p^{l-1})+b_q^l, \tag{2}$$

where a_q^l is the activity of the qth node in layer l, o_p^{l-1} is the output of the pth node in layer $l-1$, P is the number of nodes in layer $l-1$, and, w_{pq}^l and b_q^l are adjustable parameters of the node called weights and biases, respectively. The activity of nodes in the first layer is equal to their (single) input. The output of a node is calculated from

$$o_q^l=h(a_q^l), \tag{3}$$

where h is a transfer function. In the input and output layers, h is linear while in the hidden layers it is sigmoidal, i.e., bounded and monotonically increasing. As a sigmoidal function, we used a—fast to compute—spline approximation of the hyperbolic tangent

$$h(a)=\begin{cases} -1, & a<-2 \\ \frac{1}{4}a^2+a, & -2\leq a<0 \\ -\frac{1}{4}a^2+a, & 0\leq a\leq 2 \\ 1, & a>2. \end{cases} \tag{4}$$

If the size of a neural network is not sufficient to approximate the objective function, it can be made more flexible by adding more nodes and/or layers to the network. It has been shown, on the one hand, that a network with a single hidden layer that contains an infinite number of sigmoidal nodes is sufficient to approximate any nonlinear function [14]. On the other hand, our experience has shown that a second hidden layer in some cases greatly reduces the total required number of nodes and makes the neural network approximation more smooth.

Training an MLP is adjusting its parameters $\mathbf{w}$ in order to minimize the difference between $\mathbf{F}(\mathbf{Z})$ and $\mathbf{f}_\mathbf{w}(\mathbf{Z})$ for a given set of N measurements. Taking the quadratic norm, the error function becomes

$$E(\mathbf{w})=\sum_{n=1}^{N}\|\mathbf{f}_\mathbf{w}(\mathbf{Z}^n)-\mathbf{F}(\mathbf{Z}^n)\|^2. \tag{5}$$

As recommended in [15], we have used the conjugate gradient method to solve the nonlinear minimization problem. The derivatives $\partial E/\partial w$ are calculated analytically using the backpropagation algorithm [13]. Initially, the weights are chosen randomly and small (<0.1). This makes that the output of the MLP is initially zero for any input. During training, the weights increase, and the MLP gradually approximates $\mathbf{F}$. Training is stopped when the error does not decrease more than 5% in 500 conjugate gradient iterations or when the prediction error on an independent test data set starts to increase. Typically, this requires about 2000 iterations.

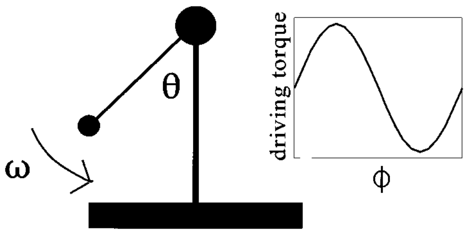

FIG. 2. Schematic drawing of the pendulum and its driving torque signal. The pendulum arm can rotate around its axis, the angle θ is measured. During the experiments, the frequency of the driving torque was 0.85 Hz.

III. PENDULUM MODEL

In this section, we describe how a suitable model structure for our experimental pendulum was obtained. To obtain an idea about the complexity of the system, we first look at the equations of motion of an ideal driven and damped pendulum. These equations are in dimensionless form,

$$\frac{d}{dt}\begin{pmatrix}\theta\\ \omega\\ \phi\end{pmatrix}=\begin{pmatrix}\omega\\ -\gamma\omega-\sin\,\theta+\alpha\,\sin\phi+\delta p\\ \omega_D\end{pmatrix}, \tag{6}$$

where θ is the angle of the pendulum, ω its angular velocity, γ is a damping constant, δp is the additional driving torque that can be used to perturb the system, and (α,ω_D,ϕ) are the amplitude, frequency, and phase, respectively, of the harmonic driving torque. In our experimental setup (Sec. IV) we can only measure the angle θ. In addition, the phase ϕ of the applied driving torque and the additional driving torque δp are known. The most straightforward application of delay coordinate embedding would be to adopt the following neural network prediction model:

$$\theta^{n+1}=\mathbf{f_w}(\theta^n,\theta^{n-1},\dots,\theta^{n-d+1},\phi^n,\delta p^n,\delta p^{n-1},\dots,\delta p^{n-r+1}), \tag{7}$$

where $\mathbf{f_w}$ is an MLP network, n is the discrete time index, d is the number of delay coordinates used, and r is the number of (past) control actions. According to Dressler and Nitsche [16], r must be taken equal to d. Unfortunately, in Eq. (7) the inputs θ and ϕ are angles that are only defined in the interval $[0,2\pi]$. It is desirable that the neural network treats an angle of almost zero the same as an angle of almost 2π. Therefore, as an alternative to Eq. (7) we use

$$\theta^{n+1}=\theta^n+\mathbf{f_w}(\sin\theta^n,\cos\theta^n,\Delta\theta^n,\dots,\Delta\theta^{n-d+2},\sin\phi^n,\cos\phi^n,\delta p^n,\delta p^{n-1},\dots,\delta p^{n-d+1}), \tag{8}$$

where θ^n is replaced by $\sin\theta^n$ and $\cos\theta^n$, ϕ^n by $\sin\phi^n$ and $\cos\phi^n$, and where the ith delay coordinate θ^{n-i+1} is replaced by $\Delta\theta^{n-i+2}=\theta^{n-i+2}-\theta^{n-i+1}$.

To understand why this specific delay coordinate embedding works, it is interesting to see what kind of model is obtained if we derive a discrete time model from the equations of motion [Eq. (6)]. These equations suggest to use θ, ω, and ϕ as state variables. Since we can only measure the angle θ we use a delay coordinate of θ instead of ω. The state vector [Eq. (1)] then becomes

$$\mathbf{Z}^n=(\theta^n,\theta^{n-1},\phi^n)^T. \tag{9}$$

The discrete time n is related to real time t by $t_n=t_0+n\Delta t$. If Δt is small, then an approximation of the function $\mathbf{F}$ in Eq. (1) can be obtained using an Euler approximation of the set of differential equations (6). After substitution of ω^n by $\theta^n-\theta^{n-1}$, this yields the following model structure:

$$\Delta\begin{pmatrix}\theta^{n+1}\\ \theta^n\\ \phi^{n+1}\end{pmatrix}\approx\begin{pmatrix}(1-\gamma)\Delta\theta^n-\sin\theta^n+a\,\sin\,\phi^n+\delta p^n\\ \Delta\theta^n\\ \omega_D\Delta t\end{pmatrix}. \tag{10}$$

It can be seen from Eq. (10) that the prediction of θ^n is trivial, and that the prediction of ϕ^{n+1} is obtained by definition. Therefore, errors will arise only in the prediction of θ^{n+1}, and they will be due to (i) the Euler approximation, (ii) the delay coordinate approximation of ω, and (iii) deviations from ideal behavior of the pendulum. The prediction of θ^{n+1} in Eq. (10) can be written as

$$\theta^{n+1}=\theta^n+F(\sin\theta^n,\sin\phi^n,\Delta\theta^n,\delta p^n). \tag{11}$$

We see that the model inputs of Eq. (11) are a subset of the model inputs of Eq. (8).

IV. MEASUREMENTS

The pendulum we use is a type EM-50 pendulum produced by Daedalon Corporation; see [17] for details and Fig. 2 for a schematic drawing. We have observed [3] that the motion of the pendulum deviates from the ideal behavior expressed in Eq. (6), because of the four electromagnetic driving coils that repulse the pendulum at certain positions (up, down, left, and right). Two different time series were measured from the pendulum, one for the unperturbed system and one for the perturbed system. The first is used to construct Poincaré plots, the second for training the neural network model. The sampling interval Δt was taken $\frac{1}{32}$ of a driving cycle,

$$\Delta t=\frac{2\pi}{\omega_D}\frac{1}{32}. \tag{12}$$

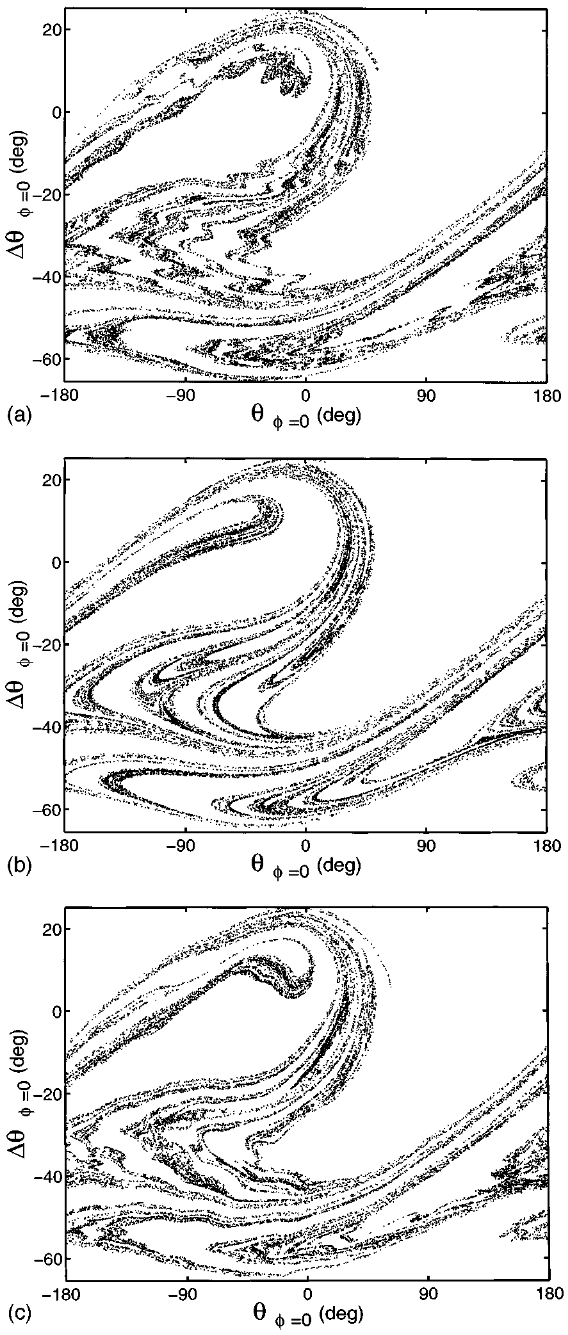

FIG. 3. Poincaré plots of measured time series (a) and time series generated by the linear-in-the-parameter (LP) model (b) and the MLP$_{6,16,1}$ model (c). Each plot shows 20 000 points $(\theta,\Delta\theta)_{\phi=0°}$, where $\Delta\theta$ is defined by $\theta_{\phi=0°}-\theta_{\phi=-11°}$. The plots clearly show the fractal geometry of both the real system and the two models.

The perturbed system had an external driving force, i.e., the control action, changed randomly every fourth time step, with a uniform distribution between plus and minus 5% of the driving torque amplitude. The perturbations were stored together with the measured time series. The driving frequency was 0.85 Hz, and the pendulum was operated in a chaotic regime, as can be seen from the fractal geometry of the measured Poincaré plot [Fig. 3(a)].

V. NETWORK TRAINING AND VALIDATION

Here we present the results of the neural network model that we selected to give the best performance. It uses one delay coordinate ($d=2$) and does not use the proposed input $\cos\phi$. The network has one hidden layer containing 16 nodes. It is shown schematically in Fig. 1, and using the notation from Sec. II it can be written

$$\theta^{n+1}=\theta^n+\mathbf{f}_{\mathrm{w}}(\sin\theta^n,\cos\theta^n,\theta^n-\theta^{n-1},\sin\phi^n,\delta p^n,\delta p^{n-1}), \tag{13}$$

with $\mathbf{f}_{\mathrm{w}}$ an MLP$_{6,16,1}$. Apart from this network, we trained networks with an extra delay coordinate, with $\cos\phi$ as an extra input, with no hidden layer at all, and with half and with twice the number of nodes in the hidden layer. None of these networks gave smaller prediction errors. The networks were trained using the first 5000 points of a time series of 15 000 data points. The last 5000 points were used for testing. As mentioned in Sec. II, the networks are initialized with small random weights. It is found that training the neural networks more than once with different random initial weights yields approximately the same prediction error.

The most straightforward way to compare the accuracy of different models is to look at their one-step-ahead prediction errors on an independent test data set. Table I shows the root-mean-squared errors (RMSE's) of the MLP$_{6,16,1}$ and, for comparison, of the model with no hidden layer. For the latter model, Eq. (13) becomes

$$\theta^{n+1}=\theta^n+\mathbf{w}(\sin\theta^n,\cos\theta^n,\theta^n-\theta^{n-1},\sin\phi^n,\delta p^n,\delta p^{n-1})^T+b, \tag{14}$$

and because this model is linear in its parameters we refer to it as the LP model. The RMSE of the neural network model is twice as low as that of the LP model, which is a considerable improvement. Note that although the number of 129 adjustable parameters of the MLP$_{6,16,1}$ model is large compared to the seven parameters of the LP model, the ratio between the number of parameters and amount of data of 129/5000 is still low, and from the error on the independent test data set it can be seen that no overfitting has occurred. The measurement of the angle of the pendulum is done with 12-bit accuracy, corresponding to a maximum error of about 0.1°. The neural network RMSE is about 3.5 times higher than this value, either due to inadequacy of the model or due to the unpredictability of the chaotic system. This unpredictability can be assessed by calculating the sum of positive Lyapunov exponents of the system, which gives a measure of the loss of information per unit time. For the pendulum,

TABLE I. Comparison of prediction error for neural network and LP model. Also included is the largest Lyapunov exponent λ of each model.

	MLP$_{:6,16,1}$	LP model
RMSE training data	0.35°	0.63°
RMSE test data	0.36°	0.62°
λ (bits/period)	2.5	2.2

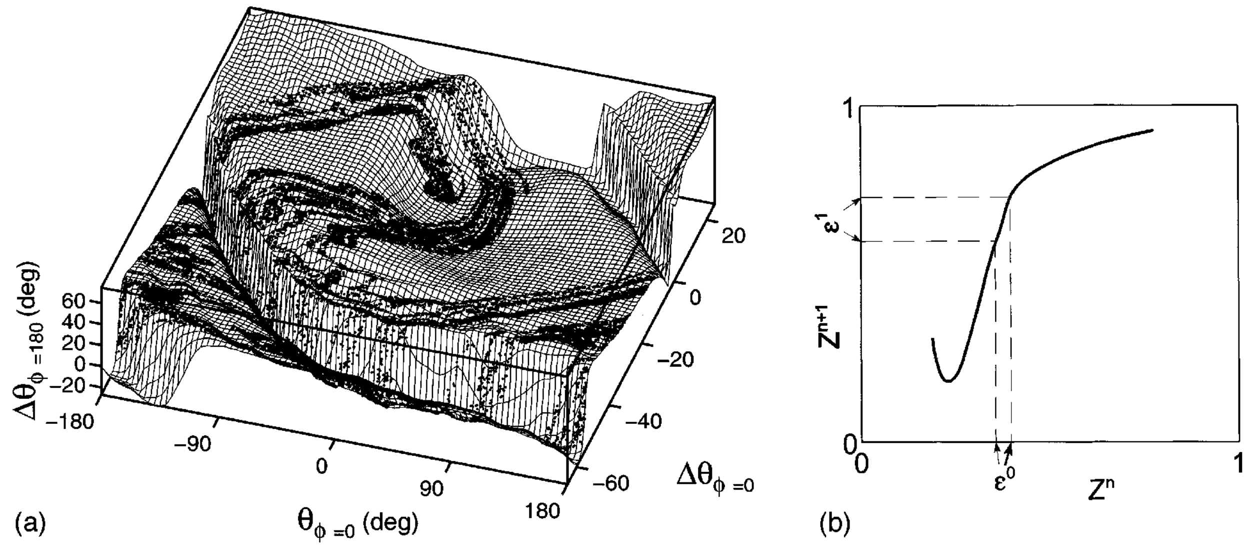

FIG. 4. Prediction surface generated by the neural network model (a). The surface shows how to predict $\Delta\theta_{\phi=180°}$ from a given state $(\theta,\Delta\theta)_{\phi=0°}$. The Poincaré plot of the measured time series, as shown in Fig. 3(a), is superimposed on the surface. The surface shows regions where the motion of the pendulum can be well predicted (flat surface) and regions where much information is lost (steep surface). In (b), the link between loss of information and steepness of the prediction surface is explained. The steeper the prediction surface, the larger the error (ε^1) in the prediction of the state Z^{n+1} will be compared to the error ε^0 in the estimation of the current state Z^n.

this sum equals the largest Lyapunov exponent since this is the only positive one. This follows from the following two observations.

(i) One of the three Lyapunov exponents is zero [18], corresponding to the fact that the phase of the driving force can be predicted without information loss.

(ii) The sum of all exponents must be negative, as the system is dissipative.

The procedure outlined by Wolf *et al.* [19] was used to compute the largest Lyapunov exponent λ of both models, calculated values are shown in Table I. The Lyapunov exponent of 2.5 bits/period for the MLP model implies that if we know the initial conditions with 12-bit accuracy, we can predict the state after one time step ($\frac{1}{32}$ period) with $12-2.5/32=11.9$ bits precision, on the average. The 0.1 bit that we have lost is far less than the RMSE of the model; therefore we *cannot* subscribe a substantial part of the model error to the unpredictability of the system. It must be either the representation of the state **Z** or the error in the approximation of the system function **F** that causes the model error.

For a three-dimensional chaotic system, a very convenient way to validate a model for the system is to compare its Poincaré plot to that of the measured data. For the experimental pendulum, the Poincaré plot is obtained by plotting points $(\theta,\Delta\theta)$ sampled at a constant value of the driving phase ϕ, $\phi=(2\pi/\omega_D)t \bmod 2\pi$, in a two-dimensional space. To calculate a Poincaré section of a prediction model, a time series generated by the model is needed. This is done by initializing the model with an arbitrary state, predict 2000 driving cycles (each driving cycle consisting of 32 time steps) ahead to let transients die out, and then predict 20000 driving cycles ahead, taking a sample each time the driving phase is zero. Poincaré plots of measured, LP model, and MLP generated data are shown in Figs. 3(a)–3(c). Note that due to the use of the nonlinear terms $\sin\theta^n$ and $\cos\theta^n$ as model inputs it is here possible to represent a chaotic system by a network without a nonlinear hidden layer. The LP model does capture the overall shape of the attractor, but the wrinkles occurring at length scales of about 10° are not captured by the model. The MLP model more closely resembles

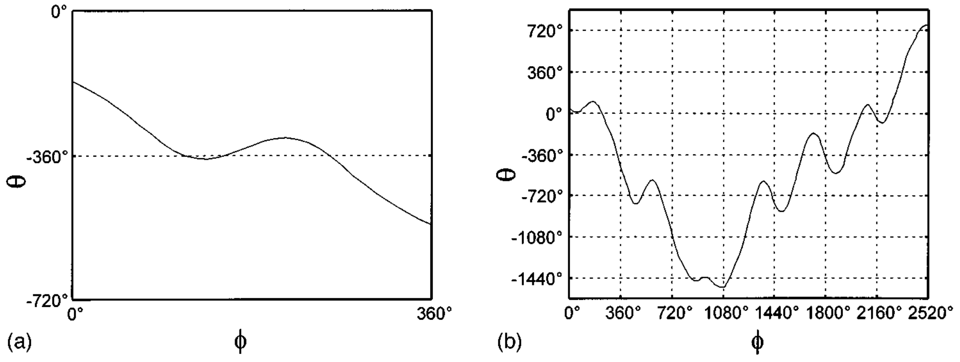

FIG. 5. Unstable periodic orbits of period 1 (a) and period 7 (b), as calculated from the neural network model. Here the angle θ is allowed to move outside the interval 0–360° to avoid having discontinuities in the plots. The definition of an UPO requires equal values of θ mod 360° and equal values of $\Delta\theta$ (slope of the curve) at the beginning and end of the orbits.

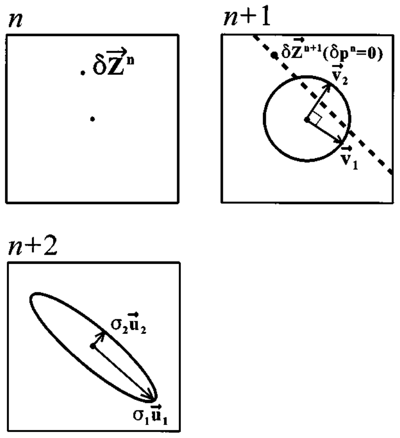

FIG. 6. Explanation of the SCC control algorithm. The figure shows three successive maps. The dot at the center of the maps indicates its fixed point, the dot labeled δZ^n the deviation from the fixed point. At time n the system can be steered to any point on the dashed line through δZ^{n+1} $(\delta p^n=0)$ with direction $\mathbf{D}_{\delta p^n}F(Z,\delta p^n,\delta p^{n-1})$. Now, as a control criterion, the choice is made to steer to the axis ν_2 (obtained with singular value decomposition) that corresponds to the maximum shrinking axis $\sigma_2 u_2$ at the next transition map at time $n+2$ (see Ref. [3]).

the measured attractor, but one can see that there is still room for improvement. We have chosen not to further optimize the model but to test whether it is satisfactory in its use for chaos control.

VI. PREDICTION SURFACE

Each point on the two-dimensional (2D) Poincaré plots shown in Figs. 3(a)–3(c), represents a state $(\theta,\Delta\theta,\phi)$ of the system, with $\phi=0$. We can extend the plot to a 3D picture and use the extra dimension to plot the prediction of one of the state variables. The prediction surface we then get shows the nonlinearity of the system. In Fig. 4(a) we have plotted the prediction surface of $\Delta\theta$, with a prediction time of half a driving period (16 time steps). The local steepness of the surface has a direct relation to the information loss. This is illustrated in Fig. 4(b), where it is shown how an initial error ε^0 growths to an error ε^1 if the steepness of the prediction curve is larger than 1. Since we are dealing with a chaotic system the average error growth will be positive, corresponding to a positive largest Lyapunov exponent. The prediction surface can be compared with Fig. 2 in [2], which shows the local Lyapunov exponent as a function of position on the Poincaré plot of an ideal pendulum [Eq. (6)].

VII. SEARCH FOR UNSTABLE PERIODIC ORBITS

For a chaotic system with a periodic driving signal, the duration of a periodic orbit must be an integer multiple of the period of the signal (because the phase of the driving signal is included in the state of the system). So, when the system is following an UPO, the state at discrete time n will be equal to the state at time $n+pT$, with p the (integer) period of the UPO and T the (integer) number of samples per driving period. The search for UPO's is thus to find a state $\mathbf{Z}^n$ that obeys

$$\mathbf{Z}^{n+pT}=\mathbf{Z}^n, \tag{15}$$

where $\mathbf{Z}^{n+pT}$ is calculated from $\mathbf{Z}^n$, applying

$$\mathbf{Z}^{n+1}=\mathbf{F}(\mathbf{Z}^n) \tag{16}$$

pT times. To find an UPO of preselected period p, an arbitrary initial state $\mathbf{Z}^n$ is chosen, and then adjusted until Eq. (15) is satisfied. Adjustments are calculated by a routine that minimizes the error

$$E=\|\mathbf{Z}^{n+pT}-\mathbf{Z}^n\|^2. \tag{17}$$

An UPO is found when the error decreases to zero. A different initial state must be tried when the error does not reach zero. For the minimization we used the conjugate gradient algorithm [20]. Since we use a neural network to approximate $\mathbf{F}$, derivatives $\partial E/\partial Z^n$ can be calculated with backpropagation through time [21]. The algorithm makes it possible to search systematically for UPO's of a preselected period, just by trying many different initial states. In Fig. 5 we show a typical period-1 UPO (a) and a period-7 UPO (b). Section VIII describes how these UPO's were stabilized using the SCC method.

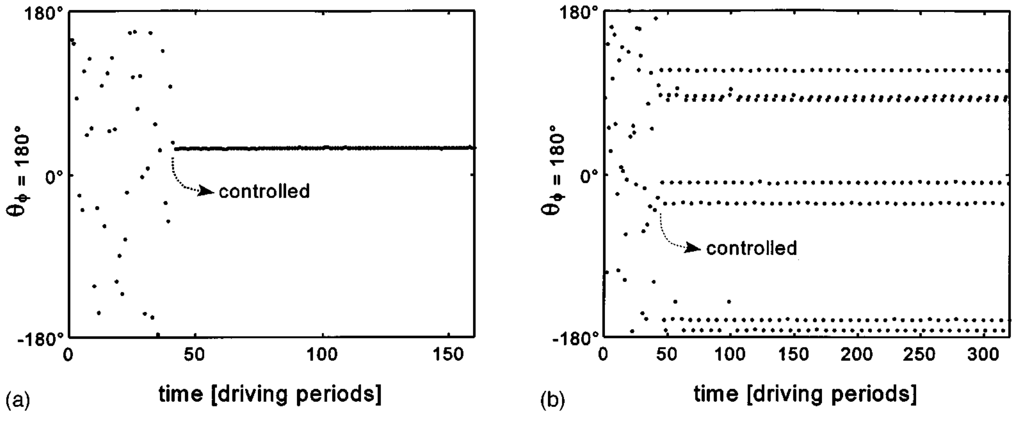

FIG. 7. The effect of control is shown by plotting the angle θ sampled once each driving cycle. For the period-1 UPO (a), it takes a while before the system is close enough to the UPO to enable control. From then on, the angle will have the same value each driving cycle. For the period-7 UPO (b), the angle is the same each seventh driving cycle.

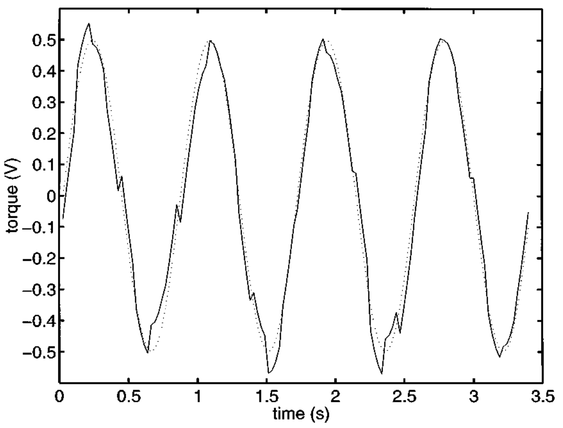

FIG. 8. The driving force of the pendulum (voltage sent out by control computer), with superimposed on it the control action during stabilization of the period-7 UPO shown in Fig. 5(b). The variance of the control action is less than 60 times that of the driving force.

VIII. APPLYING CHAOS CONTROL

In the papers by Hübinger *et al.* [2] and De Korte, Schouten, and van den Bleek [3], it is described how chaos control of the pendulum can be achieved using a semicontinuous control (SCC) extension of the well-known OGY control scheme. The algorithm changes the control parameter several times per periodic orbit and uses singular value decomposition to calculate a maximum shrinking axis; see Fig. 6 for a brief explanation. Hübinger *et al.* used the equations of motion of the pendulum as a prediction model, whereas De Korte, Schouten, and van den Bleek used local linear models, based on delay coordinates, that were fitted directly to measured data in the neighborhood of a selected UPO. A global neural network model provides a third way to create local linear models. A local linear model is equal to the local gradient, or Jacobian, of the global model. For a neural network, this gradient is easily obtained with back-propagation or, if the local linear model has to predict more than one sampling time ahead, with back-propagation through time. The advantage of this approach is that it does not require *a priori* knowledge of the system, and that it does not require that the system has visited the selected UPO during the measurement period.

The control experiments differ from those of Ref. [3] in how the UPO's are found and how local linear models are obtained. The reader is referred to this paper for a detailed explanation of the control algorithm and the chaos control experiments. Here we confine ourselves to show the results in Figs. 7 and 8. Figure 7 is an illustration of the stabilization of the period-1 and period-7 UPO's from Fig. 5. Figure 8 shows to what extent the driving force of the pendulum was disturbed by the control action. The variance of the control action is 60 times lower than the variance of the driving force.

IX. CONCLUDING REMARKS

A multilayer feedforward neural network has been developed that can accurately predict the motion of a chaotic pendulum. The network is trained on a time series of a single measured variable; additional information about the state of the system is obtained from delayed values of this variable, and from the phase of the external driving force.

Poincaré sections of measured data and model generated data show that the neural network model has captured the system's dynamics well. The prediction surface that can be computed from the neural network model gives a comprehensive insight into the nonlinearity and local predictability of the pendulum.

The neural network was used to find unstable periodic orbits of the system, and these orbits were stabilized with the SCC method. This way, two very different kinds of periodic behaviors were stabilized, applying only very small control actions. The advantages of the neural network approach over the use of local linear models is that (i) they offer the possibility to screen the system's attractor for UPO's, and (ii) they make use of all the available data.

The results obtained in this study of a simple but real and nonideal experimental system show that, in order to achieve chaos control, it is not necessary to measure more than one of the system's state variables and that the equations of motion of the system need not be known. We believe that the use of self-learning models such as neural networks makes OGY control applicable to a wide variety of chaotic systems. In future work, we intend to show that the combination of self-learning models such as neural networks and SCC control is useful in controlling and stabilizing higher-dimensional, experimental systems. As a next step, we will build a neural network model that does not use any knowledge about the external driving force applied to the pendulum, but uses extra delay coordinates of the measured angle instead. Then we will extend the method to spatiotemporal chaotic systems, for which it may be necessary to use multiple measurements of the same variable at different locations in the system. The final aim of the work is to experimentally control the chaotic hydrodynamics of a gas-solids fluidized bed reactor [4] to enhance chemical conversion and selectivity.

ACKNOWLEDGMENTS

The authors are grateful to R. J. de Korte for carrying out the experiments. The investigations were supported by the Netherlands Foundation for Chemical Research (SON) with financial aid from the Netherlands Organizations for Scientific Research (NWO).

[1] E. Ott, C. Grebogi, and J. A. Yorke, Phys. Rev. Lett. **64**, 1196 (1990).

[2] B. Hübinger, R. Doerner, H. Heng, and W. Martienssen, Int. J. Bifurc. Chaos **4**, 773 (1994).

[3] R. J. De Korte, J. C. Schouten, and C. M. Van den Bleek, Phys. Rev. E **52**, 3358 (1995).

[4] C. M. Van den Bleek and J. C. Schouten, Chem. Eng. J. **53**, 75 (1993).

[5] F. Takens, Lecture Notes Math. **898**, 366 (1981).

[6] J. L. Hudson, M. Kube, R. A. Adomatis, I. G. Kevrekidis, A. S. Lapedes, and R. M. Farber, Chem. Eng. Sci. **45**, 2075 (1990).

[7] J. B. Elsner, J. Phys A **25**, 843 (1992).

[8] J. C. Principe, A. Rathie, and Jyh-M. Kuo, Int. J. Bifurc. Chaos **2**, 989 (1992).

[9] V. Cimagalli, S. Jankowski, M. Giona, and T. Calascibetta, in *Neural Network Reconstruction and Prediction of Chaotic Dynamics, Proceedings of the 1993 IEEE InternationalSymposium on Circuits and Systems, Chicago* (IEEE, Piscataway, NJ, 1993), pp. 2176–2179.

[10] L. A. Aguirre and S. A. Billings, Int. J. Bifurc. Chaos **5**, 449 (1995).

[11] H. Leung and T. Lo, IEEE J. Ocean. Eng. **18**, 287 (1993).

[12] S. Chen, F. N. Cowan, and P. M. Grant, IEEE Trans. Neural Networks **2**, 302 (1991).

[13] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing* (MIT Press, Cambridge, MA, 1986), Vol. 1.

[14] G. Cybenko, Math. Controls, Signals Syst. **2**, 303 (1988).

[15] J. Leonard and M. A. Kramer, Comput. Chem. Eng. **14**, 337 (1990).

[16] U. Dressler and G. Nitsche, Phys. Rev. Lett. **68**, 1 (1992).

[17] J. A. Blackburn, S. Vik, and B. Wu, Rev. Sci. Instrum. **60**, 422 (1989).

[18] H. Haken, Phys. Lett. **94A**, 71 (1983).

[19] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, Physica D **16**, 285 (1985).

[20] R. Fletcher and C. M. Reeves, Comput. J. **7**, 149 (1988).

[21] P. J. Werbos, Ph.D. thesis, Harvard University, 1974.