NEURAL NETWORK BASED MONITORING AND CONTROL OF FLUIDIZED BED

FINAL REPORT

Start Date: October 1,1994 End Date December 31 1996

SUBMITTED TO

U.S. DEPARTMENT OF ENERGY
DOCUMENT CONTROL DIVISION,
FEDERAL ENERGY TECHNOLOGY CENTER (FETC)
WALLACE ROAD, BUILDING 921
P.O. BOX 10940, MS 921-118
PITTSBURG, PA 15236-0940

BY

MOHAMMAD BODRUZZAMAN, Ph.D.
PRINCIPAL INVESTIGATOR

MAGDI A. ESSAWY, Ph.D.
RESEARCH ASSOCIATE

APRIL, 1996

GRANT NUMBER: DE-FG22-94MT94015

TENNESSEE STATE UNIVERSITY
COLLEGE OF ENGINEERING AND TECHNOLOGY
3500 JOHN A. MERRIT BLVD.
NASHVILLE, TN 37209-1561

DISCLAIMER

Abstract

The goal of this project was to develop chaos analysis and neural network-based modeling techniques and apply them to the pressure-drop data obtained from the Fluid Bed Combustion (FBC) system (a small scale prototype model) located at the Federal Energy Technology Center (FETC)-Morgantown. The second goal was to develop neural network-based chaos control techniques and provide a suggestive prototype for possible real-time application to the FBC system. The experimental pressure data were collected from a cold FBC experimental set-up at the Morgantown Center. We have performed several analysis on these data in order to unveil their dynamical and chaotic characteristics. The phase-space attractors were constructed from the one dimensional time series data, using the time-delay embedding method, for both normal and abnormal conditions. Several identifying parameters were also computed from these attractors such as the correlation dimension, the Kolmogorov entropy, and the Lyapunov exponents. These chaotic attractor parameters can be used to discriminate between the normal and abnormal operating conditions of the FBC system. It was found that, the abnormal data has higher correlation dimension, larger Kolmogorov entropy and larger positive Lyapunov exponents as compared to the normal data. Chaotic system control using neural network based techniques were also investigated and compared to conventional chaotic system control techniques. Both types of chaotic system control techniques were applied to some typical chaotic systems such as the logistic, the Henon, and the Lorenz systems. A prototype model for real-time implementation of these techniques has been suggested to control the FBC system. These models can be implemented for real-time control in a next phase of the project after obtaining further measurements from the experimental model. After testing the control algorithms developed for the FBC model, the next step is to implement them on hardware and link them to the experimental system. In this report, the hardware implementation issues of the control algorithms are also discussed.

TABLE OF CONTENTS

LIST OF FIGURES

1. Executive Summary

This research report describes the work completed under the DoE Contract # DE-FG22-94MT94015 titled “Neural Network-Based Monitoring and Control of Fluidized Bed” for the period from October 1, 1994 to December 31, 1996. The first goal of this project was to develop chaos analysis and neural network-based modeling techniques and apply them on the pressure-drop data. The second goal was to develop a neural network-based chaos control techniques and provide a suggestive prototype for possible on-line or real-time application to the Fluid Bed Combustion (FBC) system (a small scale prototype model) located at the Federal Energy Technology Center (FETC) at Morgantown.

The pressure data were collected from a cold FBC experimental mock up, at the Morgantown Energy Technology Center (METC).. We have performed several analysis on these data in order to unveil their dynamical and chaotic characteristics. We have constructed the system attractor from the one dimensional time series measurements available for both normal and abnormal conditions, using the time delay embedding technique. Several identifying parameters were computed for the constructed attractors, such as the correlation dimension, the Kolmogorov entropy, and the Lyapunov exponents. These chaotic attractor parameters can be used to discriminate between the normal and abnormal operating conditions of the FBC system. It was found that, the abnormal data has higher correlation dimension, larger Kolmogorov entropy and larger positive Lyapunov exponents as compared to normal data.

Chaotic system control using neural network based techniques were investigated and compared to conventional chaotic system control techniques. Both types of chaotic system control techniques were applied to some typical chaotic systems, such as the logistic map, the Henon map, and the Lorenz system. These models can be implemented for real-time control in a next phase of the project after obtaining further measurements from the experimental model. After testing the control algorithms developed for the FBC model, the next step is to implement them on hardware and link them to the experimental system. In this report, we also discuss the hardware implementation issues of the control algorithms.

Two graduate stuents and three undergraduate students were supported under this contract. The graduate students developed their Masters thesis and published their work in conferences. The undergraduate students were trained in the area of Signal Processing, Neural Networks, Fluid Bed systems etc.

We believe that the overall project goals and objectives are met and this is a successful completion of the Contract including the submission of this Final Report.

2. Introduction

In the recent years lots of attention has been given to study systems that exhibit a type of behavior known as chaotic. Those systems were confused in the past with stochastic systems. Recently, it has been found that chaotic behavior can be driven by low order deterministic phenomena.[1] Simple models for those phenomena are achievable. However, general solution to those systems can not be found, and any long term prediction can not be realized. The reason is the sensitive dependence of such systems on initial conditions. One example is the nbody system problem for which no general solution was found, and also known for its sensitive dependence on initial conditions. Chaos has been sought for many years as undesirable phenomena. It was found though that this is not always the case. On occasion, chaos is a beneficial feature as it enhances mixing and chemical reactions and provides a vigorous mechanism for transporting heat and/or mass.[2] It is also the secret behind the survival of biological systems under extremely different conditions, due to their multiple attractor nature, which enables those systems to easily switch their parameters to adapt to new environment and maximize their performance. In the past few years, several methods have been developed to control the behavior of chaotic systems.[1-6] These methods were applied to both theoretical models and actual chaotic systems in many fields. They were applied to control chemical reactions,[7] communication,[8] nonlinear oscillators, lasers,[9] diodes,[6] heat convection,[2] mechanical vibrations, myocardial tissue (biological systems),[10] magnetoelastic systems,[1, 3] and several other applications. Chaotic systems' behavior control, in case it is achievable, gives them an advantage over non-chaotic systems, due to the ability to switch the chaotic systems' performance between several different modes.[1,4] For that same reason we might sometimes wish to build chaos into a system where it is naturally absent. In this report, we will present methods to monitor and control the chaotic behavior in a Fluidized Bed Combustion (FBC) systems. These methods are based on analysis we performed on data obtained from an experimental fluidized bed combustion system at the Morgantown Energy Technology Center (METC). These data represent some normal and abnormal operating conditions of the FBC system. Previous work in this area can be reviewed in references 11-13. A special type of a recurrent neural network, called the Dynamic System Imitator (DSI),[14] will be adopted as a neuro-controller and a short term chaotic time series predictor in the proposed monitoring and control method. The DSI network was used to control the chaotic behavior in the Lorenz system. Sample results from the analysis performed on the FBC data will be presented, and an outline of the proposed method will be introduced. Also results for using a chaotic system control technique developed by Edward Ott, Grebogi, and Yorke in 1990,[1,4] known as the (OGY) technique, to control the chaotic behavior in some typical chaotic system using small perturbations will be presented.

3. Fluidized Bed Combustor (FBC) Pressure Data Collection

Morgantown Energy Technology Center has built and operated a cold flow model to emulate fluid dynamics in a Fluidized Bed Reactor (FBC). The cold flow verification test facility consists of a ten foot high jetting fluidized bed made of clear acrylic and configured as a half cylinder vessel to facilitate jet observation. An illustration of the test facility is described in Figure 1. A central nozzle, made up of concentric pipes, continuously fed solids at 0 to 8 psig pressures. Separate flow loops controlled the conveyance of solids (inner pipe), the make-up air flow (middle pipe), sparger flow (outer pipe), and six air jets on the sloping conical grid. The half round fluid bed model provided useful information to study fluidization and design issues including jet penetration, chaotic pressure fluctuations, and mass flow rates of particles in various regions of the jetting fluid bed. The fluid bed tests were conducted using cork particles to simulate the relative density of gases to scale for a high pressure coal conversion reactor. As expected, the test generated chaotic pressure fluctuations. The differential pressures were measured at two location with each location consisting of two pressure taps spaced four inches apart. The lower pair of pressure taps were placed at a height just above the nozzle and the upper pair of pressure taps were placed at a height where the jet becomes evenly distributed across the diameter of the reactor. Differential pressure data collected at the higher sensor served as the primary data for the investigation of chaos. It clearly indicated the fluidization regime of the bed supported by visual observations. Data were collected on a data acquisition card at a rate of 50 Hz.

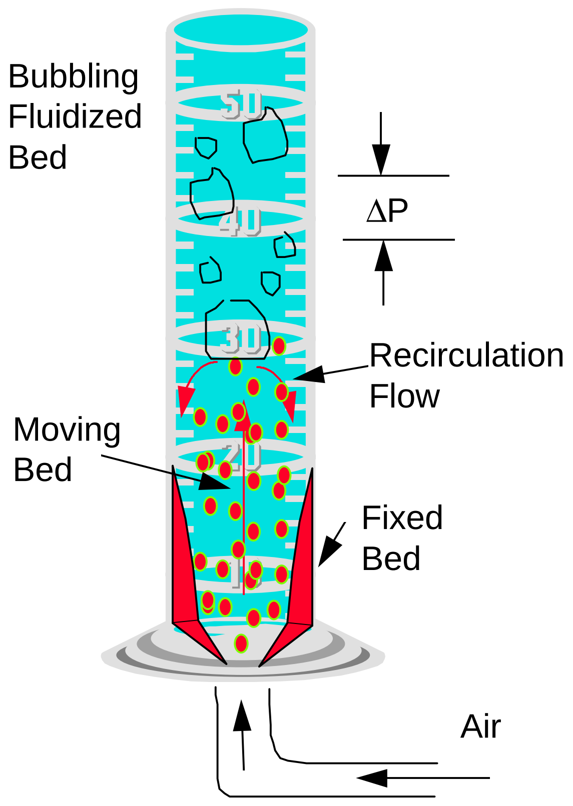


Figure 1 The general setup of the existing cold FBC experiment

4. Results and Discussion

4.1 The Chaotic Behavior in FBC Systems

Through the analysis of both the normal and abnormal FBC pressure data, it is evident that they have strange attractors, and both belong to a chaotic system. This means that the prospective control method needs to switch the system from one chaotic state to another chaotic state, which has been achieved before in simple chaotic systems such as the logistic map.[15] A two dimensional projection of the reconstructed attractor from a normal and abnormal cases are shown in Figures 2 and 3, respectively.

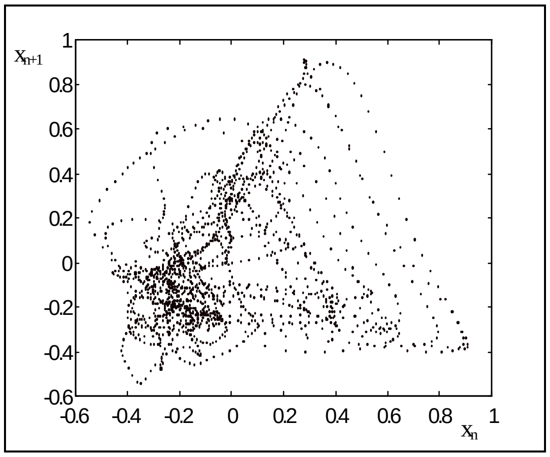


Figure 2 A plot of the FBC normal attractor projected onto two dimensional map.

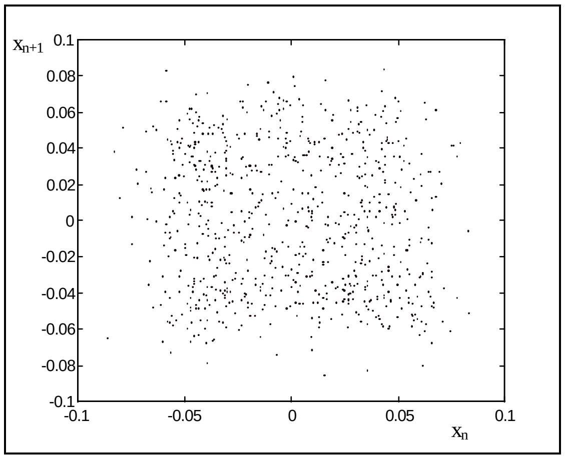


Figure 3 A plot of the FBC abnormal attractor projected onto two dimensional map.

As it appears in the two figures the normal data attractor looks much more well behaved than the abnormal data. We performed several methods to compute the parameters of both attractors. We computed the correlation dimension, the Kolmogorov entropy, and the Lyapunov exponents of the normal and abnormal attractors. The correlation integral was computed according to the following equation:[16]

$$C(N,\varepsilon) = \frac{2}{N(N-1)} \sum_{j=1}^{N} \sum_{i=j+1}^{N} \theta(\varepsilon - |x_i - x_j|), \qquad (1)$$

where $\theta(x)=1$ for $x>0$ and $\theta(x)=0$ for $x<0$. The Kolmogorov entropy was computed according to the relationship:[17]

$$K_{m,d} = \frac{1}{\tau m} \ln \frac{C_d(\varepsilon)}{C_{d+m}(\varepsilon)} \qquad (2)$$

And the Lyapunov exponents were computed according to the Sano-Swada technique to compute the Lyapunov spectrum from a chaotic time series.[18] The correlation integral graphs for the normal and abnormal attractors in a range of embedding dimensions are shown in Figures 4 and 5, respectively. The results of the chaos analysis of the FBC data are summarized in Table 1. These analysis show that both the normal and abnormal modes of the system live on a chaotic attractor, because both have fractal dimensions, positive Kolmogorov entropy, and positive Lyapunov exponents.

Table 1 Results of chaos analysis of the FBC data.

	Normal Data	Abnormal data
Correlation Dimension	3.07	17.35
Kolmogorov Entropy	8	100

However, the correlation dimension for the abnormal attractor is much higher than the normal one, which is an indication that when the system changes from its normal to its abnormal behavior, it goes from low order to high order chaos. This situation is a big challenge to any traditional control method. However, we believe that the system parameters can be adjusted through chaos control method to move the system from its high order chaotic behavior to the normal low order chaotic state. A proposed method for the FBC system monitoring and control is discussed below. Even though the chaos analysis methods described above are the best to define the condition of a chaotic system, they are not suitable for on-line monitoring because they need intensive calculations that might run several ours on digital computers. Instead, a chaotic time series predictor technique will be used.

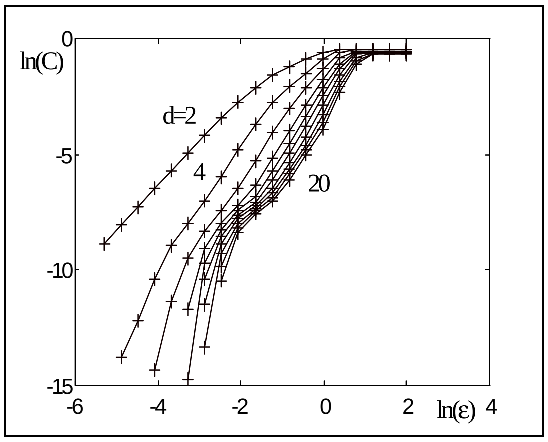


Figure 4 A plot for the correlation integral of the FBC normal attractor, for embedding dimensions 2-20.

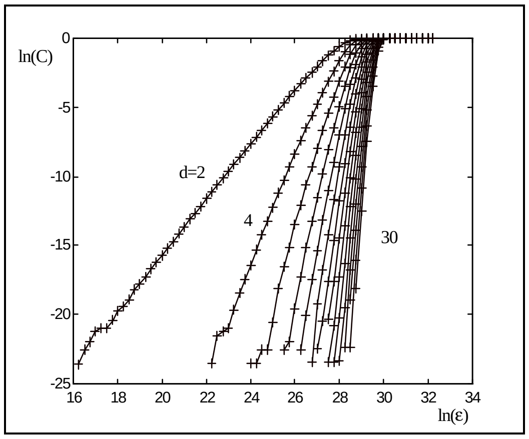


Figure 5 A plot for the correlation integral of the FBC normal attractor, for embedding dimensions 2-30.

4.2 The Proposed Monitoring and Control Method

For any control method to function properly with the prescribed system, there should be some monitoring device that will monitor the system state and switch the controller on, in case of detection of any abnormal behavior. We will monitor the system using a chaotic time series predictor that we developed using the DSI neural network. This chaotic time series predictor was able to predict the chaotic behavior of chaotic time series generated by several chaotic systems such as the logistic map, the Henon map, and the cubic map. In all cases the DSI predictor was able to predict the chaotic behavior of the time series for a short time starting from some time history of the signal. It was also able to predict the state space system attractor for the rest of the time. If we train the DSI predictor to predict the normal behavior of the system, starting from some initial measurements, then the average error between the actual and predicted time series over a certain period of time will give us an indication of how much drift did the system make from its normal condition. Once a certain threshold is violated we can switch the controller on. To be able to design a controller for such a system we need to study the effect of different system parameter on the behavior, and find which parameters are responsible for the system drift from normal. If any of these parameters would be accessible for control, we can implement a control method as illustrated in Figure 6. The only obstacle before executing and testing this method is to find an appropriate model that will describe the prescribed system behavior in all modes. This model will be used to study the effect of system parameters on the system behavior and to test the performance of the proposed control methodology. The reason we chose to use the DSI neural network in this situation is the known dynamic characteristics of such a network. The DSI is a fully recurrent neural network that was specially designed to model a wide variety of dynamic systems. It has feedback connections and integrators to form a compact representation of time lags and interactions in real systems. It is suitable for on-line applications due to its fast response and realistic interface with the outside world. It is also equipped with a multidimensional optimization technique and dynamic windowing which allows it learn system dynamics from long time series. The DSI network has been used to control the chaotic behavior in the Lorenz system, and stabilize it into either a fixed point or a periodic orbit. In that application, one state variable was fed back as an input to the DSI, and the output of the DSI was used to control the system. The error between the actual output of the system and a target behavior was used to train the DSI network. State point perturbation control and parameter perturbation control methods have been demonstrated. The details of such technique will be discussed below. We wish that a similar technique will work with the FBC system, even though we believe that the FBC system is a much more difficult problem, due to its complex behavior under both normal and abnormal conditions.

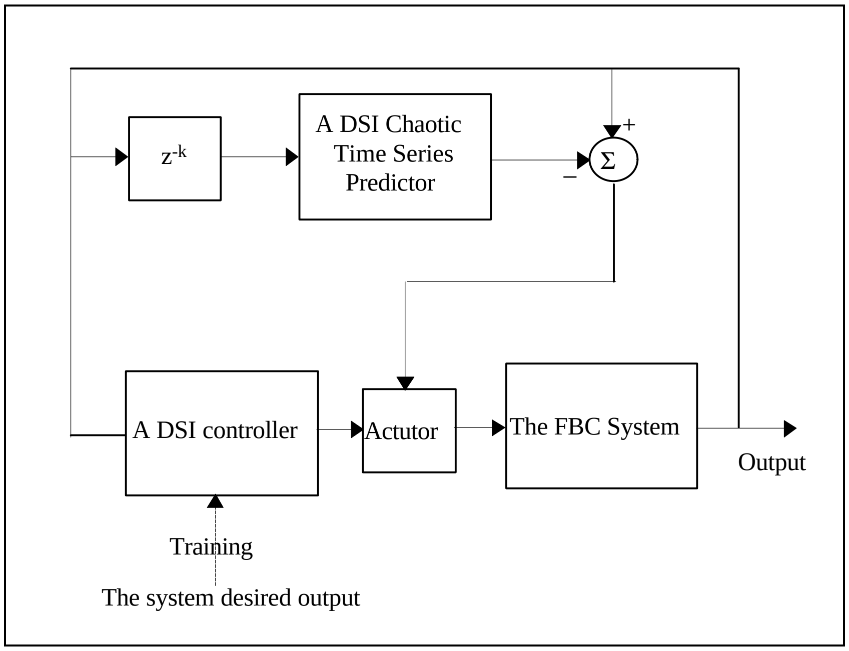


Figure 6 An illustration of the proposed control method for the chaotic FBC system.

4.2.1 The Dynamic System Imitator (DSI) neural network

The neural network used for the chaotic time series prediction in this project is a dynamic neural network called Dynamic System Imitator (DSI). The DSI is a fully recurrent neural network that is specially designed to model a wide variety of dynamic systems. [19, 20] As shown in Figure 7, the DSI has a three layer structure: input, hidden, and output layer. Connections have both weights and integrators in parallel to model short term and long term memory mechanisms that handle modeling of time behaviors and time lags in real systems. Every node in the input layer has one input, $x_k(t)$, and two outputs defined by: [19]

$$\begin{aligned} o_{1j}^{i}(t) &= x_j(t)\ , \\ o_{2j}^{i}(t) &= \int_0^t x_j(t)\ dt\ . \end{aligned} \tag{3}$$

The input layer is fully connected to the hidden and output layers. Every node in the hidden layer is connected to every other node in the hidden and output layers and to itself. The two outputs of every neuron are computed according to the relationship: [19]

$$o_{1j}^{h}(t) = A_j \psi_j (B_j^{-1}[\sum_{k=0}^{m}(w_{1jk}^{hi} o_{1k}^{i}(t) + w_{2jk}^{hi} o_{2k}^{i}(t)) + \sum_{k=0}^{n}(w_{1jk}^{hh} o_{1k}^{h}(t) + w_{2jk}^{hh} o_{2k}^{h}(t))]\ )$$

$$o_{2j}^{h}(t) = C_j \psi_j (D_j^{-1} \int_0^t o_{1j}^{h}(t)\ dt) \tag{4}$$

where y_j is a nonlinear transformation function, and A_j, B_j, C_j, and D_j are adjustable weights associated with the hidden neuron j, which are used to shape the transfer function for every node. B and D are used to adjust the steepness of the function, while A and C are used to adjust its min-max value. Also m and n are the number of processing nodes in the input and hidden layers respectively; w_1 and w_2 refer to weights associated with direct and delayed outputs, respectively. The superscript h refers to the hidden layer, i refers to the input layer, hi refers to weights from the input to hidden layer, and hh refers to weights from the hidden to hidden layer. The integrators and feedback connections promote enough asynchrony and interaction in the network to model several system state variables as a function of time. When enough intermediate state variables are generated, the network output can be a function of those state variables. The output layer has only one output per node, which is computed according to the following equation: [19]

$$o_{j}^{o}(t) = E_j \psi_j (F_j^{-1}[\sum_{k=0}^{m}(w_{1jk}^{oi} o_{1k}^{i}(t) + w_{2jk}^{oi} o_{2k}^{i}(t)) + \sum_{k=0}^{n}(w_{1jk}^{oh} o_{1k}^{h}(t) + w_{2jk}^{oh} o_{2k}^{h}(t))]) \tag{5}$$

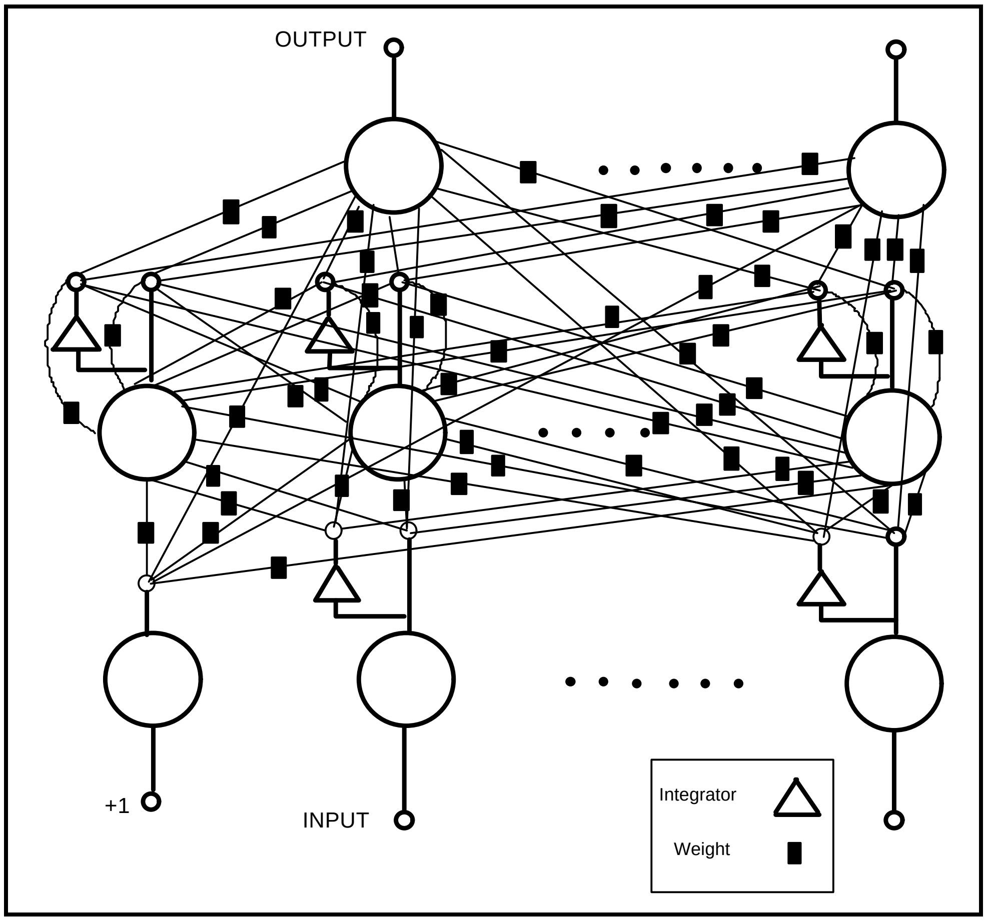


Figure 7 Schematic diagram of the Dynamic System Imitator (DSI) Network.

where m and n are the number of nodes in the input and hidden layers respectively, and E_j and F_j are two constants associated with each node in the output layer to shape its own transfer function when needed. The superscript o refers to the output layer, i refers to the input layer, h refers to the hidden layer, oi refers to weights from the input to output layer, and oh refers to weights from the hidden to output layer.

By looking at the complete DSI network design, it is easy to observe that the node interaction, information feedback, and action transfer time lags generate an activity in the network that is similar to the internal activity in real dynamic systems. Even with a simple configuration, the DSI has a complex structure, which makes it very difficult to train. A multi-dimensional optimization technique that adopts the simplex method is used to train the DSI. [6] There are two other difficulties in the training of such a network. One is that a very long time series cannot be introduced to the network at one time and must be divided into reasonably sized sections. The other is that the behavior of the network is dependent on the initial conditions of its state variables, and a certain set of initial conditions has to be found in conjunction with every network design. In other words, whenever the network is updated, a new set of initial conditions must be found. The first problem was overcome by using a moving time window that cascades the introduction of the time series segments to the network during training. The network final conditions at the final training step of every segment is taken as the initial conditions for the next segment, to keep the physical association between the consequent segments of the time series. The second problem was overcome by adding initial conditions search method that runs after every training iterate to find updated initial conditions for every modified version of the network. An arbitrary set of initial conditions can be used at the start of the training process.

4.3 Using The DSI Neural Network for Chaotic Behavior Control

In this section we will demonstrate how a special type of a neural network, Dynamic System Imitator (DSI), can be used for chaotic behavior control. The DSI network is specially designed to model a wide variety of dynamic systems. It is a multi-layer recurrent neural network supplied with integrators and extensive feedback connections to model the asynchrony and time lags in the real systems. The DSI neural network was used before for modeling complex dynamic behaviors through very simple configurations due to its temporal and spatial representations. The DSI network used in this project has one node in the input layer three nodes in the middle layer and one node in the output layer. This DSI controller was applied to control the chaotic behavior of the Lorenz system to a stable fixed point or a stable periodic orbit. Both system state point control and system parameter perturbation control strategies were implemented.

4.3.1 Introduction to Chaos Control

Scientists in many fields often encounter systems that exhibit chaotic time evolution.[1] Chaos is abundant both in nature and man-made devices, to an extent that many scientists believe that it is the rule rather than the exception.[2] The chaotic behavior is known to be unpredictable, which may be unsafe to the operation of many devices, and make it unwelcome in many situations. On occasion, chaos is a beneficial feature as it enhances mixing and chemical reactions and provides a vigorous mechanism for transporting heat and/or mass. However, in many other situations, chaos is undesirable phenomena which may lead to vibrations, irregular operation, fatigue failure in mechanical systems, temperature oscillations which may exceed safe operational conditions in thermal systems, and increasing drag in flow systems.[2] Chaotic motion has been regarded for many years as a troublesome property that is neither predictable nor controllable. Recently researchers have realized that chaos can actually be advantageous in many situations, and when it is unavoidably present, it can often be controlled to obtain desired results.[1,3] In 1990, Ott, Grebogi, and Yorke (OGY) demonstrated that one can convert the motion of a chaotic dynamical system to periodic motion by controlling one of the system's many unstable periodic orbits embedded in the chaotic attractor, through only small time-dependent perturbations in an accessible system parameter.[4, 5] Ott and Spano [1], stated that if chaos control is practical in a system, then the presence of chaos can be an advantage. Any one of a number of different unstable orbits, in a chaotic system, can be stabilized, and one can select the orbit that gives the best system performance. Thus we have the flexibility of actually switching the system behavior by stabilizing another periodic orbit. On the other hand, if the system is actually stable and periodic, we can use control to only slightly change its performance, and we do not have similar flexibility to what is available in a chaotic system. We may even sometimes wish to build chaos into a system. Several other methods to control a chaotic system have emerged from the OGY method, such as the "Occasional Proportional Feedback" (OPF).[6, 7] , and the "Proportional Perturbation Feedback" (PPF).[8] Chaotic system control has been applied usefully in several fields of science and engineering. It was applied to control chemical reactions,[9, 10] communication,[11, 12] nonlinear oscillators,[11], lasers,[1, 6] diodes,[7] heat convection,[2] mechanical vibrations, myocardial tissue (biological systems),[8] magnetoelastic systems,[1, 3] and several other applications.

In this section, we will demonstrate how the Dynamic System Imitator (DSI), can be used to control a chaotic system. For the purpose of demonstration, the method was applied to control the Lorenz system. The Lorenz system is a system of non-linear differential equations that was used by Edward Lorenz in the sixties to describe the atmosphere dynamics for weather prediction:[21]

$$\begin{aligned}\frac{dx}{dt} &= sy - sx \\ \frac{dy}{dt} &= rx - y - xz \\ \frac{dz}{dt} &= xy - bz\end{aligned} \tag{6}$$

The Lorenz system exhibits a chaotic behavior for a certain range of parameters. Figures 8 and 9 show the Lorenz system behavior for $s = 10,\ r = 28,$ and $b = 8/3$. The same parameters are used to generate the system behavior needed to test the control method demonstrated.

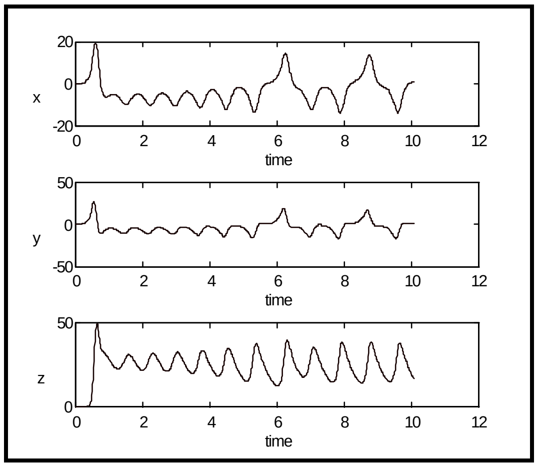


Figure 8 The chaotic behavior of the Lorenz system starting at (0.05,0.05,0.05).

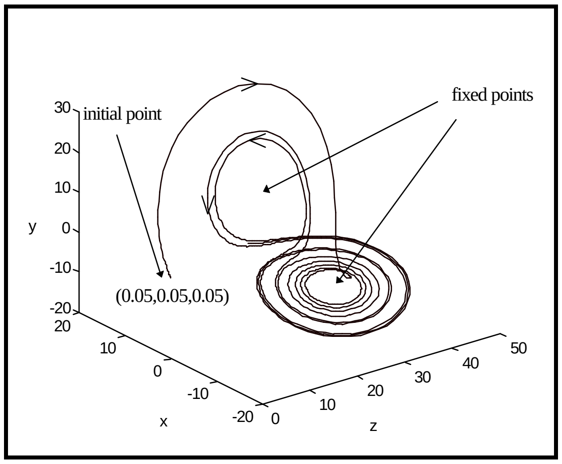


Figure 9 The Lorenz system state space attractor.

4.3.2 Methodology to Use the DSI to Control a Chaotic System

In the control method presented in this section, a DSI neural network is used to model the controller. The DSI network used, has one node in the input layer, three nodes in the hidden layer and one node in the output layer. From previous experience with the DSI network, we believe this configuration is enough to model the desired controller. The general training strategy for the DSI controller is shown in Figure 10. The DSI is trained to generate the necessary control signal to achieve certain system performance. One or more reference values in addition to feedback from the controlled system are used as inputs to the DSI. A pre-specified output behavior of the system is also supplied to the network as a target for the training. During training this pre-specified behavior is continuously compared with the actual behavior of the system to adjust the DSI

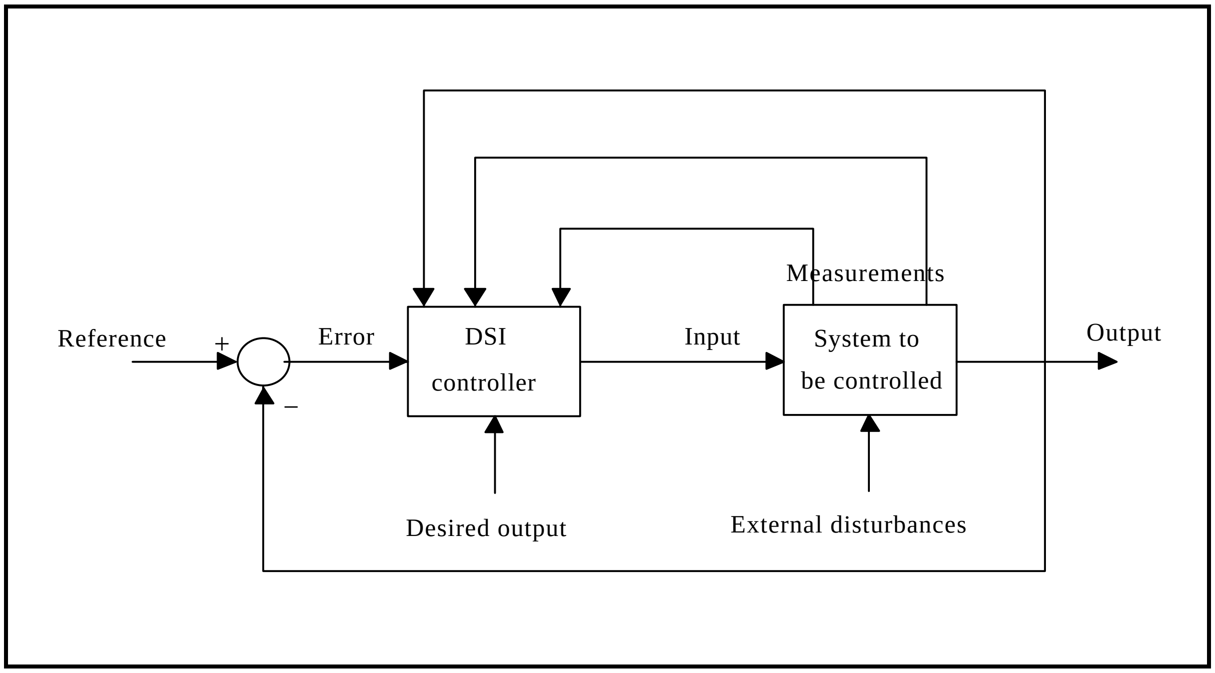


Figure 10 The general training strategy for the DSI controller.

controller parameters. We used the DSI controller to control the chaotic behavior of the Lorenz system through both system state point perturbation and system parameter perturbation. For state point perturbation we studied two cases, in one of them we insert the control action (u(t)) in the second equation and in the other we insert the control action in the third equation of the system. For system parameter perturbation we also studied the control of the parameter ‘r’ in two ways, one of them using the control action as an additive control term, and the other using the control action as a multiplicative control factor. The first case, where we controlled the system state point through the second equation of the system is represented in Equation 7:

$$\begin{aligned}\frac{dx}{dt} &= -sx + sy \\ \frac{dy}{dt} &= rx - y - xz + u(t) \\ \frac{dz}{dt} &= xy - bz\end{aligned} \tag{7}$$

The control action is calculated using the DSI controller. The input to the DSI controller is the x(t) signal of the Lorenz system. During training, system outputs (x(t), y(t), z(t)) are to be monitored to achieve a target trajectory for the system. As a less restrictive requirement in this case, only y(t) was monitored by the training algorithm to stabilize the system at a value near one of the critical points (**p** or **q**). A training algorithm that adopts the simplex method for its multidimensional minimization, was used. The objective function of such minimization included the rms difference between the actual and desired output of the system. The system was integrated using a fourth order Runge Kutta ordinary differential equation (ODE) solver. The Lorenz system equations and the ODE solver were linked to the DSI network program, to allow the interaction between the DSI and the Lorenz system, both during training and recall.

4.3.3 Sample Results

For all cases, the training achieved the target solution, through the DSI parameter adaptation, within very reasonable number of iterations (around 30 iterations). Figures 11 and 12 show the time behavior and trajectory of the Lorenz system after applying and then removing the control. The control action (u(t)) was applied shortly after the start of the run, and then removed after system stabilized at one state point (time 4), as indicated in Figure 11. We can notice that after removing the control action, the trajectory went back to the chaotic behavior. The reason is that the stabilized point is obviously not a fixed point of the original system, and the state point will not stay stable except if the control action is maintained. The second test for this control strategy was to control the chaotic system to stabilize to a periodic orbit by forcing its trajectory to pass through a pre-specified point in the space. Figures 13 and 14 show the time behavior and trajectory of the controlled Lorenz system. The control action (u(t)) was applied at the beginning of the run as indicated in Figure 13. Both the time behavior and the state space trajectory show how the system stabilized to a periodic orbit. After removing the control, the trajectory went chaotic. We did not plot the results after removing the controller, because it would elaborate the figure and hide some of the details of the periodic orbit shown.

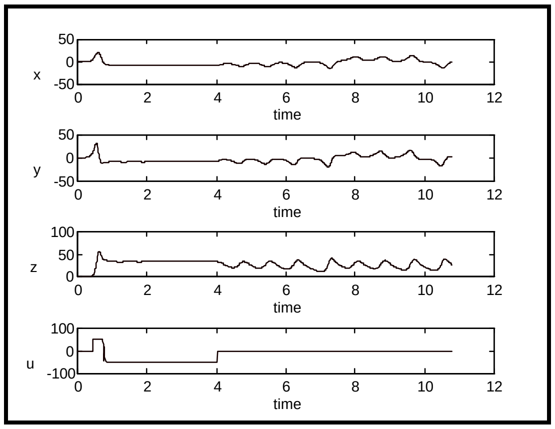


Figure 11 The DSI control signal (u(t)), and the time behavior of the Lorenz system after applying and then removing the control at the stabilized point.

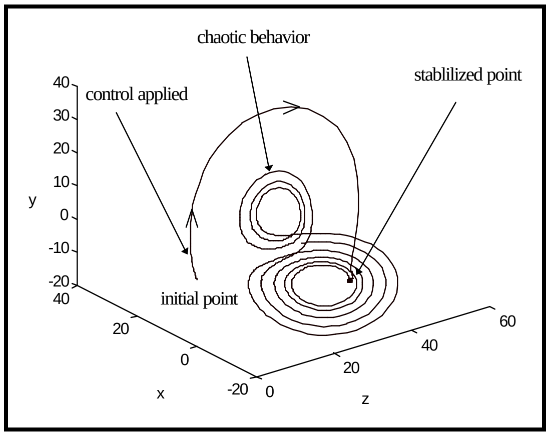


Figure 12 The trajectory of the Lorenz system, after applying and then removing the control action at the stabilized point.

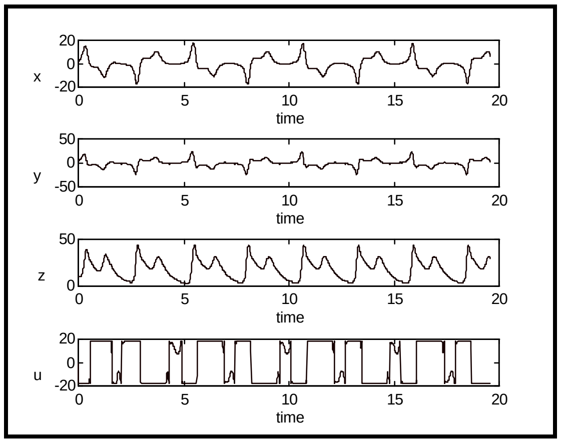


Figure 13 The time behavior of the Lorenz system, controlled by the DSI to achieve a periodic orbit, starting at (1,5,10).

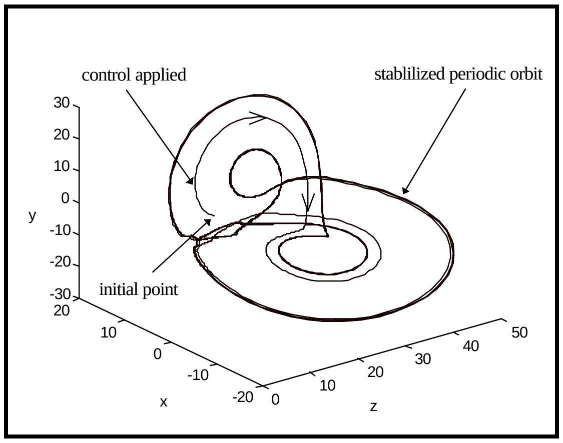


Figure 14 The trajectory for the Lorenz system, controlled by the DSI to achieve a periodic orbit, starting at (5,5,1).

4.4 Iterative Prediction of Chaotic Time Series Using the DSI Neural Network

Chaotic systems are known for their unpredictability due to their sensitive dependence on initial conditions. When only time series measurements from such systems are available, neural network based models are preferred due to their simplicity, availability, and robustness. However, the type of neural network used should be capable of modeling the highly non-linear behavior and the multi-attractor nature of such systems. In this Section we use the DSI neural network, that has been proven to be capable of modeling very complex dynamic behaviors. The prediction method presented in this section is based upon predicting one step ahead in the time series, and using that predicted value to iteratively predict the following steps. This method was applied to chaotic time series generated from the logistic, Henon, and the cubic equations, in addition to experimental pressure drop time series measured from a Fluidized Bed Combustor (FBC), which is known to exhibit chaotic behavior. The time behavior and state space attractor of the actual and network synthetic chaotic time series were analyzed and compared. The correlation dimension and the Kolmogorov entropy for both the original and network synthetic data were computed. They were found to resemble each other, confirming the success of the DSI based chaotic system modeling.

4.4.1 Introduction to Iterative Prediction of Chaotic Time Series

Chaotic systems are known for their unpredictability, due to their sensitive dependence on initial conditions which is measured by positive Lyapunov exponents. [1] In other words, even when the exact model of a chaotic system is available, it is impossible to predict a chaotic system behavior for a long period of time. [22, 23] The reason is that our measurements and calculations are never perfect and are susceptible to errors. Similar errors contribute to the non-exact determination of initial conditions. Any minute error in the initial conditions for a chaotic system will turn, with time, into great differences in the results. However, short term predictions of chaotic systems are still possible. [22, 24] How short the time duration is for valid prediction depends on the system average loss of information represented by its Lyapunov exponents and Kolmogorov entropy. Some systems are less predictable than others due to faster loss of information with time, represented by larger positive Lyapunov exponents and larger positive Kolmogorov entropy. Since exact predictions are not possible for such systems, approximate models may produce results as satisfactory as those produced by exact models. This makes neural network based models very good candidates for such applications. Neural network based models are known not to be exact models, but they are easy to implement, robust, fast and data driven. Dynamic neural network models are preferable for such applications, due to their ability to capture time behaviors. [25]

In this work we used a special type of dynamic neural network called the Dynamic System Imitator (DSI). [26] We developed the DSI a few years ago and have used it for several modeling and control applications. [26, 27, 28] The DSI is biologically motivated and is specially designed to model a wide variety of dynamic systems. It has both short term and long term memory mechanisms that enable the modeling of a system's transient and steady state behavior. In addition, the DSI behavior depends on its initial conditions the same as any differential equation model does, even though no explicit differential equation solving is incorporated in this case. What we know of the DSI characteristics encourages us to recommend it for modeling non-linear systems in general and chaotic systems in particular. Since the dynamics of most real systems are accessed via time series measurements, the focus in this section will be on modeling chaotic time series. The way the chaotic time series model is implemented in this section is through a one step predictor model. The dynamics of a chaotic time series are modeled through training the DSI to perform a one step prediction. However, at any point of time, the DSI response depends on the initial conditions at time zero, the history of inputs and network state variables, and the current network input. Assuming the network was able to capture the dynamics in the time series, we can start the trained network with any set of initial conditions, use a number of initial data points to put the network on track, and iteratively feed the output of the network back to compute next predicted values. Even though we applied this methodology to several theoretical systems, the current motive is to use it in the proposed strategy to identify normal/abnormal chaotic behavior modes encountered in an (FBC) system. This identification can be achieved by comparing the actual measurement from the chaotic system with the time series predicted by the DSI iterative predictor model, starting from a short time history of the actual data. In this section, results from the DSI iterative predictor are discussed for chaotic time series generated using the logistic, Henon and the cubic equations, in addition to one experimental time series measured from an FBC system. The DSI network model was evaluated based on comparison made on the time series, phase space trajectories, and chaotic parameters computed from these trajectories. However, in this case, time series similarities are not as important as similar phase space trajectories and similar chaotic parameters. [25]

4.4.2 Using the DSI for iterative prediction of Chaotic Time Series

A simple configuration of the DSI neural network was used for the iterative prediction of a chaotic time series. This configuration has one node in the input layer, three nodes in the hidden layer, and one node in the output layer. The network was trained to predict one point ahead of the time series, using a set of previous values. These values are not explicitly used for prediction, but are implicitly used by adjusting the state of the network from which the prediction is performed. The prediction method is based upon the idea that once the network is trained to predict one point ahead with good accuracy, this same point can be used as an input to the network to predict the next point. This process can be repeated iteratively to predict many points in the time series. Naturally, the accuracy of prediction will deteriorate over time. During training, a time window of 200 points was used to cascade the time series to the network. The algorithm was applied to three types of simulated chaotic time series generated from the logistic, Henon, and cubic equations, in addition to one experimental time series measurement taken from an FBC system. FBC systems are known for their chaotic behavior, as discussed in several references. [29, 30, 31, 32] The network was able to learn simple one step prediction in a reasonable number of training iterations. After training, the output of the DSI was used iteratively to generate the time series. However, the training initial conditions together with the first actual 25 points of the training time series should be used to start the DSI, in case the time behavior of the training time series must be generated. If not, any network initial conditions and any starting points can be used to generate the state space behavior of the system to which the training time series belongs.

Comparing the predicted time series to the actual time series, we found that the DSI was able to track the training time series time behavior for a short period of time (around 30 points), when started from the training initial conditions , and activated by the first 25 points of the training time series. However, it was able to track the state space attractor to which the training time series belongs, starting from any initial conditions, activated by any arbitrary set of starting points. The only case that fails is zero initial conditions together with zero starting points, which leads to zero solution The actual and predicted time series for all cases are shown in Figures 15-22, while the actual and predicted state space attractors are shown in Figures 23-30. To quantitatively compare these attractors, the correlation integral and Kolmogorov entropy for the actual and predicted attractors were computed according to Equations 1 and 2 respectively. . The results of the correlation dimension and Kolmogorov entropy of the different cases are summarized in Table 2. The correlation dimension is computed, according to the box-counting method, from the slope of the lines representing the correlation integral versus ε (the size of a computing box) on log-log curves for different embedding dimensions.

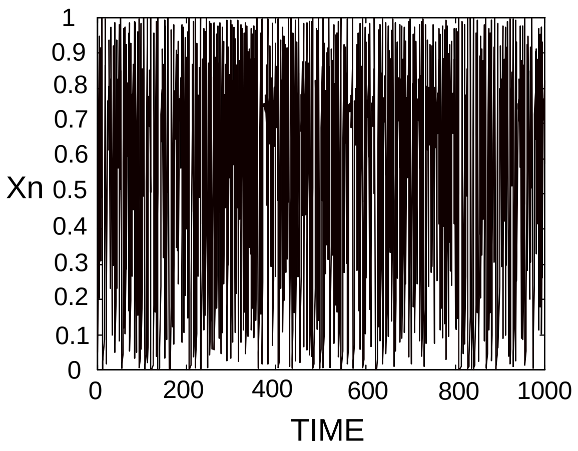


Figure 15 Actual logistic time series.

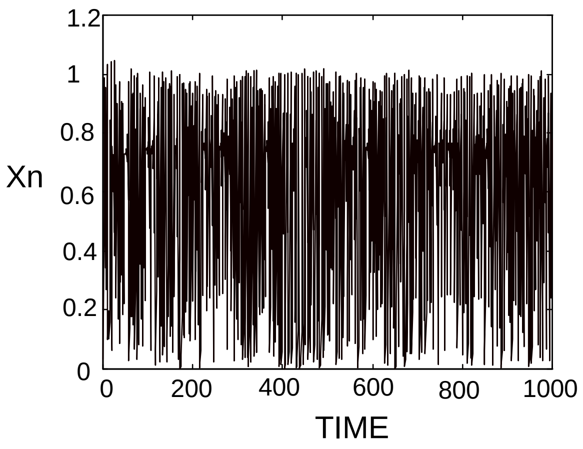


Figure 16 Synthetic logistic time series.

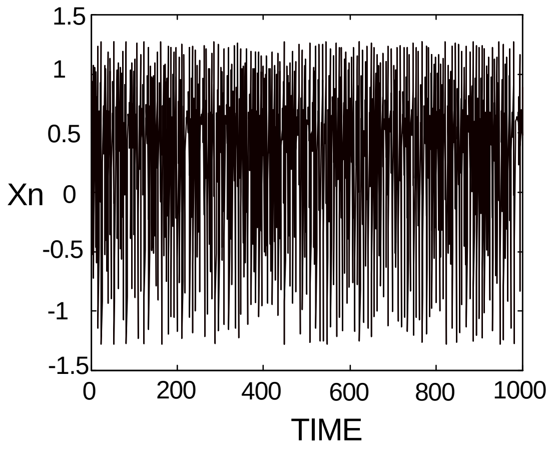


Figure 17 Actual Henon time series.

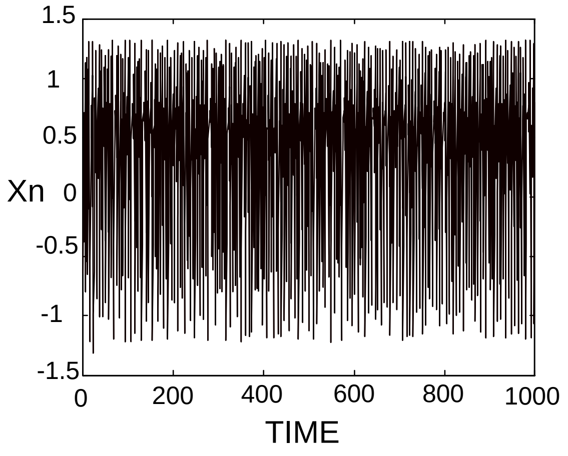


Figure 18 Synthetic Henon time series.

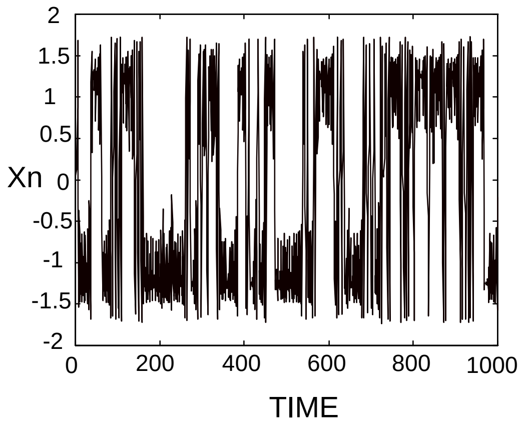


Figure 19 Actual cubic time series.

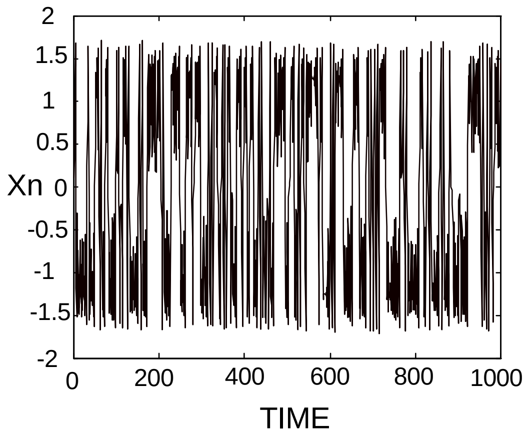


Figure 20 Synthetic cubic time series.

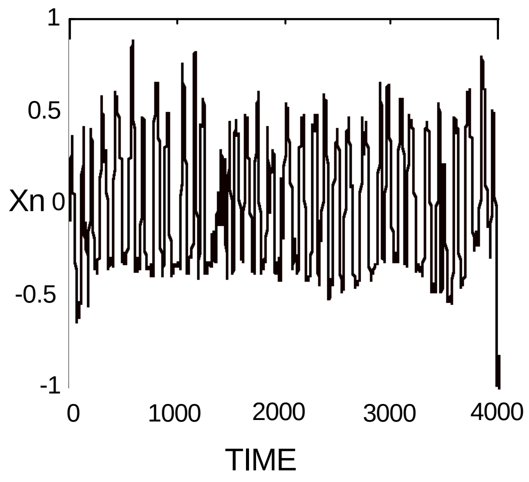


Figure 21 Actual normal FBC time series.

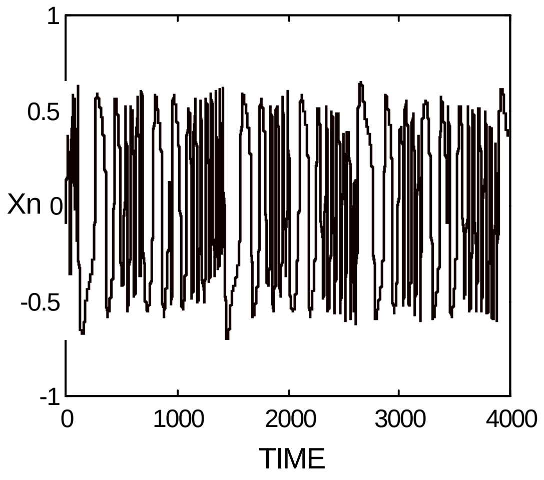


Figure 22 Synthetic normal FBC time series.

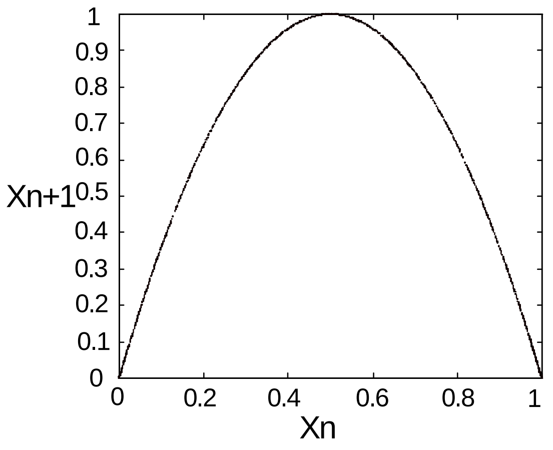


Figure 23 Actual logistic attractor.

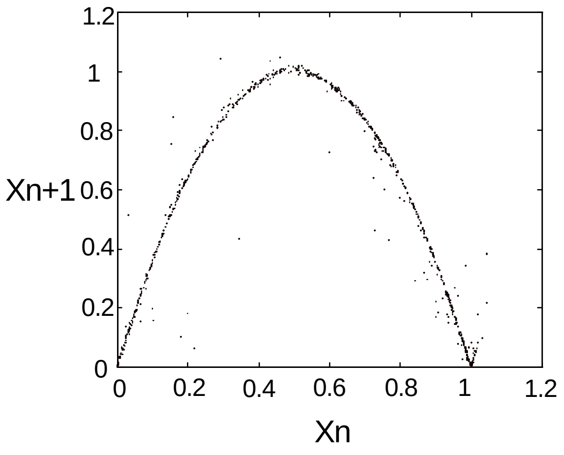


Figure 24 Synthetic logistic attractor.

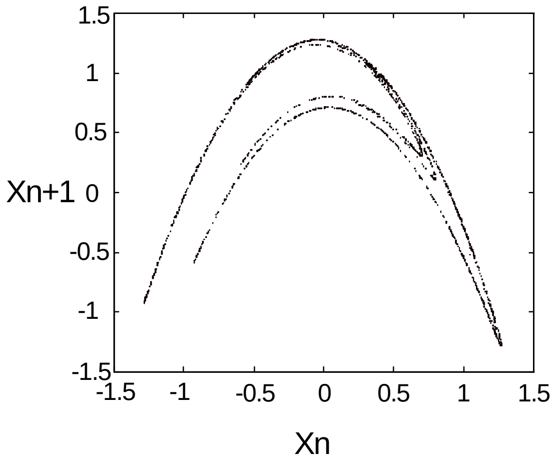


Figure 25 Actual Henon time attractor.

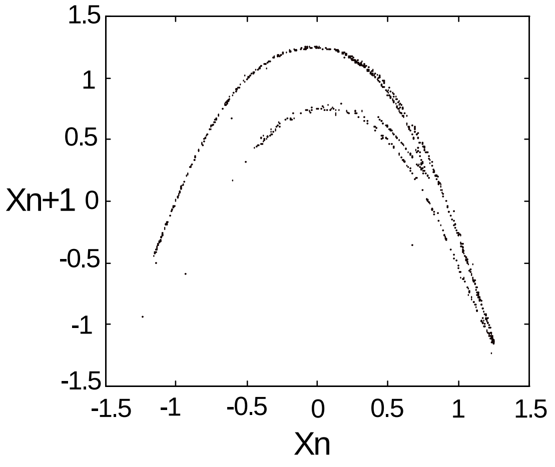


Figure 26 Synthetic Henon attractor**.**

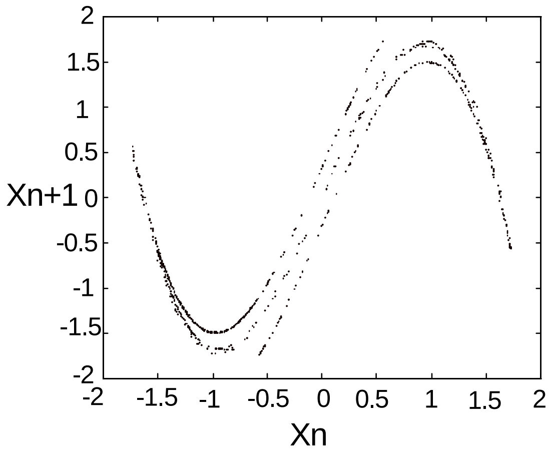


Figure 27 Actual cubic attractor.

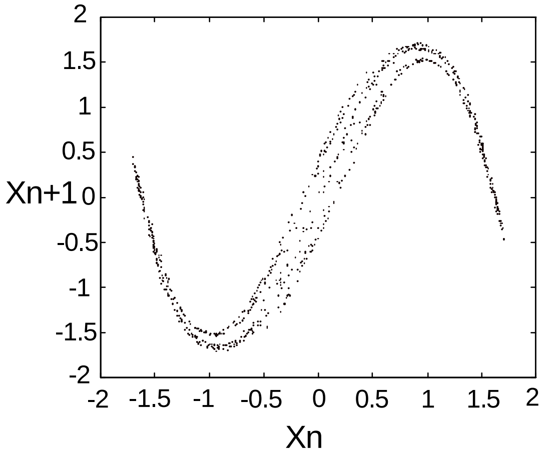


Figure 28: Synthetic cubic attractor.

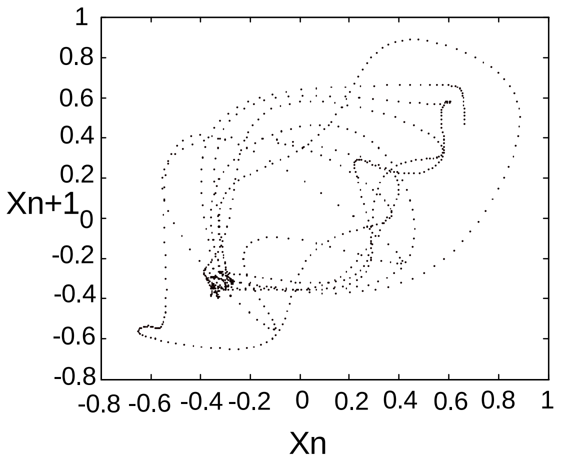


Figure 29 Actual normal FBC attractor.

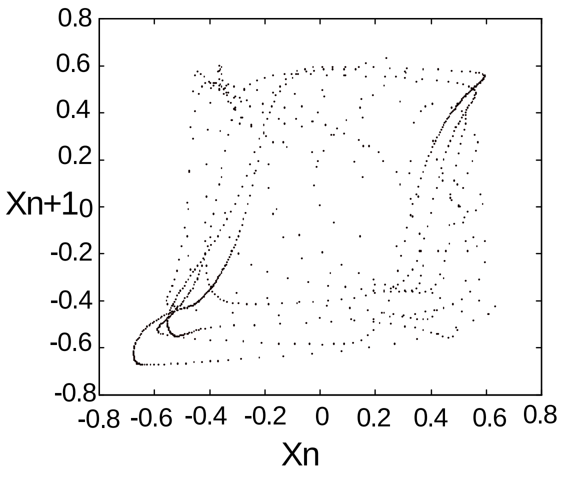


Figure 30 Synthetic normal FBC attractor.

TABLE 2: COMPARISON OF THE ACTUAL AND DSI SYNTHETIC ATTRACTORS PARAMETERS

Time Series	Correlation Dimension	Kolmogorov Entropy
Logistic Map (actual)	1.0457±0.0057	0.6872±0.0198
Logistic map (synthetic)	1.2316±0.0374	0.6013±0.0435
Henon Map (actual)	1.2607±0.0331	0.3267±0.0135
Henon map (synthetic)	1.3171±0.0725	0.2962±0.0239
Cubic Map (actual)	1.2248±0.0090	0.4245±0.0183
Cubic Map (synthetic)	1.8171±0.01365	0.5340±0.0134
FBC normal (actual)	2.934±.065	5.034±.095
FBC normal (synthetic)	2.12±.07	6.3764±1.26

4.5 Using The OGY Technique to Control a Chaotic Behavior to a Stable Periodic Orbit

The sensitivity to initial conditions and orbit complexity which characterize chaotic systems give them a great flexibility to control as compared to non-chaotic systems. The motion of chaotic dynamical systems can be converted to periodic motion through small time-dependent perturbations, which is a technique developed by Edward Ott, Celso Grebogi, and James York in 1990, known as the OGY technique. In this section, a systematic way to compute the OGY control parameters from a chaotic time series is discussed. It is shown how to use autoregressive modeling to estimate the linear map that describes the system iterative behavior around a fixed point, which is essential for the OGY technique. It is also explained how to experimentally compute the rate of change of a fixed point with respect to a control parameter. And the procedure to be followed according to the order of the controlled periodic orbit is presented. Finally, it is demonstrated how these techniques can be applied to control the chaotic behavior of some typical chaotic systems, such as the logistic and Henon maps, to any periodic orbit. Similar techniques can be combined with neural network-based techniques, discussed above, to control the abnormal chaotic behavior in FBC systems.

4.5.1 Small Perturbation Control of Chaotic Systems

Chaotic systems are known for their unpredictability. Thus, they were mistaken in the past with random systems. It was found later on that chaotic systems are driven by deterministic phenomena, and their unpredictable behavior is due to their sensitive dependence on initial conditions [1]. The presence of chaos may be a great advantage for control in some situations [3]. In non-chaotic systems small control force will change the system dynamics slightly. On the other hand, in chaotic systems small control force can cause a large change in the system behavior. Also, a wide choice between a rich variety of dynamical behavior is possible[3]. Based on this observation, several chaotic system control methods have been developed [1-6, 33]. Small perturbation control of chaotic systems is a technique developed by Edward Ott, Celso Grebogi, and James York in 1990, known as the OGY technique [4]. In this technique, the chaotic system motion can be stabilized to one of its naturally unstable periodic orbits. This can be achieved by applying time-dependent small pre-calculated perturbations to a system parameter. These control perturbations are applied only when the system behavior comes to a vicinity of the fixed point corresponding to the periodic orbit to be stabilized.

The first step in the OGY technique is to map the system behavior to a chosen surface of section such that the periodic orbit to be stabilized appears as a fixed point. Second, the stable and unstable manifolds at this fixed point are computed from the eigenvalues and the corresponding eigenvectores of the linear map that describes the system behavior around the fixed point. The rate of change of the fixed point with respect to the control parameter is also computed. We wait for the system to come in the vicinity of the fixed point and then apply a small perturbation such that the system's next iterate falls on the stable manifold of the fixed point. In theory , once the system is on the stable manifold, its iterates will move toward the fixed point by natural forces. For the OGY method to be implemented, a linear map that describes the system behavior around the fixed point to be controlled is essential. This linear map can be estimated from time series measurements from the system. No exact model for the system is needed, which makes it possible for this method to be implemented in a wide range of problems [7-10]. In this section, we present a systematic way to compute the OGY control parameters from a chaotic time series. We show how to use autoregressive modeling to estimate the linear map that describes the system iterative behavior around a fixed point. We also explain how to experimentally compute the rate of change of a fixed point with respect to a control parameter, and how to modify the algorithm according to the order of the periodic orbit to be controlled. Finally, we demonstrate how these techniques can be applied to control the chaotic behavior of some typical chaotic systems, such as the logistic and Henon maps, to any periodic orbit.

4.5.2 Review for the OGY Technique

The OGY method is a technique to stabilize the chaotic behavior in an n-dimensional chaotic system to one of its naturally unstable periodic orbits. First, the system behavior has to be mapped on a chosen surface of section or a return map, in order to observe the system dynamics as a sequence of points on a two dimensional map. Let $\zeta_1,\ \zeta_2,\ \zeta_3,\ldots\ldots\zeta_n$ denote the coordinates in the surface of section at the n'th piercing of the surface of section. Suppose the iterates are represented by

$$\zeta_{n+1} = f(\zeta_n, p) \tag{8}$$

where P is some accessible system parameter

Then, we examine the unstable periodic orbits and select the one to be used for control, and obtain its stability properties. For purposes of simplicity, let us assume that a first order return map is constructed from a one dimensional time series $x_1,\ x_2,\ \ldots\ldots\ldots,\ x_n$, then the two dimensional iterates ζ_n will be defined as:

$$\zeta_n = \begin{bmatrix} x_{n+1} \\ x_n \end{bmatrix} \tag{9}$$

From this map a fixed point or a periodic orbit will be selected and examined. A fixed point is defined as $x_{n+1} = x_n$, while a period k orbit is defined as $x_{n+k} = x_n$. The period k orbit, as an example, will appear as k distinct points on the first return map, while it will appear as a fixed point on the k-order return map. We will always need to select the appropriate return map order that shows the controlled periodic orbit as a fixed point

$$\zeta_F = \begin{bmatrix} x_F \\ x_F \end{bmatrix} \tag{10}$$

The OGY technique assumes that there is an acceptable maximum perturbation δp_m in the system control parameter P. It is also assumed that the position of the periodic orbit is a function of P, while the local dynamics around this periodic orbit do not vary much when the parameter P is changed within the allowable perturbation [5].
In order to examine the stability of the selected periodic orbit we need to perform the following computations:
1. We find the linear approximation of the map f around the selected fixed point, as shown in the following equation:

$$\zeta_{n+1} - \zeta_F = M(\zeta_n - \zeta_F) \tag{11}$$

where M is a two dimensional matrix.

2. Let λ_s and λ_u be the experimentally determined stable and unstable eigenvalues of the matrix M, respectively, and $(e_s,\ e_u)$ are the corresponding stable and unstable eigenvectors. The eigenvectors e_s and e_u represent the stable and unstable directions of the map around the selected fixed point, and are used to compute the unstable contravariant eigenvector according to the relationships $f_u . e_u = 1$ and $f_u . e_s = 0$ [4].

3. The rate of change of the selected fixed point with respect to the system control parameter P is computed as:

$$g = \frac{\partial \zeta_F}{\partial p} \approx \frac{\Delta \zeta_F}{\Delta p} \tag{12}$$

4. Given the maximum allowable perturbation in the control parameter δp_m, we can compute the maximum effective control distance around the fixed point as:

$$\delta \zeta_m = (\frac{\lambda_u - 1}{\lambda_u})\ \delta p_m\ (g. f_u) \tag{13}$$

Knowing the parameters computed above, we can compute the necessary control force at iterate n as:

$$\delta p = C(\zeta_n - \zeta_F). f_u \tag{14}$$

$$\text{where} \qquad C = (\frac{\lambda_u}{\lambda_u - 1})\ (g. f_u)^{-1} \tag{15}$$

The perturbation δp is the control action necessary to put the system's next iterate on the stable manifold of the fixed point. Then, in the following iterates the system behavior should move towards the fixed point with the system's natural forces. However, to adjust for noise and computation inaccuracies, we need to calculate a correcting control force δp at the end of every period.

4.5.3 Methodology to Compute The OGY Parameters from a Chaotic Time Series

In this section, we present the details of our approach to compute the OGY parameters and control action from chaotic time series measurements. The procedure is summarized in the following steps:

Step 1. *Determination of the System Periodic Orbits and Fixed Points*

Starting from a time series measurement at a nominal value of the control parameter P_0, we need to localize the periodic orbits embedded in the system. Actually, we will be interested in only one of these orbits with the targeted period. All periodic orbits embedded in a chaotic system are unstable orbits. This means that the system will visit those orbits briefly at certain instances. We need to catch this brief periodic behavior of the system, localize it, and determine its behavior, in order to compute the control actions needed to stabilize it. The time series measurements can be easily arranged in a two column matrix defined as:

$$X = \begin{bmatrix} x_1 & x_{1+k} \\ x_2 & x_{2+k} \\ . & . \\ . & . \\ x_{n-k} & x_n \end{bmatrix} \tag{16}$$

where n is the total number of data points in the time series, and k is the period of the targeted orbit. Any two points on the same row of the matrix X are separated by k number of points. If a pair of points x_n and x_{n+k} on the time series are equal, then any of the periods k, k/2, k/3.....or k/k=1 is localized. Any of these periods, if exist, will appear as a fixed point on the k-order return map (x_n vs. x_{n+k}). If the target period k exists, any point on it visited by the system for a complete cycle in the measured interval will appear as a distinct fixed point on the k-order return map. If all k orbital points were visited, they will appear as k distinct fixed points on the k-order return map. We compute the absolute value of the difference of the two columns of the matrix X that comes to a vector of n-k values. We search this vector for any value less than a minute tolerance ε. Any of the values passing the selection criteria will mark one fixed point on the k-order return map. After marking a fixed point on a specific row, we need to go back and unfold all points between x_n and x_{n+k} at the marked row. If points x_n ...x_{n+k-1} are all distinct, then this is one period k orbit, and the corresponding fixed point on the k-order return map is a point of interest to the current control task. Any of the k fixed points belonging to the orbit $(\zeta_{01}, \zeta_{02}, \ldots\ldots, \zeta_{0k})$ can be used to control the system.

Step 2. *Computing the Rate of Change of the Fixed Point with Respect to the Control Parameter*

New measurements from the system need to be recorded at values of the control parameter $p_0 \pm \delta p$. For every measurement we need to recalculate the location of the period k fixed points on the k-order return map $(\zeta_{m1}, \zeta_{m2}, \ldots\ldots, \zeta_{mk})$, using the method on step 1. In case we have decided to use the fixed point ζ_{0j} to control the system, the change in its position will be $\delta\zeta_i = \zeta_{ij} - \zeta_{0j}$, and the rate of change of that fixed point will be computed as $g_i = \frac{\delta\zeta_i}{\delta p_i}$, where $\delta p_i = p_i - p_0$. We repeat the computation for the available measurements (0, 1, ..., m) and average them to get the value of g to be used.

Step 3. *Estimating the Linear Approximation for the Map f around a Fixed Point*

As shown in equation 8 the map *f* correlates the consecutive points on a two dimensional map. To apply the OGY method, we need a linear estimate of that map around the fixed point used to control the system. This means we need to estimate the two dimensional matrix that maps the consecutive points in a close vicinity of the fixed point. Equation 11 can be expanded as:

$$\begin{bmatrix} x_{n+2} - x_F \\ x_{n+1} - x_F \end{bmatrix} = \begin{bmatrix} m_{11} m_{12} \\ m_{21} m_{22} \end{bmatrix} \begin{bmatrix} x_{n+1} - x_F \\ x_n - x_F \end{bmatrix} \tag{17}$$

From the second row equality of this equation, we can easily infer that $m_{21} = 1$, and $m_{22} = 0$. On the other hand the first row equality is:

$$x_{n+2} - x_F = m_{11}(x_{n+1} - x_F) + m_{12}(x_n - x_F) \tag{18}$$

This equation means that the difference between an iterate and the fixed point can be obtained from the difference of the two previous iterates to the fixed point. We can consider this equation as an autoregressive model for the iterates, and use the least squares method to estimate the parameters of that model. From the matrix X in equation 16, in which every row represents an iterate on the two dimensional k-order return map, we collect all pairs of points ζ_n and ζ_{n+1} that are in a close vicinity to the fixed point. In other words, we collect the pairs that lie inside a small pre-defined circle around the fixed point. According to the least squares estimate of the autoregressive model, we can estimate the m_{11} and m_{12} elements of the matrix M as:

$$\begin{bmatrix} m_{11} \\ \\ m_{12} \end{bmatrix} = D^{-1} \begin{bmatrix} \frac{1}{N}\sum (x_{n+2} - x_F)(x_{n+1} - x_F) \\ \frac{1}{N}\sum (x_{n+2} - x_F)(x_n - x_F) \end{bmatrix} \tag{19}$$

where N is the number of pairs and D^{-1} is given by

$$D^{-1} = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix}$$

$$d_{11} = \frac{1}{N}\sum (x_{n+1} - x_F)^2$$

$$d_{12} = \frac{1}{N}\sum (x_{n+1} - x_F)(x_n - x_F)$$

$$d_{21} = \frac{1}{N}\sum (x_{n+1} - x_F)(x_n - x_F)$$

$$d_{22} = \frac{1}{N}\sum (x_n - x_F)^2$$

Step 4 . *Finding the OGY Parameters*

We compute the eigenvalues and the corresponding eigenvectors of the matrix M. Assume that λ_s and λ_u are the experimentally determined stable and unstable eigenvalues, respectively, and e_s and e_u are the corresponding eigenvectors. The eigenvectors e_s and e_u represent the stable and unstable directions around the fixed point. From the eigenvectors e_s and e_u we compute the unstable contravariant eigenvector according to the relationships $f_u . e_u = 1$ and $f_u . e_s = 0$ [4]. Hence, the maximum allowable distance for control around the fixed point can be computed according to equation 13 as:

$$\delta\zeta_m = (\frac{\lambda_u - 1}{\lambda_u})\ \delta p_m\ (g. f_u)$$

and the necessary control perturbation for the n'th iterate can be computed by equation 14 as:

$$\delta p = C(\zeta_n - \zeta_F). f_u$$

where $C = (\frac{\lambda_u}{\lambda_u - 1})\ (g. f_u)^{-1}$

The way the OGY control method is applied, is to wait for the system until an iterate falls into a circle with a radius $\delta\zeta_m$ around the fixed point, then apply the perturbation $p = p_0 + \delta p$. Then, we wait k iterates and check if the system falls back within the radius $\delta\zeta_m$ around the fixed point. If it does, we measure the distance between that iterate and the fixed point, re-estimate the control force and reapply the necessary control action. These time dependent kicks will stabilize the system around the selected periodic orbit.

4.5.4 Testing the OGY control Method Developed

The control procedure discussed above was applied to control the chaotic behavior in two typical chaotic systems, the logistic map defined as:

$$x_{n+1} = x_n \lambda (1 - x_n) \tag{20}$$

and the Henon map defined as:

$$\begin{aligned} x_{n+1} &= 1 - \alpha x_n^{\ 2} + y_n \\ y_{n+1} &= \beta x_n \end{aligned} \tag{21}$$

The chaotic behavior in the two systems was stabilized to several periodic orbits. Figure 31 shows the results of applying and then removing the control on the logistic map ($\lambda = 3.825$) to stabilize it to a period-1 orbit. The computed parameters for this case are as follows:

$x_F = 0.738504$

$$g = \begin{bmatrix} 0.059 \\ 0.059 \end{bmatrix}$$

$$M = \begin{bmatrix} -1.8252 & 0.0001 \\ 1 & 0 \end{bmatrix}$$

$\lambda_u = -1.8253$

$\lambda_s = 4.3315e-005$

$$f_u = \begin{bmatrix} -1.1402143 \\ 0.0000493 \end{bmatrix}$$

$C = -9.6$

$\delta\zeta_m = 0.01822$

The control was applied at the iterate 500 and then removed at the iterate 2500. We notice that the effect of the controller did not appear right after it was applied, the reason is that we have to wait for the system iterates to fall within the vicinity of the fixed point to be stabilized before we actually apply the control kicks. It is also clear that once the control was removed the chaotic behavior has returned. This is due to noise and inaccuracies in the computation as mentioned above. In addition, the system actually needs an infinite time until it becomes exactly on the periodic orbit. In theory, once it is there the iterates should stay on the orbit unless perturbed. Figure 32 shows the control action applied as a function of time. It is clear that after few iterates from the control start, the kicks necessary to keep the system stabilized at that orbit are very small, which makes this control algorithm very inexpensive and easy to implement in many practical systems. Figures 33 and 34 show how the OGY method was used to stabilize the logistic map to period-2 and period-5 respectively. Also, figures 35 and 36 show how it was used to stabilize the Henon map to period-1 and period-2 respectively.

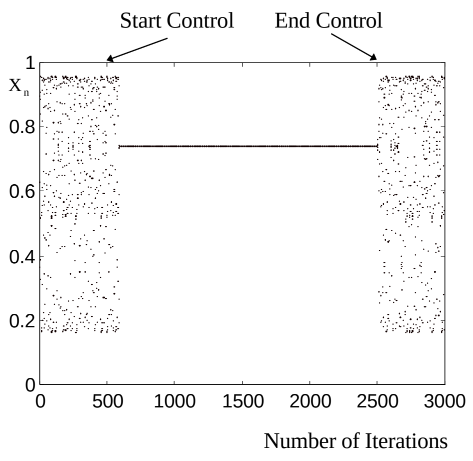


Figure 31 Controlling the logistic map to a fixed point

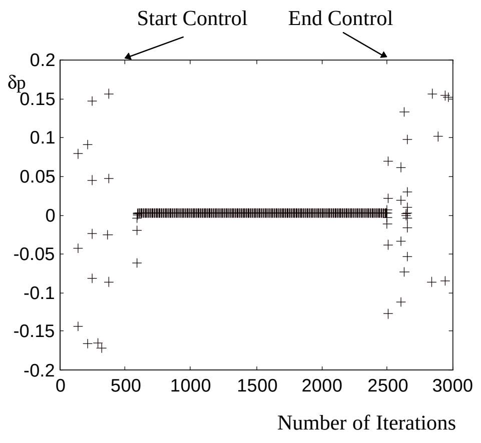


Figure 32 The control perturbations during the control shown in Figure 31.

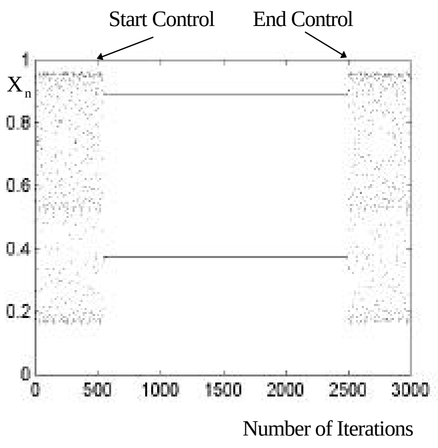


Figure 33 Controlling the logistic map to period two.

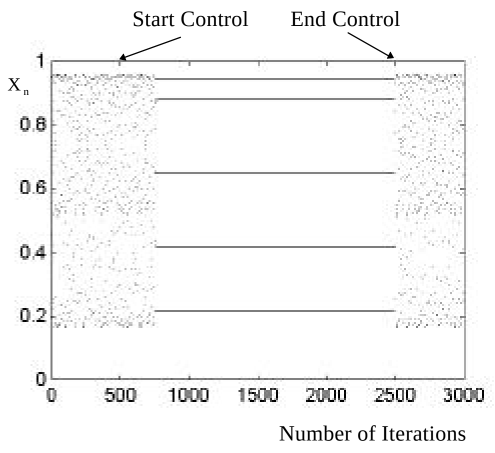


Figure 34 Controlling the logistic map to period five.

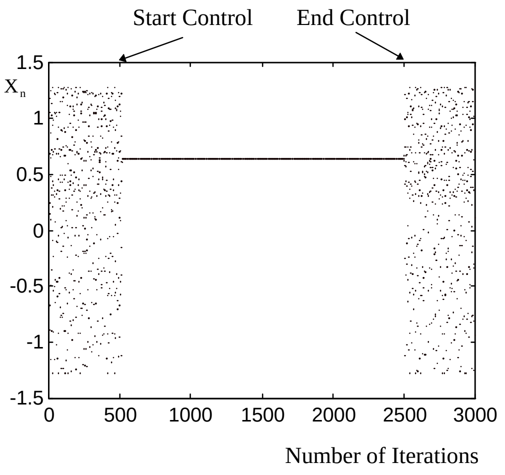


Figure 35 Controlling the Henon map to period one

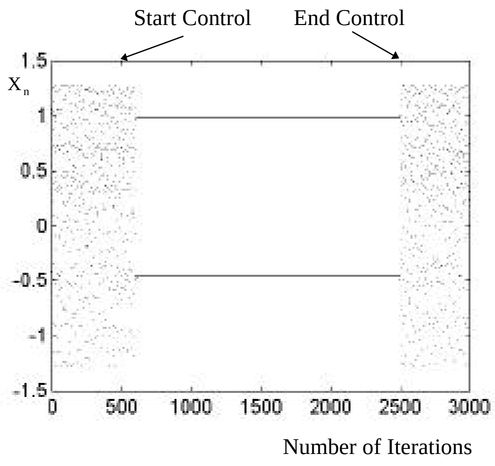


Figure 36 Controlling the Henon map to period two

5. The Hardware Implementation of the Proposed System

We have developed techniques to control the chaotic behavior in the Fluidized Bed (FBC) Systems using Artificial Neural Networks (ANNs), and the OGY technique. We have also discussed a Data Acquisition Board setup that will enable communication between our programs and external systems. Communication is planned to be enabled in both ways to deliver feedback signals from a system to the control programs in one way, and the control signals from the control programs to the controlled system in the other way. On the other hand, since most of our programs are PC based, they have to follow the revolutionary progress in the PC technology. Our programs for the DSI neural network were developed in the DOS environment using an early version of Microsoft C compiler. For those programs to meet the current needs of most PC users, we are working on converting those programs to the Windows environment, using a very advanced and up to date C++ compiler. This compiler is known as the Microsoft Visual C++ Version 4.0. This compiler enables the implementation of very professional and sophisticated Windows 95, 32 bit applications. It also allows a simple utilization of the Object Oriented Programming (OOP) techniques, and lots of powerful graphical and communication tools known as the Microsoft Foundation Classes (MFC). This compiler also allows creating Dynamic Link Libraries (DLLs) that can be liked to other Windows programs. These two main aspects, the computer-system interface and the DOS-Windows migration will give our programs a leap frog towards their real implementation.

5.1 How the Dynamic System Imitator (DSI) Programs Will be Modified

The DSI controller programs are simulations to the DSI recurrent and dynamic nature that is capable of creating very sophisticated time behaviors [14, 19, 20]. This enables the DSI to model a variety of dynamic systems and non-linear controllers. During training, the DSI has to monitor the behavior of the controlled system and work on creating the necessary control signal to modify the system behavior and bring it to a pre-specified time behavior. In the time being the DSI simulation programs deal with models of the controlled system instead of the real system. The system models are supplied through a Runge Kutta fourth order simulation routines that solve a coupled system of ordinary differential equations. Outputs from those simulation routines are passed to the DSI network and the output of the DSI network is also passed as a control action to the simulation model. The desired output of the system is supplied to the network through other set of routines that solve some pre-specified functions. In reality, the network has to deal with the actual system, during its training and recall modes. Therefore, modification need to be made to the DSI programs to communicate with the controlled system through computer interface that performs the necessary Analog to Digital (A/D) and Digital to Analog (D/A) conversions. On the other hand almost all parts of the DSI programs will be converted in the context of the Object Oriented Programming, Microsoft Foundation classes, and Microsoft Visual C++ 4.0 compiler [34]. New classes and objects will be created to represent the network nodes, variables, interfaces, and connections. User interface tools that were not available in the old programs will be developed, such as menus, icons, pop up menus, radio buttons, and mouse handlers. Also all graphics concepts will be transferred to the Application Programming Interface (API) tools that are controlled by the Microsoft Foundation Classes.

5.2 The Data Acquisition System

An example Data acquisition system using a portable PC is illustrated in Figure 37. It utilizes a National Instrument DAQ 1200 PC MCIA card, which is a multifunction I/O DAQ unit that communicates with a PC through the parallel port on IBM PC/XT/AT and compatibles, with a maximum sampling rate of 100 KHz. The DAQ-1200 has 12-bit Analog to Digital Converters (DAC) with eight analog inputs, configurable as eight single ended or four differential inputs [35]. It also have two 12-bit, double buffered Digital to Analog Converters (DACs), and programmable gains of 1, 2, 5, 10, 20, 50, or 100.

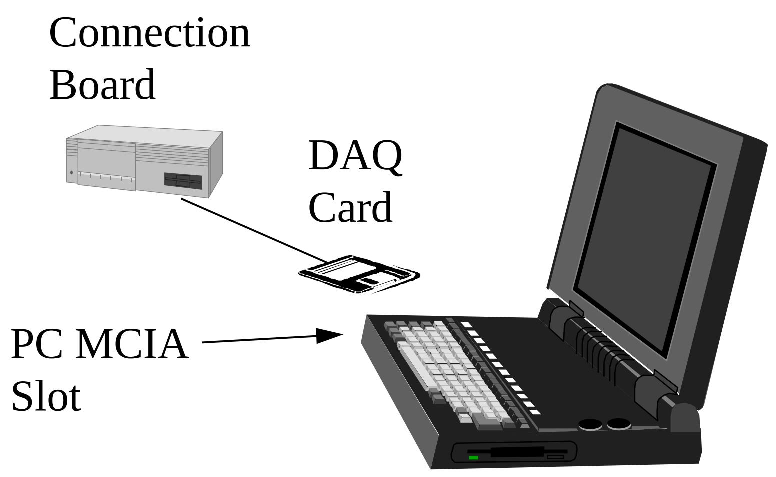


Figure 37 An Example data acquisition system using a portable PC.

6. Conclusions

In this report, analysis to data measured from an FBC facility has been presented. The purpose of this analysis is to build basis for a neuro-controller design that can be used to control some undesirable abnormal behavior in FBC systems. This analysis shows that both the normal and abnormal behavior represented in the data available are chaotic. The system's normal and abnormal attractors have fractal dimensions, some positive Lyapunov exponents and positive Kolmogorov entropy. However, the correlation dimension of the abnormal case is much higher than the normal case. This indicates that when the system switches to its abnormal situation it suffers a very complex behavior, most likely belongs to a higher order chaos. An appropriate controller is desired to control the system abnormal chaotic behavior to its normal chaotic behavior. A general method to monitor and control the chaotic behavior in an FBC system has been outlined. A recurrent neural network called the Dynamic System Imitator (DSI) was adopted. The only missing chain to test the proposed method is an appropriate non-linear model that will describe the FBC in different modes. Finding this model, and testing the proposed monitoring and control method, will be our main focus of future work in this project.

We presented the application of a special type of the DSI, for chaotic system control. The proposed methodology was applied to the Lorenz system. Both the system state point control and the parameter perturbation control strategies were explored. The control action generated by the neural network was fed in several locations in the system of equations to demonstrate the ability of the algorithm to perform in several configurations. This flexibility makes the algorithm more applicable to real systems. In all cases, the DSI was able to control the Lorenz chaotic behavior to a stable fixed point or to a stable periodic orbit, depending on the way it was trained. We believe this technique can be further applied to many other dynamic systems.

A dynamic neural network based model for chaotic time series has been developed. A one step predictor model was used to iteratively generate chaotic time series, using the DSI neural network. The DSI has distinguishable dynamic features due to its special architecture. The DSI time behavior depends on its initial conditions. After training, the DSI was able to generate the chaotic time series in all test cases. For a short period of time, it was able to generate the same time behavior of the training time series if started with the same initial conditions and the first initial points of the time series. Furthermore, it was able to track the system attractor to which the training time series belongs, for any period of time, for any initial conditions and any initial points, in all test cases. The only case that fails is the zero initial conditions together with zero starting points, which leads to a zero solution. This methodology was applied to three known chaotic models, the logistic, Henon and cubic maps, in addition to one experimental time series taken from an FBC system. The correlation dimension and the Kolmogorov entropy for the actual and DSI network synthetic data were computed and compared. There is a very good match between the actual and synthetic time series parameters in all cases which indicates that the DSI was able to learn the dynamics in those chaotic time series to a very good extent.

We have presented a systematic way to compute the parameters of the OGY technique from time series measurements, and to implement this method to control the chaotic behavior in a chaotic system, and stabilize it into one of its naturally unstable periodic orbits. We have developed a technique to compute the linear map that describes the system behavior on a two dimensional map around a fixed point, which is essential to the OGY method, using autoregressive modeling. We have shown a detailed procedure to localize and separate a naturally unstable periodic orbit in a chaotic time series. We have also explained how to experimentally compute the rate of change of the fixed point corresponding to the periodic orbit to be controlled with respect to a control parameter. Finally, we have applied the techniques developed to control the chaotic behavior in some typical chaotic systems, such as the logistic and the Henon maps, and stabilize it to one of its naturally unstable periodic orbits. Stabilizing such chaotic systems to several example periods were presented. In the control method described, we have to wait for the system until its iterates fall into a close vicinity of the fixed point corresponding to the orbit to be stabilized, and then apply time dependent sequence of kicks that stabilizes the system on that naturally unstable orbit. From the results summarized above, it is clear that the control actions necessary to stabilize the system become very small and almost negligible after few iterates from the start of the control, which makes this method very inexpensive, and easy to implement in many practical systems. However, the control kicks has to be maintained or control will be lost, and the system behavior will go back to its naturally chaotic behavior.

Based on the results of FBC data analysis described above, we have developed some chaotic system control techniques and applied them to some typical chaotic systems. However, we were not able to test those techniques on an FBC system, because there is no suitable model available. In the same time, it was not possible to build such a model from the available data, because they are all one dimensional single variable measurements which is not enough to model the input output relationships required to test our control techniques. However, we were able to find a prediction model for those pressure measurement time series that can be used for on line monitoring of the process. Two main types of chaotic system controllers were developed. The first is based on a special type of a recurrent neural network developed, known as the Dynamic System Imitator (DSI), which is specially designed to model a wide variety of dynamic systems. The second technique is based on the chaotic system control method known as the OGY, developed by Edward Ott, Yorke and Grebogi in 1990. We have developed an autoregressive algorithm to estimate a system map matrix necessary for the OGY method. The OGY control parameters can be estimated from one dimensional measurements from the system to be controlled. However, also the sensitivity of the control parameters with respect to the control variables in a form of derivatives has to be accurately measured. The OGY technique is designed such that to control a chaotic system in the vicinity of a periodic orbit or a fixed point and force it to stabilize to that periodic orbit or fixed point.

Other types of measurements are certainly necessary to build a simple model that well describes the chaotic behavior of the FBC system at both normal and abnormal operation. On the other hand, this model has to describe the input output relationships between the control variables and the other variables and parameters in the system. In addition, this model has to describe the system behavior under both transient and steady state conditions, and the system transients from normal to abnormal and visa versa. This type of model is necessary to train and test the neuro-controller and to test the OGY controller. Other specialized measurements are necessary to compute the derivative of a control parameter with respect to a control input around the periodic orbit or the fixed point to be controlled using the OGY technique.

7. Future Work

We summarize the goals and objectives of the next phase of this project and the proposed fluidized bed experimental setup in the following points:

1. Develop and execute an experimental setup that will enable collecting the data necessary to develop a simple non-linear model for the FBC system, and further develop the chaotic system control techniques developed in the first phase of the experiment, including neural network control, and the conventional OGY method.
2. Build a simple and sufficiently fast computer model for the FBC system that can be used to train the neural controller and test its performance. This model will be also used to test the performance of the OGY controller.
3. Train and test the neural network based control techniques for the purpose of chaotic system control in general and FBC system control in particular.
4. Design and test a methodology to control an FBC system based on the OGY technique, and another methodology based on a combination of both neural and OGY controllers.

Taking all previous considerations and the existing setup of the cold FBC experiment shown in Figure 1 into account, we propose a setup that will enable the test procedure under the following conditions:

1. System pressure at different levels, the operating pressure, the air flow rate, and the cork flow rate, in addition to any other contributing inputs, parameters or conditions, such as humidity or temperature, have to be measured simultaneously and recorded carefully at all times in all tests. This is to simplify the utilization of such data and parameters in the following analysis, modeling and parameter estimation procedure.
2. Input/output tests will be performed by doing the measurements during input disturbances following some standard behaviors such as ramp, step, impulse, square, sinusoidal, triangular, or random. The only known effective input to the system at the time being is the air flow. However, other inputs might be considered, such as the cork feed rate and the working pressure. The system outputs in the case are the differential pressures at different levels.
3. Steady state tests will be performed for enough periods of times for both normal and abnormal conditions. More specifically, the system inputs will be adjusted to allow certain system condition either normal or abnormal, then they will be held constant at that level for certain time during which the system variables will be recorded.
4. Transients from normal to abnormal situations and the opposite will be created and monitored through measurements. First, certain condition will be created (normal or abnormal), then the disturbance necessary to kick the system to the opposite condition

(abnormal or normal) will be applied. The system variables and parameters will be recorded during this process.

5. Steady state measurements will be performed at different values of operating parameters, such as the working pressure. Steady state recording will be performed at some value for that parameter, and then the measurement will be repeated with that parameter incremented up and down around its original value.

8. Student Participation

The following are brief portrait for students under Tennessee State University (TSU) graduate program who were supported under this project to do the research required for their master thesis. In the same time they contributed to the overall objective of the project:

Mr. James Osa, He has finishing his Master Thesis, and graduated summer 1996. 96. His master thesis involvesd investigating using some static neural network techniques to control the chaotic behavior in some typical chaotic system.

Mr. Jihad Ababneh, He worked on his Master Thesis while supported by this grant. He has been involved on research related to the using the OGY (Ott, Grebogi, and York) method to control the chaotic behavior using small perturbations. He was the first author on one of our publications to the SSSE. He is now investigating using neural network techniques to estimate the OGY parameters from a chaotic time series.

9. References

1. E. Ott, “Controlling chaos,” Physics Today, pp 34-40, May 1995.
2. J. Singer, “Controlling a Chaotic System,” Physical Review Letters, Vol. 66, pp 1230-1232, 1991.
3. T. Shinbrot, “Using Small Perturbations to Control Chaos,” Nature, Vol. 363, pp 411-417, 3 June, 1993.
4. E. Ott, “Controlling Chaos,” Physical Review Letters, Vol. 64, No 11, pp 1196-1199, 12 March, 1990.
5. W. L. Ditto, “Experimental Control of Chaos,” Physical Review Letters, Vol. 65, No 26, pp 3211-3214, 24 December, 1990.
6. R. Hunt, “Stabilizing High-Period Orbits in a Chaotic System: The Diode Resonator,” Physical Review Letters, Vol. 67, No 15, pp 1953-1955, October 1991.
7. V. Petrov, “Tracking Unstable Periodic Orbits in the Belousov-Zhabotinsky Reaction,” Physical Review Letters, Vol. 72, No 18, pp 2955-2958, May 1994
8. S. Hayes, “Experimental Control of Chaos for Communication,” Physical Review Letters, Vol. 73, No 13, pp 1781-1784, September 1994.
9. R. Roy, “Dynamical Control of a Chaotic Laser: Experimental Stabilization of a Globally Coupled System,” Physical Review Letters, Vol. 68, No 9, pp 1259-1262, March 1992.
10. A. Garfinkel, “Controlling Cardiac Chaos,” Science, Vol. 257, pp 1230-1235, 1992.
11. Fuller, “Interpretation of Pilot Scale, Fluidized Bed Behavior Using Chaotic Time Series Analysis,” 12th international FBC Conference, LA Jolla, CA, May, 1993.
12. S. Daw, “Characterization of Voidage and Pressure Signals from Fluidized Beds, Using Deterministic Chaos Theory,” Fluidized Bed Combustion, ASME Conference 1991, pp 777-785.
13. S. Halow, “Characterizing Fluidized Bed Behavior, By Decomposition of Chaotic Phase Space Trajectories,” 1993 Annual AIChE Meeting, St. Louis, MO, November, 1993.
14. Magdi A. Essawy, Robert E. Uhrig, *" The Dynamic System Imitator: A New Approach for a Dynamic Neural Network Architecture"*, the 9th Power Plant Dynamics, Control & Testing Symposium, Knoxville, Tennessee, May 24-26 1995.
15. E. Jackson, “The Entrainment and Migration Controls of Multiple Attractor Systems,” Physics Letters A, Vol. 151, No 9, pp 478-484.
16. M. Ding, C. Grebogi, E. Ott, T. Sauer, J. A. Yorke, “Plateau Onset for Correlation Dimension: When Does It Occur ?”, Physics Review Letters 70, 3872, 1993.
17. Peter Grassberger, “Estimation of the Kolmogorov Entropy from a chaotic Signal”, Physical Review A, Vol. 28, No. 4, pp 2591-2593, October, 1983.
18. M. Sano, “Measurement of the Lyapunov Exponents from a Chaotic Series”, Physical Review Letters, Vol. 55, No. 10, pp 1082-1085, September, 1983.
19. Magdi A. Essawy, “Artificial Neural Networks for System Modeling Monitoring and Control,” Ph.D. Dissertation, University of Tennessee, Knoxville, July 1995.
20. Magdi A. Essawy, Robert E. Uhrig, “The Dynamic System Imitator: A New Approach for a Dynamic Neural Network Architecture,” the 9th Power Plant Dynamics, Control & Testing Symposium, Knoxville, Tennessee, May 24-26 1995.
21. Denny Gulick, “Encounters With Chaos,” McGraw-Hill, Inc., 1992.
22. J.D. Farmer, J. Sidorowich, “Predicting Chaotic Time Series, ” Phys. Rev. Lett. 59, pp 845, 1987.
23. M. Casdagli, “Non-linear Prediction of Chaotic Time Series,” Physica D 35, pp 335, 1989.

24. G. Sugihara, R. M. May, “Non-linear Forecasting as a way of Distinguishing Chaos from Measurement Error in Time Series,” Nature 344, pp 734, 1990.
25. J. C.Principe, J-M Kuo, “Dynamic Modeling of Chaotic time Series with Neural Networks,” Advances in Neural Information Processing Systems 7, Ed. Tesauro, Touretzky, Leen, pp 311-318, 1995.
26. Magdi A. Essawy, “Artificial Neural Networks for System Modeling Monitoring and Control,” Ph.D. Dissertation, University of Tennessee, Knoxville, July 1995.
27. Magdi A. Essawy, Robert E. Uhrig, “The Dynamic System Imitator: A New Approach for a Dynamic Neural Network Architecture,” the 9th Power Plant Dynamics, Control & Testing Symposium, Knoxville, Tennessee, May 24-26 1995.
28. Magdi A. Essawy, Mohammad Bodruzzaman, “Using A Recurrent Neural Network for Chaotic Behavior Control,” World Congress on Neural Networks '96, Sep. 15-20 1996.
29. Magdi A. Essawy, Mohammad Bodruzzaman, “Recurrent Neural Network-Based Monitoring and Control of Chaotic Fluidized Bed Combustion Systems,” the 4th Annual HBCU Private Sector Energy Research and Development Technology Transfer Symposium, Greensboro, NC, April 2-3 1996.
30. Fuller, “Interpretation of Pilot Scale, Fluidized Bed Behavior Using Chaotic Time Series Analysis,” 12th international FBC Conference, LA Jolla, CA, May, 1993.
31. S. Daw, “Characterization of Voidage and Pressure Signals from Fluidized Beds, Using Deterministic Chaos Theory,” Fluidized Bed Combustion, ASME Conference 1991, pp 777-785.
32. S. Halow, “Characterizing Fluidized Bed Behavior, By Decomposition of Chaotic Phase Space Trajectories,” 1993 Annual AIChE Meeting, St. Louis, MO, November, 1993.
33. U. Dressler, “Controlling Chaos Using Time Delay Coordinates” Physical Review Letters, Vol. 68, No 1, pp 1-4, January 1992.
34. Ivor Horton, "Visual C++ 4.0, The Complete Tutorial to C++, OOP and Windows Programming," Wrox Press, 1996.
35. National Instruments, "1996 Instrumentation Reference and Catalog, Test and Measurement, Industrial Automation".

10. Appendixes

I.

COMPUTER PROGRAMS

```
%
%
%                    Program name:  Lya5.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%          This program was written under the DOE contract # DE-FG22-94MT94015,
%          period July, 1995-December 1996.
%
%                              Dr. Magdi A. Essawy, Postdoctoral Research Associate
%                              Dr. Mohammad Bodruzzaman, Principal Investigator
%                              Department of Electrical and Computer Engineering
%                              College of Engineering and Technology
%                              Tennessee State University, Nashville TN 37209
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program is to compute the Lyapunov exponents spectrum for a time series embedded
% in the matrix "x" that sould be resident in the matlab environment when the program is running.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dim=3;                         % the embedding dimension
tau=.01;             % the sampling time
tm=5;                          % the traveling time
ors=1;                         % the starting point
ori=5;                         % the step
orf=20;              % the final point for the spectrum
rec=[];
rec1=[];
eps=1.0;                       % the size of the epsilon ball it needs to be about 2.5 % of the maximum extension
of the                         % attractor
count=0;
skip=0;                        % the number of points to be skipped in the beginng

lam=zeros(1,dim);
max=size(x,1)-tm;              % the matrix x is the embedding matrix created by the embedd program
a=randn(dim);
be=eye(dim);
for orp=ors:ori:orf
maty=[];
%bn=zeros(dim);
aa=a*be;
for k=1:dim
          rr(k,k)=norm(aa(1:dim,k));
          qq(1:dim,k)=aa(1:dim,k)/rr(k,k);
for j=k+1:dim
          rr(k,j)=qq(1:dim,k)'*aa(1:dim,j);
          aa(1:dim,j)=aa(1:dim,j)-qq(1:dim,k)*rr(k,j);
end
```

```
end
be=aa;
mat=x(orp,:);                    % the matrix x is the embedding matrix created by the embedd program
ny=0;
nmy=[];
for i=1:max
mat1=x(i,:);
dif=(mat1-mat);
rrr=norm(dif);
if rrr<eps
maty=[maty dif'];
ny=ny+1;
nmy=[nmy i];
end
if ny>20
break;
end
end
orp

if ny>dim
matz=[];
matm=x(orp+tm,:);
ny
for j=1:ny
matm1=x(nmy(j)+tm,:);
dif=matm1-matm;
matz=[matz dif'];
end
v=1/ny*(maty*maty');
c=1/ny*(matz*maty');
a=c*inv(v);
for i=1:dim
lam(i)=lam(i)+log(norm(a*be(:,i)));
end
if(orp-ors) > skip
ppp=lam*ori/(orp-ors-count-skip+1)/tm/tau;
rec=[rec ppp'];
rec1=[rec1 orp];
end
else
count=count+1;
end
end
plot(rec1,rec);
count
end
```

```
%
%
%                  Program name:  corr2.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%        This program was written under the DOE contract # DE-FG22-94MT94015,
%        period July, 1995-December 1996.
%
%                             Dr. Magdi A. Essawy, Postdoctoral Research Associate
%                             Dr. Mohammad Bodruzzaman, Principal Investigator
%                             Department of Electrical and Computer Engineering
%                             College of Engineering and Technology
%                             Tennessee State University, Nashville TN 37209
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program is to to compute the correlation integral points for a time series contained in the
%  matrix "inp" that sould be resident in the matlab environment when the program is running.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%

m=1;                          // embedding delay
mdim=2;                       // minimum embedding dimension
idim=2;                       // increment for embedding dimension
xdim=20;                      // maximum embedding dimension
mneps=.005;                   // minimum ball size
ieps=1.5;                     // ball size increment factor
mxeps=10;                     // maximum ball size
veps=[];
tn=[];
tnp=[];

eps=mneps;
while eps<=mxeps
veps=[veps eps];
eps=ieps*eps;
end
neps=size(veps,2);
```

```
tn=zeros((xdim-mdim)/idim+1,neps);
ind=1;
for dim=mdim:idim:xdim
max=size(inp,1)-(dim-1)*m;
xx=[];
xa=[];
for j=1:dim
xa=[xa inp(1+(j-1)*m:max+(j-1)*m)];
end
xx=xa';
N=size(xx,2)
N=N-1;
for i=1:N
y=[];
dy=[];
p=[];
y=xx(:,i+1:N+1)-xx(:,i)*ones(1,N+1-i);
dy=sqrt(sum(y.^2));
for j=1:neps
p=find(dy<veps(j));
tn(ind,j)=tn(ind,j)+length(p);
end
end
ind=ind+1;
end
tnp=log(tn/(N^2));
vepsp=log(veps);
plot(vepsp,tnp);
hold on;
plot(vepsp,tnp,'+');
hold off;
```

```
%
%
%                    Program name:  Cplo.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%          This program was written under the DOE contract # DE-FG22-94MT94015,
%          period July, 1995-December 1996.
%
%                              Dr. Magdi A. Essawy, Postdoctoral Research Associate
%                              Dr. Mohammad Bodruzzaman, Principal Investigator
%                              Department of Electrical and Computer Engineering
%                              College of Engineering and Technology
%                              Tennessee State University, Nashville TN 37209
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program is to compute the linear regression fit for the correlation integral results computed
%  by the "corr2.m" program listed above
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


mdim=2;                             % minimum embedding dimension
idim=2;                             % embedding dimension inccrement
xdim=20;                            % maximum embedding dimension
ddd=[];
a=[];
b=[];

tnd=tnp';
veps=vepsp';
```

```
ind=1;
for i=mdim:idim:xdim
ddd=[ddd i];
s=ii(ind,2)-ii(ind,1)+1;
sx=sum(veps(ii(ind,1):ii(ind,2)));
sxx=sum(veps(ii(ind,1):ii(ind,2)).^2);
sy=sum(tnd(ii(ind,1):ii(ind,2),ind));
sxy=sum(veps(ii(ind,1):ii(ind,2)).*tnd(ii(ind,1):ii(ind,2),ind));
dt=s*sxx-sx^2;
a(ind)=(sxx*sy-sx*sxy)/dt;
b(ind)=(s*sxy-sx*sy)/dt;
ind=ind+1;
end
ttt=figure;
plot(ddd,b,'+');
hold on
plot(ddd,b);
xlabel('embeding dimension');
ylabel('correlation dimension');
hold off


%
%
%                       Program name:   kento.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%          This program was written under the DOE contract # DE-FG22-94MT94015,
%          period July, 1995-December 1996.
%
%                                   Dr. Magdi A. Essawy, Postdoctoral Research Associate
%                                   Dr. Mohammad Bodruzzaman, Principal Investigator
%                                   Department of Electrical and Computer Engineering
%                                   College of Engineering and Technology
%                                   Tennessee State University, Nashville TN 37209
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program is to compute the Kolmogorove Entropy for the time series represented by the
%  linear regression fit constants computed by the "cplo.m" program listed above.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


ent=[];
m=1
ind=1;
xd=-4;
for i=mdim:idim:xdim-1
```

```
ent=[ent (1/(m*idim)*(a(ind)+b(ind)*xd-a(ind+1)-b(ind+1)*xd))];
ind=ind+1;
end
dd=ddd(1:(xdim-mdim)/idim);
tt=figure
plot(dd,ent);
hold on;
plot(dd,ent,'+');
hold off;


/*
```

```
%
%
%                Program name:  news.c
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%       This program was written under the DOE contract # DE-FG22-94MT94015,
%       period July, 1995-December 1996.
%
%                       Dr. Magdi A. Essawy, Postdoctoral Research Associate
%                       Dr. Mohammad Bodruzzaman, Principal Investigator
%                       Department of Electrical and Computer Engineering
%                       College of Engineering and Technology
%                       Tennessee State University, Nashville TN 37209
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       This is a C program to simulate the Dynamic System Imitator (DSI) neural network
%       in order to perform time series prediction.
%       The program contains both training and recall algorithms
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
*/

#include "stdio.h"
#include "stdlib.h"
#include "graph.h"
#include "math.h"
#include "conio.h"
#include "nr.h"
#include "nrutil.h"
#define dd 1
#define ord 4
#define mum 200
#define pmum 200
#define ccc 1
#define mmp 33
#define nnp 32
#define imp 9
#define inp 8
#define dddt 0.01900

double y=0,xp=0,enrg0;
float c1,c2,c3,c4,c5,c6;
float flag=0;

FILE *af,*bf,*az,*bz;
ord2=ord*ord;
int im=0,colf=1,colt=14,coll=15,colln=2,coltn=15,nplot=0;
float *op,*dop,*tmop,*tmdop,*inpm,*desop,*top,*err,*xx0;
float **w,**d;
/*float b[4]; */
int lpos[2]={0,0};

char fname[15]="inp.dat";
char sname[15]="opt.dat";
char oname[15]="sav.dat";
char rname[15]="rec.dat";
char cname[15]="comp.dat";
char lname[15]="out.dat";
char zname[15]="inc.dat";
char ename[15]="err.dat";
char mname[15]="info.dat";

double func(float x[]);
double funk(float x[]);
void title();
void perr(char *dc);
void mfopen(char *pp);
void fmfopen(char *pp);
void rein(void);
```

```
/*void adjust2(int max,float **mat,int k);*/
/*void adjustm(int max,int **mot);*/
void recall(char *ocp);
void outside(int lim);
void outsid(int lim);
void start(int *ms);
void plot(int m, int n,float cp[],int nom);
void plotv(int m, int n);
void plote(int m, int n,int * icp,double y);
void myinput(char *np);
void mydoput(char *np);
void final();
void myout(char *onp);
void random(int m,int n,float **fp);
void randomb(int m,float *lp);
void incd(float x[],float y[]);
void incsav(float x[],float y[]);
void inscr(void);
void calcdop(register int i,float dt);
/*void setc();*/
double rmserr(int num);
double rms(int num);
float calclop(int i);

void main(void)
{
register int kn=0,jm,km=0,in,j;
int count=0,fall,ndim=nnp,idim=inp;
int maxi=0,iiix;
float *incond,*dincon,*xx,*xy,*yy,*yx,*psum,*ptry,**pp,**pp0,**px;
int mm=0,k=0, sss=0, mslotn=0,nfunc,nfunk;

char arg,rf,rs,rt;
float  sum=0;
char buffer[10];

start(&mslotn);

ptry=vector((long int)0,(long int)ndim-1);
psum=vector((long int)0,(long int)ndim-1);
incond=vector((long int)0,(long int)ord-1);
dincon=vector((long int)0,(long int)ord-1);
inpm=vector((long int)0,(long int)mum-1);
desop=vector((long int)0,(long int)mum-1);
w=matrix((long int)0,(long int)ord-1,(long int)0,(long int)ord-1);
d=matrix((long int)0,(long int)ord-1,(long int)0,(long int)ord-1);
pp=matrix((long int)0,(long int)mmp-1,(long int)0,(long int)nnp-1);
pp0=matrix((long int)0,(long int)mmp-1,(long int)0,(long int)nnp-1);
px=matrix((long int)0,(long int)imp-1,(long int)0,(long int)inp-1);
op=vector((long int)0,(long int)ord-1);
```

```
dop=vector((long int)0,(long int)ord-1);;
tmop=vector((long int)0,(long int)ord-1);
tmdop=vector((long int)0,(long int)ord-1);
top=vector((long int)0,(long int)mum-1);
err=vector((long int)0,(long int)mum-1);
xx=vector((long int)0,(long int)nnp-1);
xx0=vector((long int)0,(long int)nnp-1);
xy=vector((long int)0,(long int)inp-1);
yy=vector((long int)0,(long int)mmp-1);
yx=vector((long int)0,(long int)imp-1);

for(in=0;in<mmp;in++){
  for(j=0;j<nnp;j++)
             pp[in][j]=(in == (j+1) ? 1.0 : 0.0);
}


incd(incond,dincon);
/*setc();*/
printf("is this traning (t) ??   or recall (r) ??\n");
arg=getche();
switch(arg){
case 't' :
printf("\nwhat is the max number of iterations...??\n");
scanf("%d",&maxi);
/*printf("enter your choice for ...(dw)...\n");*/
/*scanf("%f",&dw);*/
printf("is this a traning from scratch...?? if not write ...n...\n");
arg=getche();
if(arg != 'n'){
/*randomb(b);*/
random(ord,ord,w);
random(ord,ord,d);
k=0;

for(in=0;in<4;in++){
 for(j=0;j<4;j++){
pp[0][k]=w[in][j];
pp[0][k+ord*ord]=d[in][j];
k++;
}
}

}
else {
recall(oname);
k=0;
for(in=0;in<4;in++){
 for(j=0;j<4;j++){
pp[0][k]=w[in][j];
```

```
pp[0][k+ord*ord]=d[in][j];
k++;
}
}
}

myinput(fname);
mydoput(sname);
inscr();
title();
plotv(335,340);

for(in=0;in<mmp;in++){
  for(j=0;j<nnp;j++)
              pp0[in][j]=pp[in][j];
}

for(in=0;in<imp;in++){
  for(j=0;j<inp;j++)
              px[in][j]=(in == (j+1) ? 1.0 : 0.0);  ;

}

for(km=0;km<ord;km++){
tmop[km]=incond[km];
tmdop[km]=dincon[km];
}

rein();

for(jm=0;jm<maxi;jm++){
fseek(az,0,SEEK_SET);
fseek(bz,0,SEEK_SET);
_settextposition(15,73);
sprintf(buffer,"%d",jm+1);
_outtext(buffer);
/*if(jm>(maxi-2))mslotn=1;*/

iiix=0;
for(im=0;im<mslotn;im++){

enrg0=0;
 for(km=0;km<mum;km++){
if(fread(&inpm[km],sizeof inpm[0],1,az)!=1){
printf("read error");
exit(1);
}
if(fread(&desop[km],sizeof desop[0],1,bz)!=1){
printf("read error");
exit(1);
}
enrg0 +=desop[km]*desop[km];
}

rein();
plot(70,135,inpm,pmum);
plot(70,340,desop,pmum);
outside(mum);
y=rmserr(mum);

plot(335,135,top,pmum);
plote(335,340,lpos,y);

/*for(kn=0;kn<3;kn++){*/

nplot=0;
for(in=0;in<mmp;in++){
  for(j=0;j<nnp;j++)
              xx[j]=pp[in][j];
              yy[in]=func(xx);
}
/*nfunc=99.;
_settextposition(0,0);
printf(" aa cc bb xx  %f",ftol);*/
amoeba(pp,yy,ndim,func,&nfunc,psum,ptry);

for(j=0;j<nnp;j++) pp0[iiix][j]=xx0[j];

if(iiix>30)iiix=0;
for(in=0;in<mmp;in++){
  for(j=0;j<nnp;j++)
              pp[in][j]=pp0[in][j];
}

/*
k +=10;
adjust2(ord,w,k);
k +=10;
adjust2(ord,d,k);
k +=10;
if(y<.2){
adjustm(ord,mw);
k=k+1;
adjustm(ord,md);
k=k+1;
}

if(k>(mum-20))k=0;
*/
rein();
```

```
outside(mum);
y=rmserr(mum);
plot(335,135,top,pmum);
plote(335,340,lpos,y);
count=count+1;
if(kbhit())break;
if(y<.001)break;
/*}*/
iiix++;
for(km=0;km<ord;km++){
tmop[km]=op[km];
tmdop[km]=dop[km];
}

if(y<.001)break;
if(kbhit())break;
}

/*mum=2*mum;*/
im=0;

_setcolor(4);
_moveto(0,0);
_rectangle(_GFILLINTERIOR,162,-101,190,-86);
fseek(az,0,SEEK_SET);
fseek(bz,0,SEEK_SET);

for(in=0;in<mum;in++){
if(fread(&inpm[in],sizeof inpm[0],1,az)!=1){
printf("read error");
exit(1);
}
if(fread(&desop[in],sizeof desop[0],1,bz)!=1){
printf("read error");
exit(1);
}

}

for(km=0;km<ord;km++){
tmop[km]=incond[km];
tmdop[km]=dincon[km];
}

rein();
outside(mum);
y=rmserr(mum);
plot(70,135,inpm,pmum);
plot(70,340,desop,pmum);
plot(335,135,top,pmum);
```

```
plote(335,340,lpos,y);

 for(in=0;in<imp;in++){
  for(j=0;j<inp;j++)
              xy[j]=px[in][j];
              yx[in]=funk(xy);
}

amoeba(px,yx,idim,funk,&nfunk,psum,ptry);

for(in=0;in<ord;in++){
incond[in]=tmop[in];
dincon[in]=tmdop[in];
}

if(kbhit())break;
if(y<=.04)break;
}

_settextposition(20,71);
_settextcolor(colt);
_outtext("end of");
_settextposition(21,70);
_outtext("training");
myout(oname);
incsav(incond,dincon);
break;
case 'r' :

printf("\nwhat is the max number of slots (no more than 5) ...??\n");
scanf("%d",&maxi);
inscr();
title();
mfopen(ename);
fmfopen(lname);
plotv(335,340);
myinput(rname);
mydoput(cname);
recall(oname);

for(im=0;im<ord;im++){
op[im]=incond[im];
dop[im]=dincon[im];
}
fseek(az,0,SEEK_SET);
fseek(bz,0,SEEK_SET);

for(im=0;im<maxi;im++){
if(im>=1)
{flag=1.;
```

```
 }
/* else
 { */
 for(km=0;km<mum;km++){
if(fread(&inpm[km],sizeof inpm[0],1,az)!=1){
printf("read error");
exit(1);
}
if(fread(&desop[km],sizeof desop[0],1,bz)!=1){
printf("read error");
exit(1);
}
}
/* }*/

plot(70,135,inpm,pmum);
plot(70,340,desop,pmum);

outsid(mum);

y=rms(mum);
sum=sum+y;
plot(335,135,top,pmum);
plote(335,340,lpos,y);
final();
perr(ename);

fall=getche();
}
sum=sum/maxi;
/*myout(oname);*/
fprintf(bf,"\n\n\n%f\n",sum);
fclose(af);
fclose(bf);
fclose(az);
fclose(bz);
break;
default:
 printf("unrecognized letter");
}

fall=getche();
while(!kbhit());
_setvideomode(_DEFAULTMODE);

free_vector(yx,(long int)0,(long int)imp-1);
free_vector(yy,(long int)0,(long int)mmp-1);
free_vector(xy,(long int)0,(long int)inp-1);
free_vector(xx0,(long int)0,(long int)nnp-1);
free_vector(xx,(long int)0,(long int)nnp-1);
free_vector(err,(long int)0,(long int)mum-1);
free_vector(top,(long int)0,(long int)mum-1);
free_vector(tmdop,(long int)0,(long int)ord-1);
free_vector(tmop,(long int)0,(long int)ord-1);
free_vector(dop,(long int)0,(long int)ord-1);
free_vector(op,(long int)0,(long int)ord-1);
free_matrix(px,(long int)0,(long int)imp-1,(long int)0,(long int)inp-1);
free_matrix(pp0,(long int)0,(long int)mmp-1,(long int)0,(long int)nnp-1);
free_matrix(pp,(long int)0,(long int)mmp-1,(long int)0,(long int)nnp-1);
free_matrix(d,(long int)0,(long int)ord-1,(long int)0,(long int)ord-1);
free_matrix(w,(long int)0,(long int)ord-1,(long int)0,(long int)ord-1);
free_vector(desop,(long int)0,(long int)mum-1);
free_vector(inpm,(long int)0,(long int)mum-1);
free_vector(dincon,(long int)0,(long int)ord-1);
free_vector(incond,(long int)0,(long int)ord-1);
free_vector(psum,(long int)0,(long int)ndim-1);
free_vector(ptry,(long int)0,(long int)ndim-1);

printf("%f   %d\n",y,count);
printf("successful end of program\n");
}

void rein(void){
register int i=0,j=0;
for(i=0;i<ord;i++){
op[i]=tmop[i];
dop[i]=tmdop[i];
}

}

/*void adjustm(int max,int **mot){

register int i=0,j=0;
double y1=0;
float tmp=0;

rein();
outside(mum);
rein();
y=rmserr(mum);
plot(335,135,top,pmum);
plote(335,340,lpos,y);

for(i=0;i<max;i++){
 for(j=0;j<max;j++){
tmp=mot[i][j];
mot[i][j]=!(abs(mot[i][j]));
outside(mum);
rein();
y1=rmserr(mum);
if(y1>y){
mot[i][j]=tmp;
}
else{
y=y1;
}
}
}
}
*/
void mfopen(char *pp){

/* open file for output*/
if((bf=fopen(pp,"w"))==NULL){
 printf("Read error occured");
 exit(1);
}

}
void fmfopen(char *pp){

/* open file for output*/
if((af=fopen(pp,"w"))==NULL){
 printf("Read error occured");
 exit(1);
}

}


void perr(char *dc){


/* write output to file*/
fprintf(bf,"%f\n",y);

}

void final(){
register int i;

/* write output to file*/
for(i=0;i<mum;i++){
fprintf(af,"%f\n",top[i]);
}
}

void outside(int lim){
```

```
register int k,i;
for(k=0;k<lim;k++){
op[0]=inpm[k];
calcdop(0,dddt);

for(i=1;i<ord;i++){
op[i]=calclop(i);
}
for(i=1;i<ord;i++){
calcdop(i,dddt);
}
top[k]=calclop(ord);
err[k]=(top[k]-desop[k]);
}
}

void outsid(int lim){
register int k,i;
for(k=0;k<lim;k++){
if(k<dd){
op[0]=fabs(flag-1)*inpm[k]+flag*top[mum+k-dd];
}
else{
  op[0]=fabs(flag-1)*inpm[k]+flag*top[k-dd];
     }

calcdop(0,dddt);

for(i=1;i<ord;i++){
op[i]=calclop(i);
}
for(i=1;i<ord;i++){
calcdop(i,dddt);
}

top[k]=calclop(ord);
err[k]=(top[k]-desop[k]);
}
}
double rmserr(int num){

register int i;
int j;
double sum;
double enrg;
double x,ee;

j=mum-1;
sum=0.0;
enrg=0.;
```

```
for(i=0;i<num;i++){
sum +=err[i]*err[i];
enrg +=top[i]*top[i];
}
sum=sum/num;
ee=(enrg-enrg0)*(enrg-enrg0);
x=c1*sqrt(sum)+c2*sqrt(ee)/enrg0+c5*sqrt((err[0]*err[0]+.5*err[1]*err[1]+.5*err[2]*err[2]+.25*err[3]*err
[3]+.25*err[j/4]*err[j/4])/5.)+c6*sqrt((0.25*err[j/2]*err[j/2]+.25*err[j/4*3]*err[j/4*3]+err[j]*err[j]+.5*err[
j-1]*err[j-1]+.5*err[j-2]*err[j-2])/5.);
return x;
}
double rms(int num){

register int i;
double sum;
double enrg;
double x,ee;

sum=0.0;
enrg=0.;
for(i=0;i<num;i++){
sum +=err[i]*err[i];
enrg +=top[i]*top[i];
}
sum=sum/num;
x=sqrt(sum);
return x;
}

void inscr(void){

_setvideomode(_VRES16COLOR);
_setbkcolor(_RED);
_setcolor(coll);
}

void random(int m,int n,float **fp){

char r1;
char r2;
register int i,j;


for(i=0;i<m;i++){
 for(j=0;j<n;j++){
srand(1);
r1=rand();
r2=rand();
if(r2==0)r2=1;
fp[i][j]=pow(-1,r1)/((float) r2);
}
}
}

void randomb(int m,float *lp){
char r1;
char r2;

register int i;

for(i=0;i<m;i++){
srand(3);
r1=rand();
r2=rand();
if(r2==0)r2=1;
lp[i]=pow(-1,r1)/((float) r2);
}
}

void myinput(char *np){

register int i;

/* open file for input*/
if((az=fopen(np,"rb"))==NULL){
 printf("Read error occured");
 exit(1);
}
}
void mydoput(char *np){
register int i;

/* open file for input*/
if((bz=fopen(np,"rb"))==NULL){
 printf("Read error occured");
 exit(1);
}
}

void myout(char *onp){

FILE *a;


register int i;
register int j=0;
/* open file for output*/
if((a=fopen(onp,"w"))==NULL){
 printf("Read error occured");
 exit(1);
```

```
}

/*write net to data file
for(i=0;i<ord;i++){
fprintf(a,"%f\n",b[i]);
}*/

for(i=0;i<ord;i++){
for(j=0;j<ord;j++){
fprintf(a,"%f\n",w[i][j]);
fprintf(a,"%f\n",d[i][j]);
}
}
/*for(i=0;i<ord;i++){
fprintf(a,"%f\n",op[i]);
fprintf(a,"%f\n",dop[i]);
fprintf(a,"%f\n",mov[i]);
fprintf(a,"%f\n",movd[i]);
fprintf(a,"%f\n",dev[i]);
fprintf(a,"%f\n",devd[i]);
}                        */

fclose(a);
}

void recall(char *ocp){

FILE *a;
register int i=0;
register int j=0;

/* open file for output*/
if((a=fopen(ocp,"r"))==NULL){
 printf("Read error occured");
 exit(1);
}

/* read net from data file
for(i=0;i<ord;i++){
fscanf(a,"%f\n", &b[i]);
}*/

for(i=0;i<ord;i++){
for(j=0;j<ord;j++){
fscanf(a,"%f\n", &w[i][j]);
fscanf(a,"%f\n", &d[i][j]);
}
}
/*for(i=0;i<ord;i++){
fscanf(a,"%f\n", &op[i]);
fscanf(a,"%f\n", &dop[i]);
fscanf(a,"%f\n", &mov[i]);
fscanf(a,"%f\n", &movd[i]);
fscanf(a,"%f\n", &dev[i]);
fscanf(a,"%f\n", &devd[i]);
}                         */

fclose(a);
}



void plot(int m, int n,float cp[],int nom){
register int i,j;
char buffer[10];
float tim,tim1;

_setcolor(colf);
_setlogorg(m,n);
_settextcolor(coltn);
_rectangle(_GFILLINTERIOR,0,80,ccc*nom,-80);
_settextposition((n/15-6),(m/11+m/100*3+2));
tim=im*nom/300.;
sprintf(buffer,"%.2f",tim);
_outtext(buffer);
_settextposition((n/15-6),(m/11+m/100*3+27));
tim1=(im+1)*nom/300.;
sprintf(buffer,"%.2f",tim1);
_outtext(buffer);
_settextposition(16,64);
sprintf(buffer,"%d",im+1);
_outtext(buffer);
_setcolor(colln);
_moveto(0,0);
_lineto(ccc*nom,0);
_setcolor(coll);
_moveto(0,0);
for(i=0;i<nom;i++){
j=-80*cp[i];
if(abs(j)>80)j=j/abs(j)*80;
_lineto(ccc*i,j);
_moveto(ccc*i,j);
}

}

void plotv(int m, int n){
register int i,j;
```

```
_setcolor(colf);
_setlogorg(m,n);
_rectangle(_GFILLINTERIOR,0,80,ccc*pmum,-80);
_setcolor(colln);
_moveto(0,0);
_lineto(ccc*pmum,0);
_setcolor(coll);
_moveto(0,0);
}



void plote(int m, int n,int * icp,double y){
register int i,j;

_setlogorg(m,n);
_moveto(icp[0],icp[1]);
_setcolor(coll);
i=icp[0]+5;
if(i>ccc*pmum){
i=0;
_setcolor(colf);
plotv(335,340);
_setcolor(coll);
}
j=-(80.0 * y);
if(abs(j)>80)j=j/abs(j)*80;
_lineto(i,j);
icp[0]=i;
icp[1]=j;

}

void title(){


/*_setcolor(13);
_rectangle(_GFILLINTERIOR,10,10,615,465);*/

_settextposition(15,14);
_settextcolor(colt);
_outtext("(a) input");
_settextposition(3,23);
_outtext("time [s]");
_settextposition(16,23);
_outtext("time [s]");
_settextposition(3,55);
_outtext("time [s]");
_settextposition(16,55);
_outtext("slot no");
```

```
_settextposition(7,5);
_outtext("pu V");
_settextposition(7,38);
_outtext("pu V");
_settextposition(20,5);
_outtext("pu V");
_settextposition(28,14);
_outtext("(b) desired output");
_settextposition(15,46);
_outtext("(c) network output");
_settextposition(28,45);
_outtext("(d) rms error history");
_settextposition(14,71);
_outtext("It #");
_settextcolor(coltn);
_settextposition(4,6);
_outtext("+1");
_settextposition(14,6);
_outtext("-1");
_settextposition(17,6);
_outtext("+1");
_settextposition(27,6);
_outtext("-1");
_settextposition(4,39);
_outtext("+1");
_settextposition(14,39);
_outtext("-1");
_settextposition(17,39);
_outtext("+1");
_settextposition(27,39);
_outtext("-1");

}

void start(int *ms){

FILE *df;


/* open file for input*/
if((df=fopen(mname,"r"))==NULL){
 printf("Read error occured");
 exit(1);
}

fscanf(df,"%15s\n",fname);
fscanf(df,"%15s\n",sname);
fscanf(df,"%15s\n",rname);
fscanf(df,"%15s\n",cname);
fscanf(df,"%15s\n",lname);
fscanf(df,"%15s\n",ename);
fscanf(df,"%d\n",ms);
fscanf(df,"%f\n",&c1);
fscanf(df,"%f\n",&c2);
fscanf(df,"%f\n",&c3);
fscanf(df,"%f\n",&c4);
fscanf(df,"%f\n",&c5);
fscanf(df,"%f\n",&c6);


fclose(df);
}

void incd(float x[],float y[]){
FILE *cf;
int in=0;

if((cf=fopen(zname,"r"))==NULL){
 printf("Read error occured");
 exit(1);
}

for(in=0;in<ord;in++){
fscanf(cf,"%f\n",&x[in]);
fscanf(cf,"%f\n",&y[in]);
}
fclose(cf);

}

void incsav(float x[],float y[]){

FILE *cf;
int in=0;

if((cf=fopen(zname,"w"))==NULL){
 printf("Read error occured");
 exit(1);
}

for(in=0;in<ord;in++){
fprintf(cf,"%f\n",x[in]);
fprintf(cf,"%f\n",y[in]);
}
fclose(cf);

}

/*void setc(){
register im,jm;
```

```
for(im=0;im<ord;im++){
mov[im]=1.;
movd[im]=1.;
dev[im]=1.;
devd[im]=1.;
}

for(im=0;im<ord;im++){
 for(jm=0;jm<ord;jm++){
mw[im][jm]=1;
md[im][jm]=1;
}
}
mw[3][0]=0;
md[3][0]=0;
}
                  */

double func(float x[]){
register int i,j,k;
float yy;

k=0;
nplot++;

for(i=0;i<nnp;i++) xx0[i]=x[i];

for(i=0;i<ord;i++){
 for(j=0;j<ord;j++){
w[i][j]=x[k];
d[i][j]=x[k+ord*ord];
k++;
}
}

rein();
outside(mum);
yy=rmserr(mum);
if(nplot>100){
plot(335,135,top,pmum);
plote(335,340,lpos,yy);
nplot=0;
}
return yy;
}

/*void adjust2(int max,float **mat,int k){
register int i=0,j=0;
double y1=0.;
float tmp=0;
```

```
rein();
outside(mum);
rein();
y=rmserr(mum);
plot(335,135,top,mum);
plote(335,340,lpos,y);
for(i=0;i<max;i++){
 for(j=0;j<max;j++){
if(kbhit())break;
tmp=mat[i][j];
mat[i][j]=mat[i][j]+dw*(err[k]);
outside(mum);
rein();
y1=rmserr(mum);
if(y1>y){
mat[i][j]=tmp;
}
else{
y=y1;
}
tmp=mat[i][j];
mat[i][j]=mat[i][j]-dw*(err[k]);
outside(mum);
rein();
y1=rmserr(mum);
if(y1>y){
mat[i][j]=tmp;
}
else{
y=y1;
}
}
}
}
*/

double funk(float x[]){
register int i;
float yy,yt1,yt2;


 nplot++;


for(i=0;i<ord;i++){
tmop[i]=x[i];
tmdop[i]=x[i+ord];
}
rein();
outside(mum);
```

```
yt1=sqrt(err[0]*err[0]+err[2]*err[2]+err[5]*err[5]);
yt2=rmserr(mum);
yy=c3*yt1+c4*yt2;
 if(nplot>10){
plot(335,135,top,pmum);
plote(335,340,lpos,yy);
 nplot=0;
 }
return yy;
}



void calcdop(register int i, float dt){
dop[i]=/*tanh*/ sin((dop[i]+op[i]*dt));
/*dop[i]=dop[i]+op[i]*dt;*/
}

float calclop(int i){
float a=0.;
register int j;
i=i-1;
/*a=op[i];*/
for(j=0;j<ord;j++){
a=a+ w[i][j]*op[j] + d[i][j] * dop[j] ;
}
a=atan(a);
return a;
}
```

/* This is a C function modefied from the numerical recipe version, that simulates the Amoeaba behavior for the simplex method */

```
#include <math.h>
#define NRANSI
#include "nrutil.h"
#define NMAX 5000
#define ftol 1e-6
#define GET_PSUM \
                                        for (j=0;j<ndim;j++) {\
                                        for (sum=0.0,i=0;i<mpts;i++) sum += p[i][j];\
                                        psum[j]=sum;}
#define SWAP(a,b) {swap=(a);(a)=(b);(b)=swap;}

void amoeba(float **p, float *y, int ndim,
        double (*funk)(float []), int *nfunk,float *psum,float *ptry)
{
        double amotry(float **p, float *y, float *psum, int ndim,
                double (*funk)(float []), int ihi, float fac,float *ptry);
        int i,ihi,ilo,inhi,j,mpts=ndim+1;
        float rtol,sum,swap,ysave,ytry/* ,*psum */;

/*      psum=vector((long int)0,(long int)ndim-1);*/
        *nfunk=0;
/*        printf("xx %f",ftol);*/
        GET_PSUM
        for (;;) {
                ilo=0;
                ihi = y[0]>y[1] ? (inhi=1,0) : (inhi=0,1);
                for (i=0;i<mpts;i++) {
                        if (y[i] <= y[ilo]) ilo=i;
                        if (y[i] > y[ihi]) {
                                inhi=ihi;
                                ihi=i;
                        } else if (y[i] > y[inhi] && i != ihi) inhi=i;
                }
                rtol=2.0*fabs(y[ihi]-y[ilo])/(fabs(y[ihi])+fabs(y[ilo]));
                if (rtol < ftol) {
                        SWAP(y[0],y[ilo])
                        for (i=0;i<ndim;i++) SWAP(p[0][i],p[ilo][i])
                        break;
                }
                if (*nfunk >= NMAX) return;
                *nfunk += 2;
                ytry=amotry(p,y,psum,ndim,funk,ihi,-1.0,ptry);
            /* printf("%f", ytry);*/
                if (ytry <= y[ilo])
                        ytry=amotry(p,y,psum,ndim,funk,ihi,2.0,ptry);
                else if (ytry >= y[inhi]) {
                        ysave=y[ihi];
                        ytry=amotry(p,y,psum,ndim,funk,ihi,0.5,ptry);
                        if (ytry >= ysave) {
                                for (i=0;i<mpts;i++) {
                                        if (i != ilo) {
                                                for (j=0;j<ndim;j++)
                                                        p[i][j]=psum[j]=0.5*(p[i][j]+p[ilo][j]);
                                                y[i]=(*funk)(psum);
                                        }
                                }
                                *nfunk += ndim;
                                GET_PSUM
                        }
                } else --(*nfunk);
        }
/*      free_vector(psum,(long int)0,(long int)ndim-1);*/
}
#undef SWAP
#undef GET_PSUM
#undef NMAX
```

```
#undef NRANSI
/* (C) Copr. 1986-92 Numerical Recipes Software D0>4^c1&.5#Z>K'VIkaz. */
```

/* This is a C function modefied from the numerical recipe version, to test the different vertices for the simplex */

```
#define NRANSI
#include "nrutil.h"

double amotry(float **p, float y[], float psum[], int ndim,
        double(*funk)(float []), int ihi, float fac,float ptry[])
{
        int j;
        float fac1,fac2,ytry;


        fac1=(1.0-fac)/ndim;
        fac2=fac1-fac;
        for (j=0;j<ndim;j++) ptry[j]=psum[j]*fac1-p[ihi][j]*fac2;
    /* printf("%f",ptry[j]);*/
        ytry=(*funk)(ptry);
        if (ytry < y[ihi]) {
                y[ihi]=ytry;
                for (j=0;j<ndim;j++) {
                        psum[j] += ptry[j]-p[ihi][j];
                        p[ihi][j]=ptry[j];
                }
        }

        return ytry;
}
#undef NRANSI
/* (C) Copr. 1986-92 Numerical Recipes Software D0>4^c1&.5#Z>K'VIkaz. */
```

// A data file that have data file names information and training constants should be created
// as follows

```
tbfb.dat            // training input binary data file name
tbfdb.dat           // training desired output binary data file name
tbfb.dat            // recall input binary data file name
tbfdb.dat           // recal desired output binary data file name
out.dat             // actual output data file name
err.dat             // the rms error data file name
11                  // no of slots for training or recall
1.0                 // one of the error computing coefficients
0.09                // one of the error computing coefficients
1.07                // one of the error computing coefficients
0.8                 // one of the error computing coefficients
```

```
.4                                    // one of the error computing coefficients
.3                                    // one of the error computing coefficients
```

// note that the numerical recipes library file NRUTIL.C should be included in the C program project

APPENDIX B

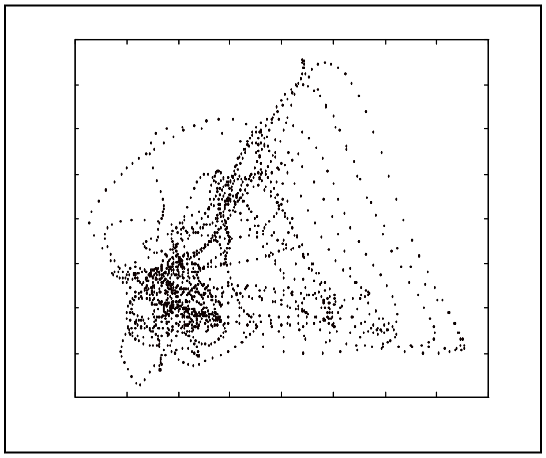
PUBLICATIONS

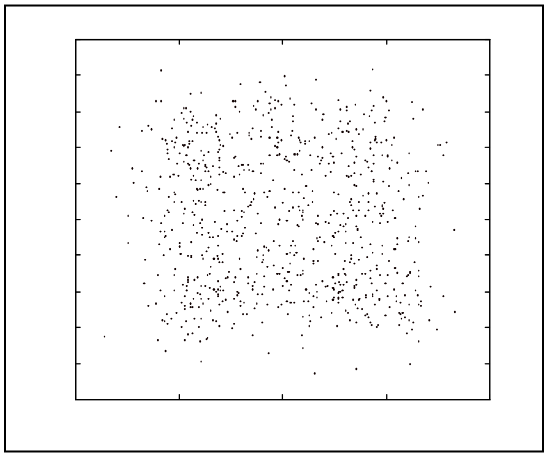
Recurrent Neural Network-Based Monitoring and Control of Chaotic Fluidized Bed Combustion (FBC) Systems

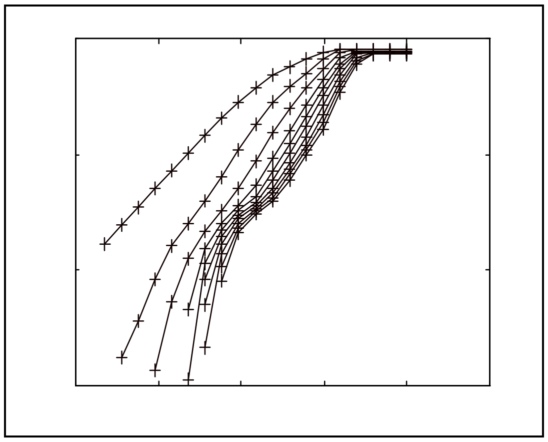
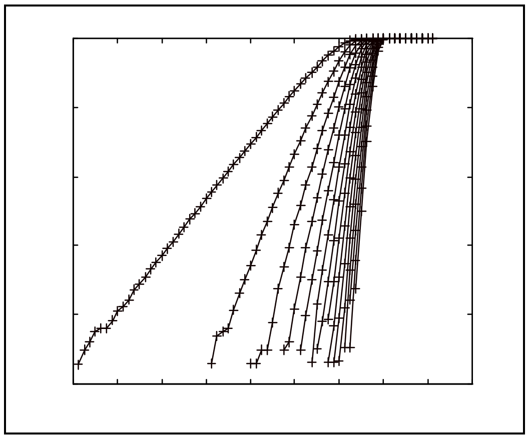
The 4th Annual Private Sector Energy Research and Development Technology Transfer Symposium, April 2-3 1996.

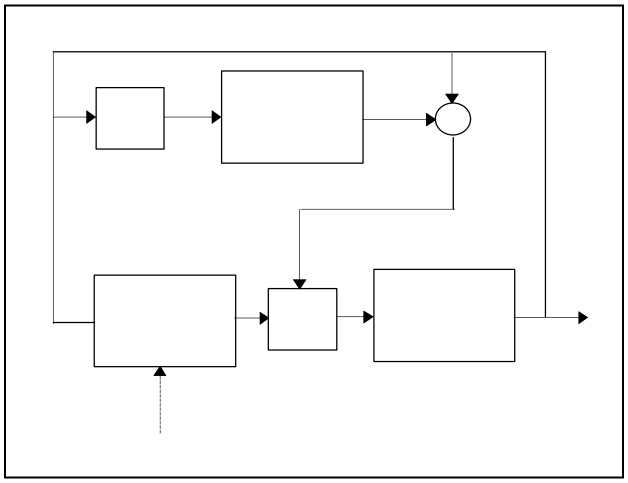
by

Department of Electrical and Computer Engineering
Tennessee State University, Nashville, TN 37209







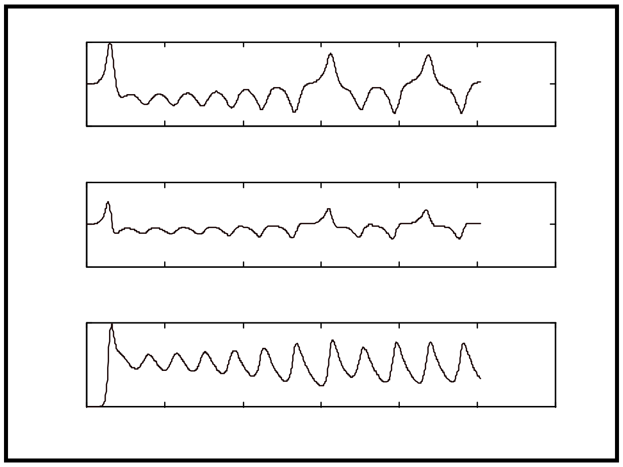
Magdi A. Essawy, Robert E. Uhrig, *" The Dynamic System Imitator: A New Approach for a Dynamic Neural Network Architecture"*, the 9th Power Plant Dynamics, Control & Testing Symposium, Knoxville, Tennessee, May 24-26 1995.

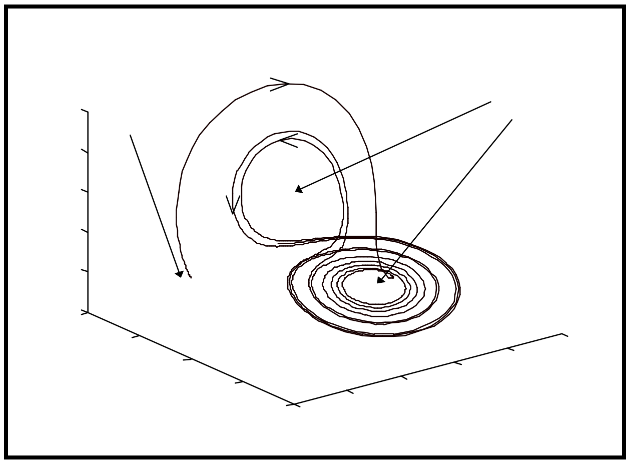
Using A Recurrent Neural Network for Chaotic Behavior Control

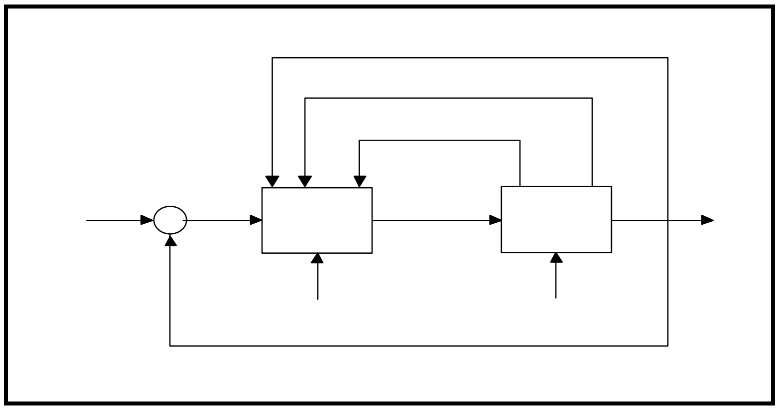
the World Congress on Neural Networks '96, Sep. 15-20 1996, pp. 143-146

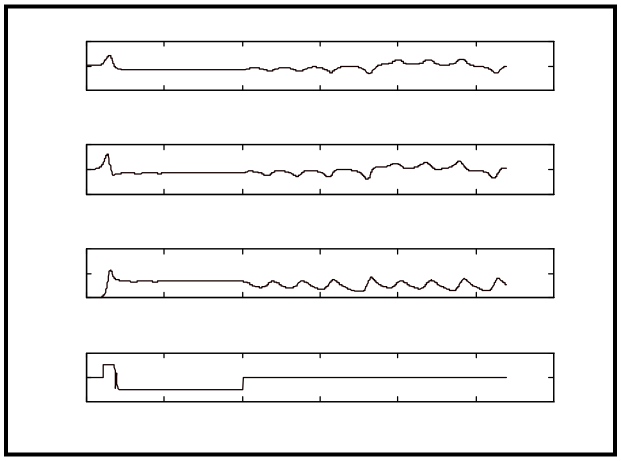
by

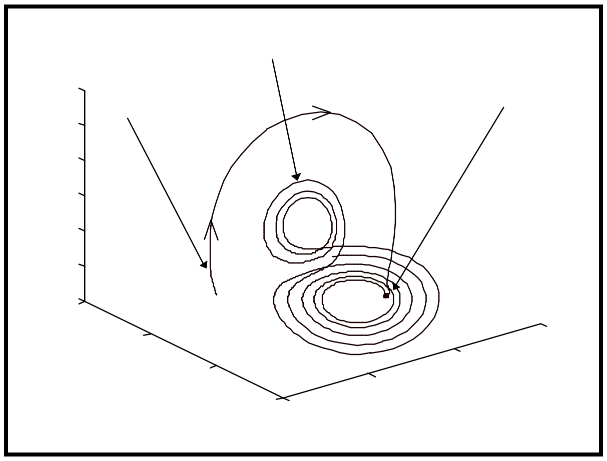
Department of Electrical and Computer Engineering
Tennessee State University, Nashville, TN 37209

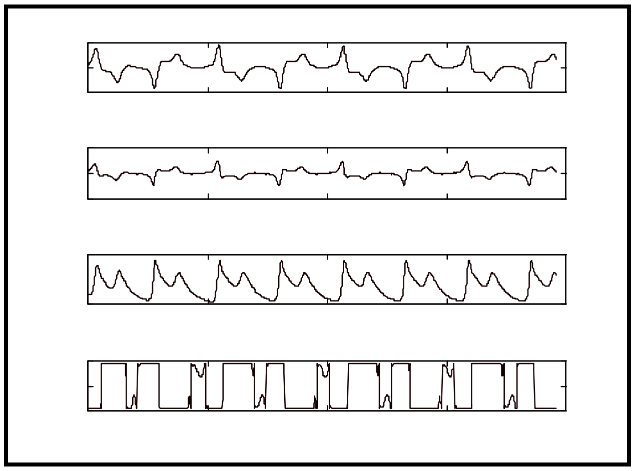
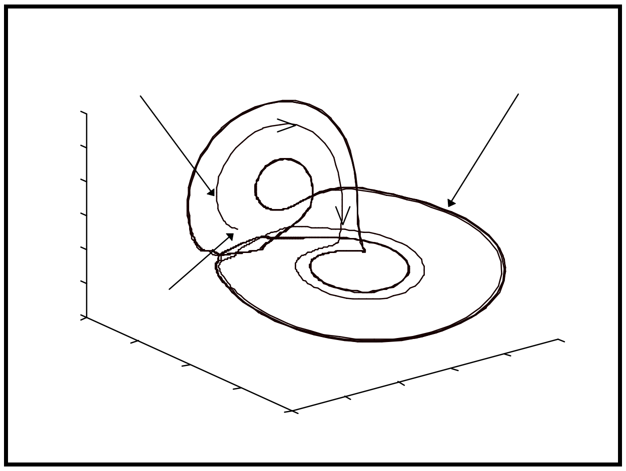










Ierative Prediction of Chaotic Time Series Using a Recurrent Neural Network

Artificial Neural Networks in Engineering (ANNIE) '96, Nov. 10-13, 1996, pp 753-762

by

MAGDI A. ESSAWY,
MOHAMMAD BODRUZZAMAN
Department of Electrical and Computer Engineering
Tennessee State University, Nashville, TN 37209

ABOLGHASEM SHAMSI,
STEPHEN NOEL
Morgantown Energy Technology Center (METC)
Morgantown, WV 26505

ITERATIVE PREDICTION OF CHAOTIC TIME SERIES USING A RECURRENT NEURAL NETWORK

MAGDI A. ESSAWY, MOHAMMAD BODRUZZAMAN
Department of Electrical and Computer Engineering
Tennessee State University, Nashville, TN 37209

ABOLGHASEM SHAMSI, STEPHEN NOEL
Morgantown Energy Technology Center (METC)
Morgantown, WV 26505

ABSTRACT:
Chaotic systems are known for their unpredictability due to their sensitive dependence on initial conditions. When only time series measurements from such systems are available, neural network based models are preferred due to their simplicity, availability, and robustness. However, the type of neural network used should be capable of modeling the highly non-linear behavior and the multi-attractor nature of such systems. In this paper we use a special type of recurrent neural network called the "Dynamic System Imitator (DSI)", that has been proven to be capable of modeling very complex dynamic behaviors. The DSI is a fully recurrent neural network that is specially designed to model a wide variety of dynamic systems. The prediction method presented in this paper is based upon predicting one step ahead in the time series, and using that predicted value to iteratively predict the following steps. This method was applied to chaotic time series generated from the logistic, Henon, and the cubic equations, in addition to experimental pressure drop time series measured from a Fluidized Bed Reactor (FBR), which is known to exhibit chaotic behavior. The time behavior and state space attractor of the actual and network synthetic chaotic time series were analyzed and compared. The correlation dimension and the Kolmogorov entropy for both the original and network synthetic data were computed. They were found to resemble each other, confirming the success of the DSI based chaotic system modeling.

INTRODUCTION

Chaotic systems are known for their unpredictability, due to their sensitive dependence on initial conditions which is measured by positive Lyapunov exponents.[1] In other words, even when the exact model of a chaotic system is available, it is impossible to predict a chaotic system behavior for a long period of time.[2,3] The reason is that our measurements and calculations are never perfect and are susceptible to errors. Similar errors contribute to the non-exact determination of initial conditions. Any minute error in the initial conditions for a chaotic system will turn, with time, into great differences in the results. However, short term predictions of chaotic systems are still possible. [2,4] How short the time duration is for valid prediction depends on the system average loss of information represented by its Lyapunov exponents and Kolmogorov entropy. Some systems are less predictable than others due to faster loss of information with time, represented by larger positive Lyapunov exponents and larger positive Kolmogorov entropy. Since exact predictions are not possible for such systems, approximate models may produce results as satisfactory as those produced by exact models. This makes neural network based models very good candidates for such applications. Neural network based models are known not to be exact models, but they are easy to implement, robust, fast and data driven. Dynamic neural network models are preferable for such applications, due to their ability to capture time behaviors. [5]

In this work we used a special type of dynamic neural network called the Dynamic System Imitator (DSI). [6] We developed the DSI a few years ago and have used it for several modeling and control applications. [6,7,8] The DSI is biologically motivated and is specially designed to model a wide variety of dynamic systems. It has both short term and long term memory mechanisms that enable the modeling of a system's transient and steady state behavior. In addition, the DSI behavior depends on its initial conditions the same as any differential equation model does, even though no explicit differential equation solving is incorporated in this case. What we know of the DSI characteristics encourages us to recommend it for modeling non-linear systems in general and chaotic systems in particular. More details about the DSI will be discussed below. Since the dynamics of most real systems are accessed via time series measurements, the focus in this paper will be on modeling chaotic time series. The way the chaotic time series model is implemented in this paper is through a one step predictor model. The dynamics of a chaotic time series are modeled through training the DSI to perform a one step prediction. However, at any point of time, the DSI response depends on the initial conditions at time zero, the history of inputs and network state variables, and the current network input. Assuming the network was able to capture the dynamics in the time series, we can start the trained network with any set of initial conditions, use a number of initial data points to put the network on track, and iteratively feed the output of the network back to compute next predicted values. Even though we applied this methodology to several theoretical systems, the current motive is to use it in a strategy to identify certain chaotic behavior modes encountered in a Fluidized Bed Reactor (FBR) system. [9] This identification can be achieved by comparing the actual measurement from the chaotic system with the time series predicted by the DSI iterative predictor model, starting from a short time history of the actual data. In this paper, results from the DSI iterative predictor are discussed for chaotic time series generated using the logistic, Henon and the cubic equations, in addition to one experimental time series measured from the FBR experiment at the Morgantown Energy Technology Center. [9] The DSI network model was evaluated based on comparison made on the time series, phase space trajectories, and chaotic parameters computed from these trajectories. However, in this case, time series similarities are not as important as similar phase space trajectories and similar chaotic parameters. [5] The actual combination of DSI iterative predictor and FBR system is published in other domains. [9]

THE DYNAMIC SYSTEM IMITATOR (DSI) NEURAL NETWORK

The neural network used for the chaotic time series prediction in this paper is a dynamic neural network called Dynamic System Imitator (DSI). The DSI is a fully recurrent neural network that is specially designed to model a wide variety of dynamic systems. [6,7] As shown in Figure.1, the DSI has a three layer structure: input, hidden, and output layer. Connections have both weights and integrators in parallel to model short term and long term memory mechanisms that handle modeling of time behaviors and time lags in real systems. Every node in the input layer has one input, $x_k(t)$, and two outputs defined by: [6]

$$o_{1j}^{i}(t) = x_j(t)\ ,$$
$$o_{2j}^{i}(t) = \int_0^t x_j(t)\ dt\ . \tag{1}$$

The input layer is fully connected to the hidden and output layers. Every node in the hidden layer is connected to every other node in the hidden and output layers and to itself. The two outputs of every neuron are computed according to the relationship: [6]

$$o_{1j}^{h}(t) = A_j\psi_j(B_j^{-1}[\sum_{k=0}^{m}(w_{1jk}^{hi}o_{1k}^{i}(t) + w_{2jk}^{hi}o_{2k}^{i}(t)) + \sum_{k=0}^{n}(w_{1jk}^{hh}o_{1k}^{h}(t) + w_{2jk}^{hh}o_{2k}^{h}(t))]\ )$$

$$o_{2j}^{h}(t) = C_j\psi_j(D_j^{-1}\int_0^t o_{1j}^{h}(t)\ dt) \tag{2}$$

where y_j is a nonlinear transformation function, and A_j, B_j, C_j, and D_j are adjustable weights associated with the hidden neuron j, which are used to shape the transfer function for every node. B and D are used to adjust the steepness of the function, while A and C are used to adjust its min-max value. Also m and n are the number of processing nodes in the input and hidden layers respectively; w_1 and w_2 refer to weights associated with direct and delayed outputs, respectively. The superscript h refers to the hidden layer, i refers to the input layer, hi refers to weights from the input to hidden layer, and hh refers to weights from the hidden to hidden layer. The integrators and feedback connections promote enough asynchrony and interaction in the network to model several system state variables as a function of time. When enough intermediate state variables are generated, the network output can be a function of those state variables. The output layer has only one output per node, which is computed according to the following equation: [6]

$$o_{j}^{o}(t) = E_j\psi_j(F_j^{-1}[\sum_{k=0}^{m}(w_{1jk}^{oi}o_{1k}^{i}(t) + w_{2jk}^{oi}o_{2k}^{i}(t)) + \sum_{k=0}^{n}(w_{1jk}^{oh}o_{1k}^{h}(t) + w_{2jk}^{oh}o_{2k}^{h}(t))]) \tag{3}$$

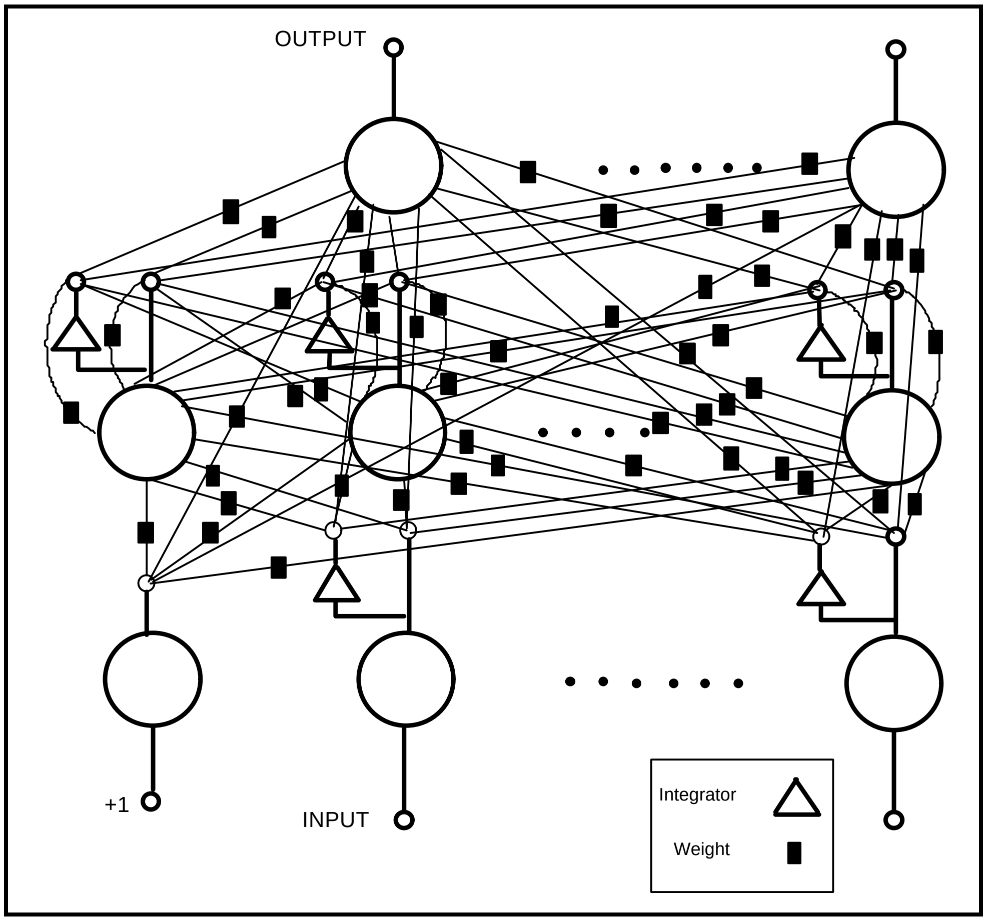


Figure 1: Schematic diagram of the Dynamic System Imitator (DSI) Network.

where m and n are the number of nodes in the input and hidden layers respectively, and E_j and F_j are two constants associated with each node in the output layer to shape its own transfer function when needed. The superscript o refers to the output layer, i refers to the input layer, h refers to the hidden layer, oi refers to weights from the input to output layer, and oh refers to weights from the hidden to output layer.

By looking at the complete DSI network design, it is easy to observe that the node interaction, information feedback, and action transfer time lags generate an activity in the network that is similar to the internal activity in real dynamic systems. Even with a simple configuration, the DSI has a complex structure, which makes it very difficult to train. A multi-dimensional optimization technique that adopts the simplex method is used to train the DSI. [6] There are two other difficulties in the training of such a network. One is that a very long time series cannot be introduced to the network at one time and must be divided into reasonably sized sections. The other is that the behavior of the network is dependent on the initial conditions of its state variables, and a certain set of initial conditions has to be found in conjunction with every network design. In other words, whenever the network is updated, a new set of initial conditions must be found. The first problem was overcome by using a moving time window that cascades the introduction of the time series segments to the network during training. The network final conditions at the final training step of every segment is taken as the initial conditions for the next segment, to keep the physical association between the consequent segments of the time series. The second problem was overcome by adding initial conditions search method that runs after every training iterate to find updated initial conditions for every modified version of the network. An arbitrary set of initial conditions can be used at the start of the training process.

FLUIDIZED BED REACTOR (FBR) PRESSURE DATA COLLECTION

Morgantown Energy Technology Center has built and operated a cold flow model to emulate fluid dynamics in a Fluidized Bed Reactor (FBR). The cold flow verification test facility consists of a ten foot high jetting fluidized bed made of clear acrylic and configured as a half cylinder vessel to facilitate jet observation. A central nozzle, made up of concentric pipes, continuously fed solids at 0 to 8 psig pressures. Separate flow loops controlled the conveyance of solids (inner pipe), the make-up air flow (middle pipe), sparger flow (outer pipe), and six air jets on the sloping conical grid. The half round fluid bed model provided useful information to study fluidization and design issues including jet penetration, chaotic pressure fluctuations, and mass flow rates of particles in various regions of the jetting fluid bed. The fluid bed tests were conducted using cork particles to simulate the relative density of gases to scale for a high pressure coal conversion reactor. As expected, the test generated chaotic pressure fluctuations. The differential pressures were measured at two location with each location consisting of two pressure taps spaced four inches apart. The lower pair of pressure taps were placed at a height just above the nozzle and the upper pair of pressure taps were placed at a height where the jet becomes evenly distributed across the diameter of the reactor. Differential pressure data collected at the higher sensor served as the primary data for the investigation of chaos. It clearly indicated the fluidization regime of the bed supported by visual observations. Data were collected on a data acquisition card at a rate of 50 Hz.

USING THE DSI FOR ITERATIVE PREDICTION OF CHAOTIC TIME SERIES

A simple configuration of the DSI neural network was used for the iterative prediction of a chaotic time series. This configuration has one node in the input layer, three nodes in the hidden layer, and one node in the output layer. The network was trained to predict one point ahead of the time series, using a set of previous values. These values are not explicitly used for prediction, but are implicitly used by adjusting the state of the network from which the prediction is performed. The prediction method is based upon the idea that once the network is trained to predict one point ahead with good accuracy, this same point can be used as an input to the network to predict the next point. This process can be repeated iteratively to predict many points in the time series. Naturally, the accuracy of prediction will deteriorate over time. During training, a time window of 200 points was used to cascade the time series to the network. The algorithm was applied to three types of simulated chaotic time series generated from the logistic, Henon, and cubic equations, in addition to one experimental time series measurement taken from a Fluidized Bed Reactor (FBR). FBR systems are known for their chaotic behavior, as discussed in several references. [9,10,11,12] The network was able to learn simple one step prediction in a reasonable number of training iterations. After training, the output of the DSI was used iteratively to generate the time series. However, the training initial conditions together with the first actual 25 points of the training time series should be used to start the DSI, in case the time behavior of the training time series must be generated. If not, any network initial conditions and any starting points can be used to generate the state space behavior of the system to which the training time series belongs.

Comparing the predicted time series to the actual time series, we found that the DSI was able to track the training time series time behavior for a short period of time (around 30 points), when started from the training initial conditions , and activated by the first 25 points of the training time series. However, it was able to track the state space attractor to which the training time series belongs, starting from any initial conditions, activated by any arbitrary set of starting points. The only case that fails is zero initial conditions together with zero starting points, which leads to zero solution The actual and predicted time series for all cases are shown in Figures 2-9, while the actual and predicted state space attractors are shown in Figures 10-17. To quantitatively compare these attractors, the correlation dimension and Kolmogorov entropy for the actual and predicted attractors were computed. The results of the correlation dimension and Kolmogorov entropy of the different cases are summarized in Table.1. The correlation dimension is computed, according to the box-counting method, from the slope of the lines representing the correlation integral versus ε (the size of a computing box) on log-log curves for different embedding dimensions. The correlation integral was computed according to the following equation: [13]

$$C(N,\varepsilon) = \frac{2}{N(N-1)} \sum_{j=1}^{N} \sum_{i=j+1}^{N} \theta(\varepsilon - |x_i - x_j|) \qquad (4)$$

where θ(x)=1 for x>0 and θ(x)=0 for x<0. The Kolmogorov entropy was computed according to the equation: [14]

$$K_{m,d} = \frac{1}{\tau m} \ln \frac{C_d(\varepsilon)}{C_{d+m}(\varepsilon)} \qquad (5)$$

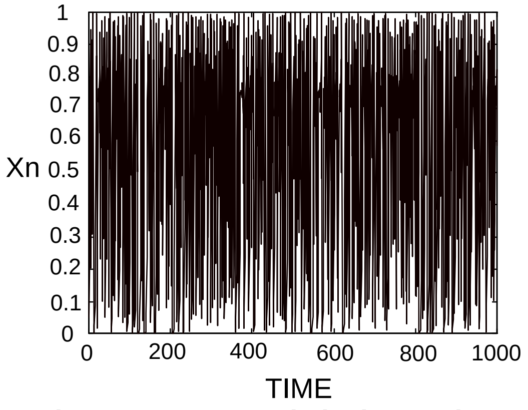


Figure 2: Actual logistic time series.

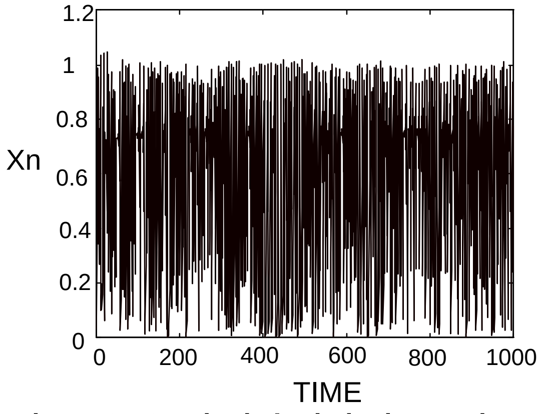


Figure 3: Synthetic logistic time series.

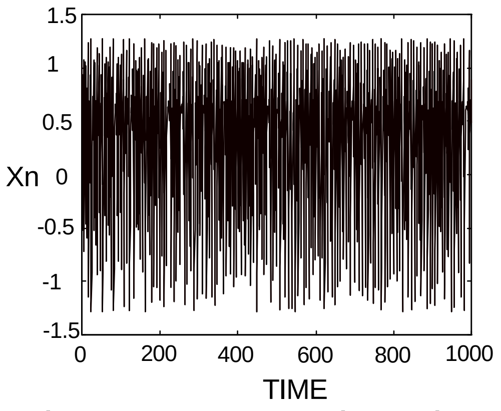


Figure 4: Actual Henon time series.

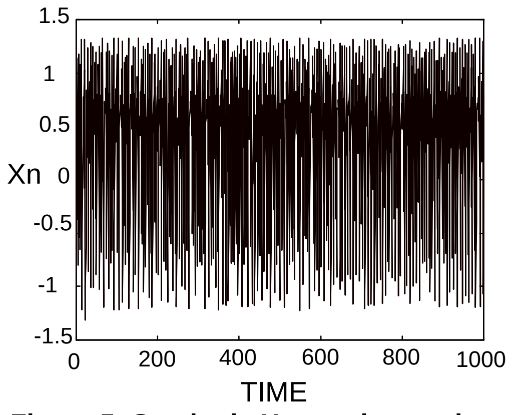


Figure 5: Synthetic Henon time series.

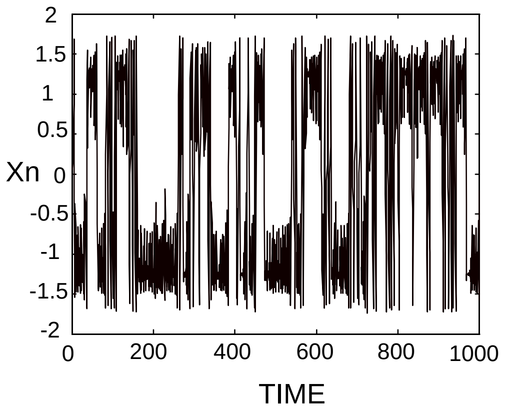


Figure 6: Actual cubic time series.

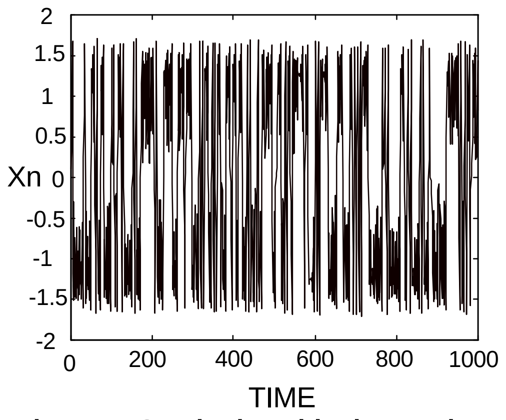


Figure 7: Synthetic cubic time series.

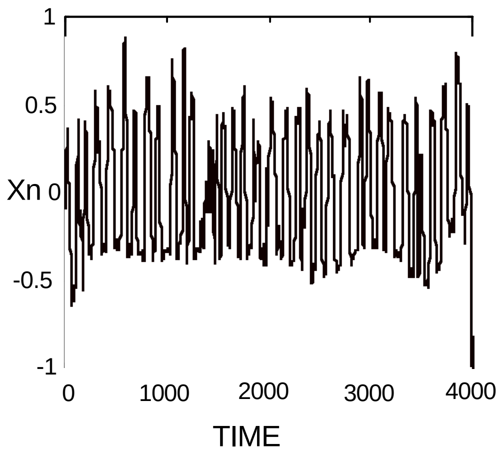


Figure 8: Actual normal FBC time series.

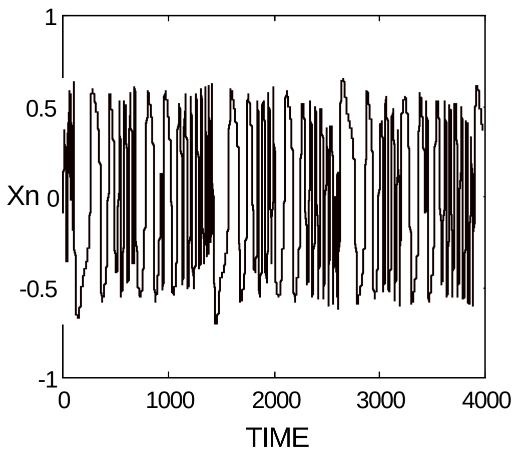


Figure 9: Synthetic normal FBC time series.

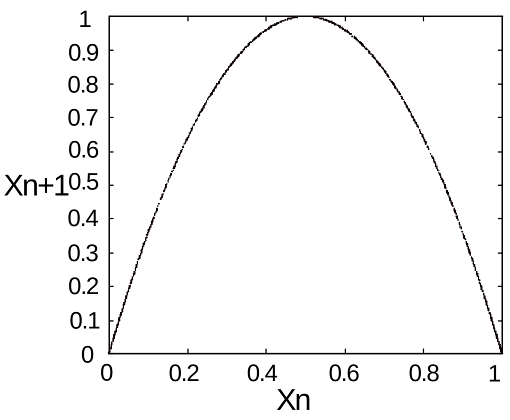


Figure 10: Actual logistic attractor.

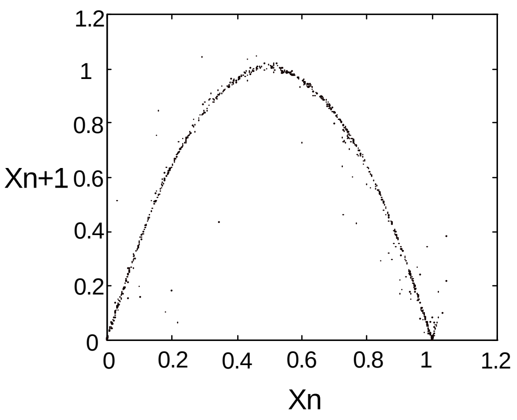


Figure 11: Synthetic logistic attractor.

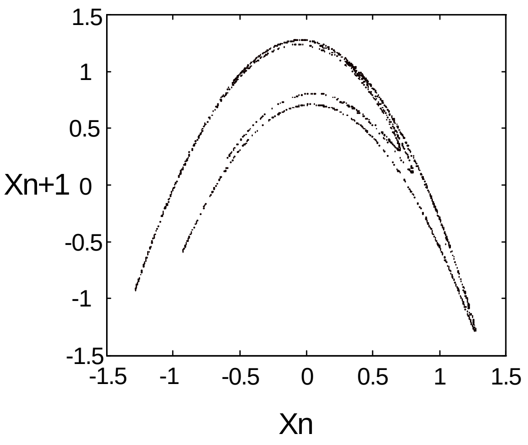


Figure 12: Actual Henon time attractor.

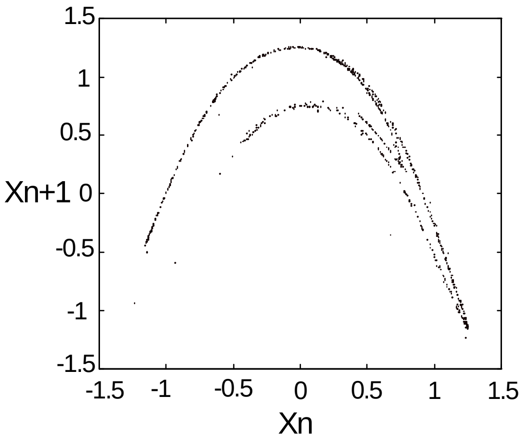


Figure 13: Synthetic Henon attractor.

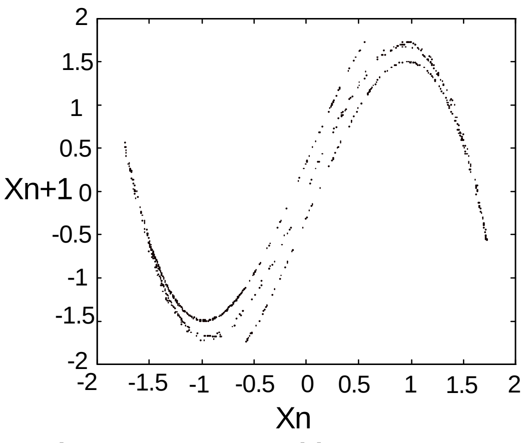


Figure 14: Actual cubic attractor.

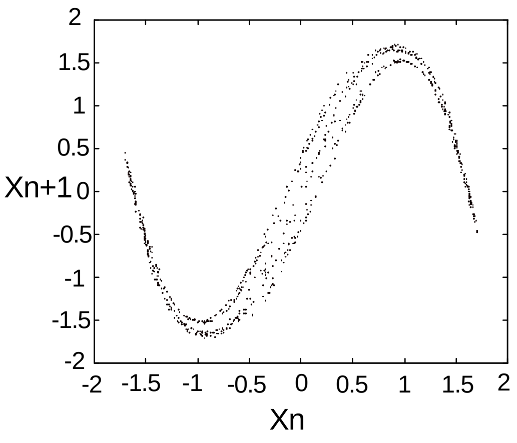


Figure 15: Synthetic cubic attractor.

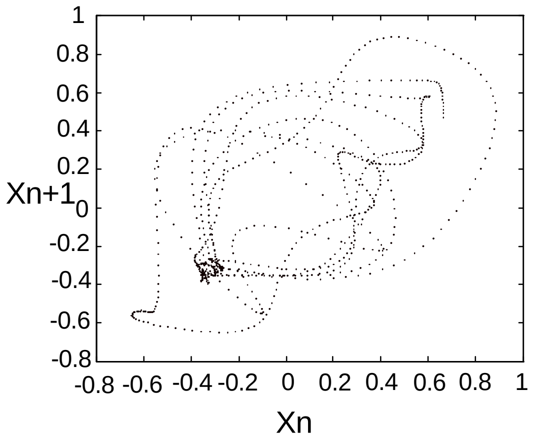


Figure 16: Actual normal FBC attractor.

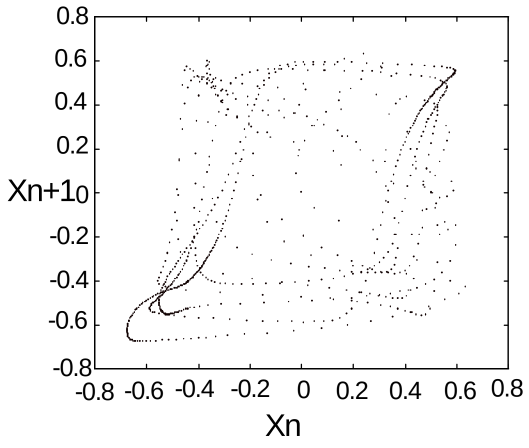


Figure 17: Synthetic normal FBC attractor.

TABLE 1: COMPARISON OF THE ACTUAL AND DSI SYNTHETIC ATTRACTORS PARAMETERS

Time Series	Correlation Dimension	Kolmogorov Entropy
Logistic Map (actual)	1.0457±0.0057	0.6872±0.0198
Logistic map (synthetic)	1.2316±0.0374	0.6013±0.0435
Henon Map (actual)	1.2607±0.0331	0.3267±0.0135
Henon map (synthetic)	1.3171±0.0725	0.2962±0.0239
Cubic Map (actual)	1.2248±0.0090	0.4245±0.0183
Cubic Map (synthetic)	1.8171±0.01365	0.5340±0.0134
FBC normal (actual)	2.934±.065	5.034±.095
FBC normal (synthetic)	2.12±.07	6.3764±1.26

CONCLUSION

In this paper a dynamic neural network based model for chaotic time series has been developed. A one step predictor model was used to iteratively generate chaotic time series. A dynamic neural network called the Dynamic system Imitator (DSI) was utilized. The DSI has distinguishable dynamic features due to its special architecture. The DSI time behavior depends on its initial conditions. After training, the DSI was able to generate the chaotic time series in all test cases. For a short period of time, it was able to generate the same time behavior of the training time series if started with the same initial conditions and the first initial points of the time series. Furthermore, it was able to track the system attractor to which the training time series belongs, for any period of time, for any initial conditions and any initial points, in all test cases. The only case that fails is the zero initial conditions together with zero starting points, which leads to a zero solution. This methodology was applied to three known chaotic models, the logistic, Henon and cubic maps, in addition to one experimental time series taken from differential pressure measurement of a Fluidized Bed Reactor (FBR). The correlation dimension and the Kolmogorov entropy for the actual and DSI network synthetic data were computed and compared. There is a very good match between the actual and synthetic time series parameters in all cases which indicates that the DSI was able to learn the dynamics in those chaotic time series to a very good extent.

ACKNOWLEDGMENT

This work has been supported by the DOE Grant No. DE-FG22-94MT94015. We wish to thank the DOE for their support of this research.

REFERENCES

1. E. Ott, “Controlling chaos,” Physics Today, pp 34-40, May 1995.
2. J.D. Farmer, J. Sidorowich, “Predicting Chaotic Time Series, ” Phys. Rev. Lett. 59, pp 845, 1987.
3. M. Casdagli, “Non-linear Prediction of Chaotic Time Series,” Physica D 35, pp 335, 1989.
4. G. Sugihara, R. M. May, “Non-linear Forecasting as a way of Distinguishing Chaos from Measurement Error in Time Series,” Nature 344, pp 734, 1990.
5. J. C.Principe, J-M Kuo, “Dynamic Modeling of Chaotic time Series with Neural Networks,” Advances in Neural Information Processing Systems 7, Ed. Tesauro, Touretzky, Leen, pp 311-318, 1995.
6. Magdi A. Essawy, “Artificial Neural Networks for System Modeling Monitoring and Control,” Ph.D. Dissertation, University of Tennessee, Knoxville, July 1995.
7. Magdi A. Essawy, Robert E. Uhrig, “The Dynamic System Imitator: A New Approach for a Dynamic Neural Network Architecture,” the 9th Power Plant Dynamics, Control & Testing Symposium, Knoxville, Tennessee, May 24-26 1995.
8. Magdi A. Essawy, Mohammad Bodruzzaman, “Using A Recurrent Neural Network for Chaotic Behavior Control,” to appear in the World Congress on Neural Networks '96, Sep. 15-20 1996.
9. Magdi A. Essawy, Mohammad Bodruzzaman, “Recurrent Neural Network-Based Monitoring and Control of Chaotic Fluidized Bed Combustion Systems,” the 4th Annual HBCU Private Sector Energy Research and Development Technology Transfer Symposium, Greensboro, NC, April 2-3 1996.
10. Fuller, “Interpretation of Pilot Scale, Fluidized Bed Behavior Using Chaotic Time Series Analysis,” 12th international FBC Conference, LA Jolla, CA, May, 1993.
11. S. Daw, “Characterization of Voidage and Pressure Signals from Fluidized Beds, Using Deterministic Chaos Theory,” Fluidized Bed Combustion, ASME Conference 1991, pp 777-785.
12. S. Halow, “Characterizing Fluidized Bed Behavior, By Decomposition of Chaotic Phase Space Trajectories,” 1993 Annual AIChE Meeting, St. Louis, MO, November, 1993.
13. Ding, C. Grebogi, E. Ott, T. Sauer, J. A. Yorke, “Plateau Onset for Correlation Dimension: When Does It Occur ?”, Physics Review Letters 70, 3872, 1993.
14. Peter Grassberger, “Estimation of the Kolmogorov Entropy from a Chaotic Signal”, Physical Review A, Vol. 28, No. 4, pp 2591-2593, October, 1983.

Computing Chaotic System Control Parameters Using Autoregressive Techniques

Proceedings of The IEEE 20th Southeastern Symposium on System Theory (SSST-97), Cookeville, TN_, March 9-11, 1997, pp 460-465

by

JEHAD. I. ABABNEH
MAGDI A. ESSAWY,
MOHAMMAD BODRUZZAMAN

Department of Electrical and Computer Engineering
Tennessee State University, Nashville, TN 37209

Computing Chaotic System Control Parameters Using Autoregressive Techniques

Jehad I. Ababneh, B.Sc., Graduate Research Assistant
Magdi A. Essawy, Ph. D., Research Associate
Mohammed Bodruzzaman, Ph. D., Associate Professor
Department of Electrical and Computer Engineering
Tennessee State University
3500 John A., Merritt Blvrd., Nashville, TN 37209

Abstract

The sensitivity to initial conditions and orbit complexity which characterize chaotic systems give them a great flexibility to control as compared to non-chaotic systems. The motion of chaotic dynamical systems can be converted to periodic motion through small time-dependent perturbations, which is a technique developed by Edward Ott, Celso Grebogi, and James York in 1990, known as the OGY technique. In this paper, a systematic way to compute the OGY control parameters from a chaotic time series is discussed. It is shown how to use autoregressive modeling to estimate the linear map that describes the system iterative behavior around a fixed point, which is essential for the OGY technique. It is also explained how to experimentally compute the rate of change of a fixed point with respect to a control parameter. And the procedure to be followed according to the order of the controlled periodic orbit is presented. Finally, it is demonstrated how these techniques can be applied to control the chaotic behavior of some typical chaotic systems, such as the logistic and Henon maps, to any periodic orbit.

Introduction

Chaotic systems are known for their unpredictability. Thus, they were mistaken in the past with random systems. It was found later on that chaotic systems are driven by deterministic phenomena, and their unpredictable behavior is due to their sensitive dependence on initial conditions [1]. The presence of chaos may be a great advantage for control in some situations [3]. In non-chaotic systems small control force will change the system dynamics slightly. On the other hand, in chaotic systems small control force can cause a large change in the system behavior. Also, a wide choice between a rich variety of dynamical behavior is possible[3]. Based on this observation, several chaotic system control methods have been developed [1-7]. Small perturbation control of chaotic systems is a technique developed by Edward Ott, Celso Grebogi, and James York in 1990, known as the OGY technique [4]. In this technique, the chaotic system motion can be stabilized to one of its naturally unstable periodic orbits. This can be achieved by applying time-dependent small pre-calculated perturbations to a system parameter. These control perturbations are applied only when the system behavior comes to a vicinity of the fixed point corresponding to the periodic orbit to be stabilized. The first step in the OGY technique is to map the system behavior to a chosen surface of section such that the periodic orbit to be stabilized appears as a fixed point. Second, the stable and unstable manifolds at this fixed point are computed from the eigenvalues and the corresponding eigenvectores of the linear map that describes the system behavior around the fixed point. The rate of change of the fixed point with respect to the control parameter is also computed. We wait for the system to come in the vicinity of the fixed point and then apply a small perturbation such that the system's next iterate falls on the stable manifold of the fixed point. In theory , once the system is on the stable manifold, its iterates will move toward the fixed point by natural forces. For the OGY method to be implemented, a linear map that describes the system behavior around the fixed point to be controlled is essential. This linear map can be estimated from time series measurements from the system. No exact model for the system is needed, which makes it possible for this method to be implemented in a wide range of problems [8-11]. In this paper, we present a systematic way to compute the OGY control parameters from a chaotic time series. We show how to use autoregressive modeling to estimate the linear map that describes the system iterative behavior around a fixed point. We also explain how to experimentally compute the rate of change of a fixed point with respect to a control parameter. And how to modify the algorithm according to the order of the periodic orbit to be controlled. Finally, we demonstrate how these techniques can be applied to control the chaotic behavior of some typical chaotic systems, such as the logistic and Henon maps, to any periodic orbit.

The OGY Technique

The OGY method is a technique to stabilize the chaotic behavior in an n-dimensional chaotic system to one of its naturally unstable periodic orbits. First, the system behavior has to be mapped on a chosen surface of section or a return map, in order to observe the system dynamics as a sequence of points on a two dimensional map. Let $\zeta_1, \zeta_2, \zeta_3, \ldots\ldots\zeta_n$ denote the coordinates in the surface of section at the n'th piercing of the surface of section. Suppose the iterates are represented by

$$\zeta_{n+1} = f(\zeta_n, p)$$

(1)

where P is some accessible system parameter

Then, we examine the unstable periodic orbits and select the one to be used for control, and obtain its stability properties. For purposes of simplicity, let us assume that a first order return map is constructed from a one dimensional time series $x_1, x_2, \ldots\ldots, x_n$, then the two dimensional iterates ζ_n will be defined as:

$$\zeta_n = \begin{bmatrix} x_{n+1} \\ x_n \end{bmatrix}$$

(2)

From this map a fixed point or a periodic orbit will be selected and examined. A fixed point is defined as $x_{n+1} = x_n$, while a period k orbit is defined as $x_{n+k} = x_n$. The period k orbit, as an example, will appear as k distinct points on the first return map, while it will appear as a fixed point on the k-order return map. We will always need to select the appropriate return map order that shows the controlled periodic orbit as a fixed point

$$\zeta_F = \begin{bmatrix} x_F \\ x_F \end{bmatrix}$$

(3)

The OGY technique assumes that there is an acceptable maximum perturbation δp_m in the system control parameter P. It is also assumed that the position of the periodic orbit is a function of P, while the local dynamics around this periodic orbit do not vary much when the parameter P is changed within the allowable perturbation [5].

In order to examine the stability of the selected periodic orbit we need to perform the following computations:

1. We find the linear approximation of the map f around the selected fixed point, as shown in the following equation:

$$\zeta_{n+1} - \zeta_F = M(\zeta_n - \zeta_F)$$

(4)

where M is a two dimensional matrix.

2. Let λ_s and λ_u be the experimentally determined stable and unstable eigenvalues of the matrix M, respectively, and $(e_s,\ e_u)$ are the corresponding stable and unstable eigenvectors. The eigenvectors e_s and e_u represent the stable and unstable directions of the map around the selected fixed point, and are used to compute the unstable contravariant eigenvector according to the relationships $f_u . e_u = 1$ and $f_u . e_s = 0$ [4].

3. The rate of change of the selected fixed point with respect to the system control parameter P is computed as:

$$g = \frac{\partial \zeta_F}{\partial p} \approx \frac{\Delta \zeta_F}{\Delta p}$$

(5)

4. Given the maximum allowable perturbation in the control parameter δp_m, we can compute the maximum effective control distance around the fixed point as:

$$\delta\zeta_m = \left(\frac{\lambda_u - 1}{\lambda_u}\right)\ \delta p_m\ (g . f_u)$$

(6)

Knowing the parameters computed above, we can compute the necessary control force at iterate n as:

$$\delta p = C(\zeta_n - \zeta_F) . f_u$$

(7)

where $C = \left(\frac{\lambda_u}{\lambda_u - 1}\right)\ (g . f_u)^{-1}$

(8)

The perturbation δp is the control action necessary to put the system's next iterate on the stable manifold of the fixed point. Then in the following iterates should move toward the fixed point with the system's natural forces. However, to adjust for noise and computation inaccuracies, we need to calculate a correcting control force δp at the end of every period.

Methodology

In this section, we present the details of our approach to compute the OGY parameters and control action from chaotic time series measurements. The procedure is summarized in the following steps:

Step 1. *Determination of the System Periodic Orbits and Fixed Points*

Starting from a time series measurement at a nominal value of the control parameter P_0, we need to localize the periodic orbits embedded in the system. Actually, we will be interested in only one of these orbits with the targeted period. All periodic orbits embedded in a chaotic system are unstable orbits. This means that the system will visit those orbits briefly at certain instances. We need to catch this brief periodic behavior of the system, localize it, and determine its behavior, in order to compute the control actions needed to stabilize it. The time series measurements can be easily arranged in a two column matrix defined as:

$$X = \begin{bmatrix} x_1 & x_{1+k} \\ x_2 & x_{2+k} \\ . & . \\ . & . \\ x_{n-k} & x_n \end{bmatrix} \qquad (9)$$

where n is the total number of data points in the time series, and k is the period of the targeted orbit. Any two points on the same row of the matrix X are separated by k number of points. If a pair of points x_n and x_{n+k} on the time series are equal, then any of the periods k, k/2, k/3.....or k/k=1 is localized. Any of these periods, if exist, will appear as a fixed point on the k-order return map (x_n vs. x_{n+k}). If the target period k exists, any point on it visited by the system for a complete cycle in the measured interval will appear as a distinct fixed point on the k-order return map. If all k orbital points were visited, they will appear as k distinct fixed points on the k-order return map. We compute the absolute value of the difference of the two columns of the matrix X that comes to a vector of n-k values. We search this vector for any value less than a minute tolerance ε. Any of the values passing the selection criteria will mark one fixed point on the k-order return map. After marking a fixed point on a specific row, we need to go back and unfold all points between x_n and x_{n+k} at the marked row. If points x_n ...x_{n+k-1} are all distinct, then this is one period k orbit, and the corresponding fixed point on the k-order return map is a point of interest to the current control task. Any of the k fixed points belonging to the orbit $(\zeta_{01}, \zeta_{02}, \ldots\ldots, \zeta_{0k})$ can be used to control the system.

Step 2. *Computing the Rate of Change of the Fixed Point with Respect to the Control Parameter*

New measurements from the system need to be recorded at values of the control parameter $p_0 \pm \delta p$. For every measurement we need to recalculate the location of the period k fixed points on the k-order return map $(\zeta_{m1}, \zeta_{m2}, \ldots\ldots, \zeta_{mk})$, using the method on step 1. In case we have decided to use the fixed point ζ_{0j} to control the system, the change in its position will be $\delta\zeta_i = \zeta_{ij} - \zeta_{0j}$, and the rate of change of that fixed point will be computed as $g_i = \dfrac{\delta\zeta_i}{\delta p_i}$, where $\delta p_i = p_i - p_0$. We repeat the computation for the available measurements (0, 1, ..., m) and average them to get the value of g to be used.

Step 3. *Estimating the Linear Approximation for the Map f around a Fixed Point*

As shown in equation 1 the map f correlates the consecutive points on a two dimensional map. To apply the OGY method, we need a linear estimate of that map around the fixed point used to control the system. This means we need to estimate the two dimensional matrix that maps the consecutive points in a close vicinity of the fixed point. Equation 4 can be expanded as:

$$\begin{bmatrix} x_{n+2} - x_F \\ x_{n+1} - x_F \end{bmatrix} = \begin{bmatrix} m_{11} m_{12} \\ m_{21} m_{22} \end{bmatrix} \begin{bmatrix} x_{n+1} - x_F \\ x_n - x_F \end{bmatrix}$$

(10)

From the second row equality of this equation, we can easily infer that $m_{21} = 1$, and $m_{22} = 0$. On the other hand the first row equality is:

$$x_{n+2} - x_F = m_{11}(x_{n+1} - x_F) + m_{12}(x_n - x_F) \qquad (11)$$

This equation means that the difference between an iterate and the fixed point can be obtained from the difference of the two previous iterates to the fixed point. We can consider this equation as an autoregressive model for the iterates, and use the least squares method to estimate the parameters of that model. From the matrix X in equation 9, in which every row represents an iterate on the two dimensional k-order return map, we collect all pairs of points ζ_n and ζ_{n+1} that are in a close vicinity to the fixed point. In other words, we collect the pairs that lie inside a small pre-defined circle around the fixed point. According to the least squares estimate of the autoregressive model, we can estimate the m_{11} and m_{12} elements of the matrix M as:

$$\begin{bmatrix} m_{11} \\ \\ m_{12} \end{bmatrix} = D^{-1} \begin{bmatrix} \frac{1}{N}\sum(x_{n+2} - x_F)(x_{n+1} - x_F) \\ \frac{1}{N}\sum(x_{n+2} - x_F)(x_n - x_F) \end{bmatrix} \qquad (12)$$

where N is the number of pairs and D^{-1} is given by

$$D^{-1} = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix}$$

$$d_{11} = \frac{1}{N}\sum(x_{n+1} - x_F)^2$$

$$d_{12} = \frac{1}{N}\sum(x_{n+1} - x_F)(x_n - x_F)$$

$$d_{21} = \frac{1}{N}\sum(x_{n+1} - x_F)(x_n - x_F)$$

$$d_{22} = \frac{1}{N}\sum(x_n - x_F)^2$$

Step 4 . *Finding the OGY Parameters*

We compute the eigenvalues and the corresponding eigenvectors of the matrix M. Assume that λ_s and λ_u are the experimentally determined stable and unstable eigenvalues, respectively, and e_s and e_u are the corresponding eigenvectors. The eigenvectors e_s and e_u represent the stable and unstable directions around the fixed point. From the eigenvectors e_s and e_u we compute the unstable contravariant eigenvector according to the relationships $f_u . e_u = 1$ and $f_u . e_s = 0$ [4]. Hence, the maximum allowable distance for control around the fixed point can be computed according to equation 6 as:

$$\delta\zeta_m = \left(\frac{\lambda_u - 1}{\lambda_u}\right) \delta p_m \ (g . f_u)$$

and the necessary control perturbation for the n'th iterate can be computed by equation 7 as:

$$\delta p = C(\zeta_n - \zeta_F) . f_u$$

where $C = \left(\frac{\lambda_u}{\lambda_u - 1}\right) (g . f_u)^{-1}$

The way the OGY control method is applied, is to wait for the system until an iterate falls into a circle with a radius $\delta\zeta_m$ around the fixed point, then apply the perturbation $p = p_0 + \delta p$. Then, we wait k iterates and check if the system falls back within the radius $\delta\zeta_m$ around the fixed point. If it does, we measure the distance between that iterate and the fixed point, re-estimate the control force and reapply the necessary control action. These time dependent kicks will stabilize the system around the selected periodic orbit.

Results and Discussion

The control procedure discussed above was applied to control the chaotic behavior in two typical chaotic systems, the logistic map defined as:

$$x_{n+1} = x_n \lambda (1 - x_n)$$

(13)

and the Henon map defined as:

$$x_{n+1} = 1 - \alpha x_n^{\ 2} + y_n$$

(14)

$$y_{n+1} = \beta x_n$$

The chaotic behavior in the two systems was stabilized to several periodic orbits. Figure 1 shows the results of applying and then removing the control on the logistic map ($\lambda = 3.825$) to stabilize it to a period-1 orbit. The computed parameters for this case are as follows:

$$x_F = 0.738504$$

$$g = \begin{bmatrix} 0.059 \\ 0.059 \end{bmatrix}$$

$$M = \begin{bmatrix} -1.8252 & 0.0001 \\ 1 & 0 \end{bmatrix}$$

$$\lambda_u = -1.8253$$

$$\lambda_s = 4.3315e-005$$

$$f_u = \begin{bmatrix} -1.1402143 \\ 0.0000493 \end{bmatrix}$$

$$C = -9.6$$

$$\delta\zeta_m = 0.01822$$

The control was applied at the iterate 500 and then removed at the iterate 2500. We notice that the effect of the controller did not appear right after it was applied, the reason is that we have to wait for the system iterates to fall within the vicinity of the fixed point to be stabilized before we actually apply the control kicks. It is also clear that once the control was removed the chaotic behavior has returned. This is due to noise and inaccuracies in the computation as mentioned above. In addition, the system actually needs an infinite time until it becomes exactly on the periodic orbit. In theory, once it is there the iterates should stay on the orbit unless perturbed. Figure 2 shows the control action applied as a function of time. It is clear that after few iterates from the control start, the kicks necessary to keep the system stabilized at that orbit are very small, which makes this control algorithm very inexpensive and easy to implement in many practical systems. Figures 3 and 4 show how the OGY method was used to stabilize the logistic map to period-2 and period-5 respectively. Also, figures 5 and 6 show how it was used to stabilize the Henon map to period-1 and period-2 respectively.

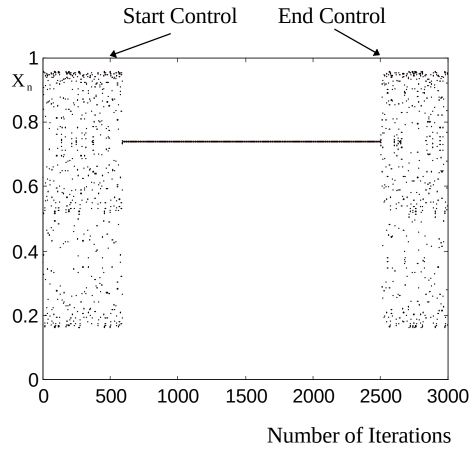


Figure 1 logistic map control to fixed point

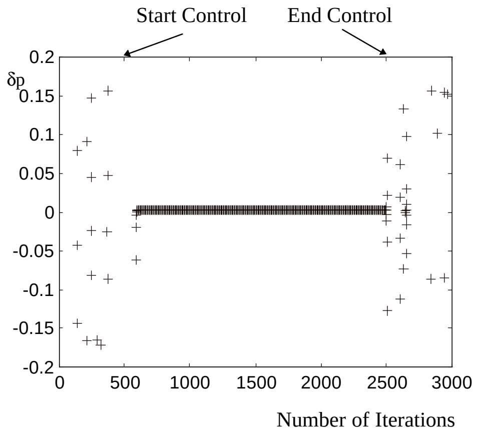


Figure 2. The control perturbations necessary to achieve the control shown in Figure 1.

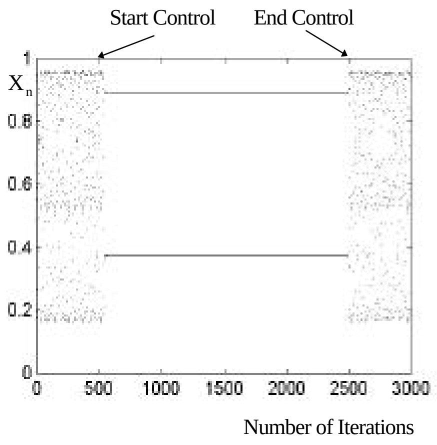


Figure 3. Controlling the logistic map to period two.

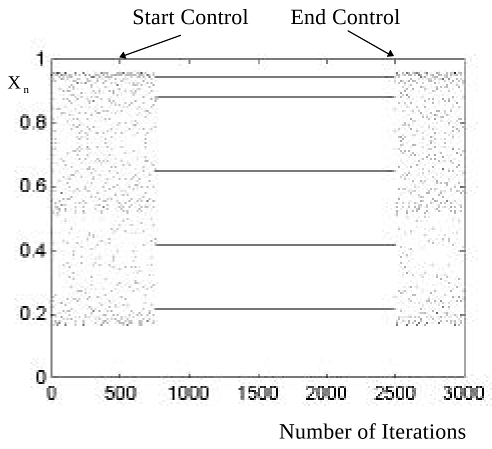


Figure 4. Controlling the logistic map to period five.

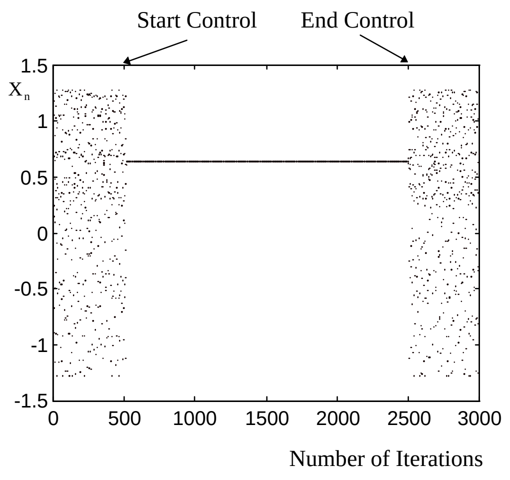


Figure 5. Controlling the Henon map to period one

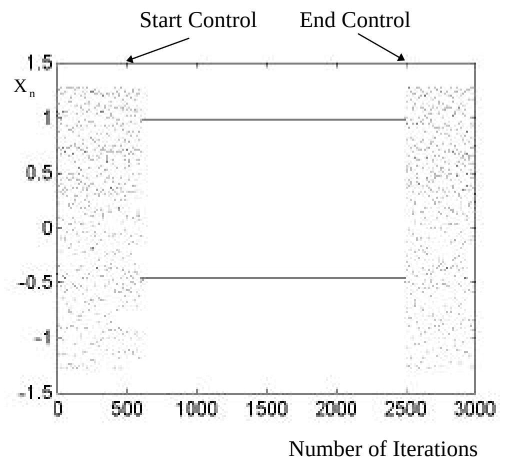


Figure 6. Controlling the Henon map to period two

Conclusions

In this paper, we have presented a systematic way to compute the parameters of the OGY technique from time series measurements, and to implement this method to control the chaotic behavior in a chaotic system, and stabilize it into one of its naturally unstable periodic orbits. We have developed a technique to compute the linear map that describes the system behavior on a two dimensional map around a fixed point, which is essential to the OGY method, using autoregressive modeling. We have shown a detailed procedure to localize and separate a naturally unstable periodic orbit in a chaotic time series. We have also explained how to experimentally compute the rate of change of the fixed point corresponding to the periodic orbit to be controlled with respect to a control parameter. Finally, we have applied the techniques developed in this paper to control the chaotic behavior in some typical chaotic systems, such as the logistic and the Henon maps, and stabilize it to one of its naturally unstable periodic orbits. Stabilizing such chaotic systems to several example periods are presented. In the control method described, we have to wait for the system until its iterates fall into a close vicinity of the fixed point corresponding to the orbit to be stabilized, and then apply time dependent sequence of kicks that stabilizes the system on that naturally unstable orbit. From the results summarized above, it is clear that the control actions necessary to stabilize the system become very small and almost negligible after few iterates from the start of the control, which makes this method very inexpensive, and easy to implement in many practical systems. However, the control kicks has to be maintained or control will be lost, and the system behavior will go back to its naturally chaotic behavior.

Acknowledgment

This work has been supported by the DOE Grant No. DE-FG22-94MT94015.

References

1. E. Ott, “Controlling chaos,” Physics Today, pp 34-40, May 1995.
2. J. Singer, “Controlling a Chaotic System,” Physical Review Letters, Vol. 66, pp 1230-1232, 1991.
3. T. Shinbrot, “Using Small Perturbations to Control Chaos,” Nature, Vol. 363, pp 411-417, 3 June, 1993.
4. E. Ott, “Controlling Chaos,” Physical Review Letters, Vol. 64, No 11, pp 1196-1199, 12 March, 1990.
5. W. L. Ditto, “Experimental Control of Chaos,” Physical Review Letters, Vol. 65, No 26, pp 3211-3214, 24 December, 1990.
6. R. Hunt, “Stabilizing High-Period Orbits in a Chaotic System: The Diode Resonator,” Physical Review Letters, Vol. 67, No 15, pp 1953-1955, October 1991.
7. U. Dressler, “Controlling Chaos Using Time Delay Coordinates” Physical Review Letters, Vol. 68, No 1, pp 1-4, January 1992.
8. V. Petrov, “Tracking Unstable Periodic Orbits in the Belousov-Zhabotinsky Reaction,” Physical Review Letters, Vol. 72, No 18, pp 2955-2958, May 1994
9. S. Hayes, “Experimental Control of Chaos for Communication,” Physical Review Letters, Vol. 73, No 13, pp 1781-1784, September 1994.
10. R. Roy, “Dynamical Control of a Chaotic Laser: Experimental Stabilization of a Globally Coupled

System," Physical Review Letters, Vol. 68, No 9, pp 1259-1262, March 1992.

11. A. Garfinkel, "Controlling Cardiac Chaos," Science, Vol. 257, pp 1230-1235, 1992.

Localization, Monitoring and Control of Chaotic Behavior in Fluidized Bed Systems

to appear in the Maintenance And Reliability CONference (MARCON 97), Knoxville, Tennessee, May 20-22 1997

by

MAGDI A. ESSAWY,

MOHAMMAD BODRUZZAMAN

Department of Electrical and Computer Engineering

Tennessee State University, Nashville, TN 37209

LOCALIZATION, MONITORING AND CONTROL OF CHAOTIC BEHAVIOR IN FLUIDIZED BED SYSTEMS

Magdi Essawy, and Mohammad Bodruzzaman
Tennessee State University
Department of Electrical and Computer Engineering
3500 John A. Merritt Blvd., Nashville, TN 37209

ABSTRACT

Coal-fired power plants are very important source for electric power generation in the United States and worldwide. This is because Coal is abundant and inexpensive compared to oil and gas. However, greater use of coal is constrained by the difficulties of solid fuel use and of cleanup of combustion products containing ash and gaseous pollutants. Results of innovative research can expand coal utilization by making it more convenient to handle and by increasing its reliability to that of liquid and gaseous fuels. Direct utilization of coal fluidized combustion is of interest because of potential cost savings and improved environmental performance. Thus, pressurized fluidized-bed combustors (FBC) are becoming very popular, efficient, and environmentally acceptable replica for conventional boilers in Coal-fired and chemical plants. In this paper, we present neural network-based methods for chaotic behavior monitoring and control in FBC systems, in addition to chaos analysis of FBC data, in order to localize chaotic modes in them. Both of the normal and abnormal mixing processes in FBC systems are known to undergo chaotic behavior. Even though, this type of behavior is not always undesirable, it is a challenge to most types of conventional control methods, due to its unpredictable nature. The performance, reliability, availability and operating cost of an FBC system will be significantly improved, if an appropriate control method is available to control its abnormal operation and switch it to normal when exists. Since this abnormal operation develops only at certain times due to a sequence of transient behavior, then an appropriate abnormal behavior monitoring method is also necessary. Those methods has to be fast enough for on-line operation, such that the control methods would be applied before the system reaches a non-return point in its transients. It was found that both normal and abnormal behavior of FBC systems are chaotic. However, the abnormal behavior has a higher order chaos. Hence, the appropriate control system should be capable of switching the system behavior from its high order chaos condition to low order chaos. It is to mention that most conventional chaos control methods are designed to switch a chaotic behavior to a periodic orbit. Since this is not the goal for the FBC case, further developments are needed. We propose neural network-based control methods which are known for their flexibility and capability to control both non-linear and chaotic systems. A special type of recurrent neural network, known as Dynamic System Imitator (DSI), will be used for the monitoring and control purposes.

INTRODUCTION

Coal fired power plants are important source of electric power in the united states and all around the globe. Due to the large amounts of coal reserve around the world, it is expected that coal will stay as one of the main sources of power for many decades to come. Improving the efficiency, performance, and safety of such power plants, and minimizing toxic pollutants coming from their stacks, will make them more economically and environmentally acceptable options.

In the recent years, attention was given to a new type of energy conversion device, a critical component of any coal fired plant. This new type of energy conversion system is known as a Fluidized Bed Combustor (FBC). In such a system, the direct utilization of the coal fuel enhances the energy conversion process and reduces the energy losses as compared to conventional energy conversion systems. Considerable efforts are being made to further improve the design and performance of FBC systems.

In this paper, we introduce a neural network-based system to monitor and control an undesirable abnormal behavior observed in FBC systems. In addition, to chaos analysis to both of the normal and abnormal behaviors. Such abnormal behavior may lead to inefficient performance, interrupted operation, or poor fuel utilization in an FBC system. Previous research show that both of the normal and abnormal behavior in FBC systems are chaotic.[1-4] A type of behavior known of its unpredictability, which makes it a challenge to any conventional monitoring or control method. This makes artificial intelligence methods a potential option due to their known abilities to deal with non-linearity, multidimensionality, and noise. Furthermore, the learning ability of neural systems will simplify the design process and will lead to more general and adaptive solutions.

Scientists in many fields often encounter systems that exhibit chaotic time evolution.[5] Chaos is abundant both in nature and man-made devices, to an extent that many scientists believe that it is the rule rather than the exception.[6] The chaotic behavior is known to be unpredictable, which may be unsafe to the operation of many devices, and make it unwelcome in many situations. On occasion, chaos is a beneficial feature as it enhances mixing and chemical reactions and provides a vigorous mechanism for transporting heat and/or mass. However, in many other situations, chaos is undesirable phenomena which may lead to vibrations, irregular operation, fatigue failure in mechanical systems, temperature oscillations which may exceed safe operational conditions in thermal systems, and increasing drag in flow systems.[6]

Chaotic motion has been regarded for many years as a troublesome property that is neither predictable nor controllable. Recently researchers have realized that chaos can actually be advantageous in many situations, and when it is unavoidably present, it can often be controlled to obtain desired results.[5,7] In 1990, Ott, Grebogi, and Yorke (OGY) demonstrated that one can convert the motion of a chaotic dynamical system to periodic motion by controlling one of the system's many unstable periodic orbits embedded in the chaotic attractor, through only small time-dependent perturbations in an accessible system parameter.[8-14] Ott and Spano [5], stated that if chaos control is practical in a system, then the presence of chaos can be an advantage. Any one of a number of different unstable orbits, in a chaotic system, can be stabilized, and one can select the orbit that gives the best system performance. Thus we have the flexibility of actually switching the system behavior by stabilizing another periodic orbit. On the other hand, if the system is actually stable and periodic, we can use control to only slightly change its performance, and we do not have similar flexibility to what is available in a chaotic system. Hence, we may even sometimes wish to build chaos into a system where it is naturally absent.

In this paper, we present a method where a recurrent type of neural network can be used to control the chaotic behavior in a chaotic system. We suggest to use this method to switch the abnormal chaotic behavior in an FBC system to its chaotic normal state. This control system should be accompanied with an on-line monitoring system that initiates the control actions once the abnormal behavior is monitored. We present a neural network-based strategy to identify the different chaotic behavior modes encountered in an FBC system. [15] This identification process can be achieved by comparing the actual measurement from the chaotic system with the time series predicted by neural network-based iterative predictor model, starting from a short time history of the actual data. A dynamic type of neural network will be used for both monitoring and control tasks. Dynamic neural networks are considered due to their time behavior and their ability to deal with transient conditions. A special type of recurrent neural network called the Dynamic System Imitator (DSI) will be used.

LOCALIZATION OF THE CHAOTIC BEHAVIOR IN THE FBC DATA AVAILABLE

Morgantown Energy Technology Center has built and operated a cold flow model to emulate fluid dynamics in an FBC system. As illustrated in Figure 1, the cold flow verification test facility consists of a ten foot high jetting fluidized bed made of clear acrylic and configured as a half cylinder vessel to facilitate jet observation. A central nozzle, made up of concentric pipes, continuously fed solids at 0 to 8 psig pressures. Separate flow loops controlled the conveyance of solids (inner pipe), the make-up air flow (middle pipe), sparger flow (outer pipe), and six air jets on the sloping conical grid. The half round fluid bed model provided useful information to study fluidization and design issues including jet penetration, chaotic pressure fluctuations, and mass flow rates of particles in various regions of the jetting fluid bed. The fluid bed tests were conducted using cork particles to simulate the relative density of gases to scale for a high pressure coal conversion reactor. As expected, the test generated chaotic pressure fluctuations. The differential pressures were measured at two location with each location consisting of two pressure taps spaced four inches apart. The lower pair of pressure taps were placed at a height just above the nozzle and the upper pair of pressure taps were placed at a height where the jet becomes evenly distributed across the diameter of the reactor. Differential pressure data collected at the higher sensor served as the primary data for the investigation of chaos. It clearly indicated the fluidization regime of the bed supported by visual observations. Data were collected on a data acquisition card at a rate of 50 Hz.

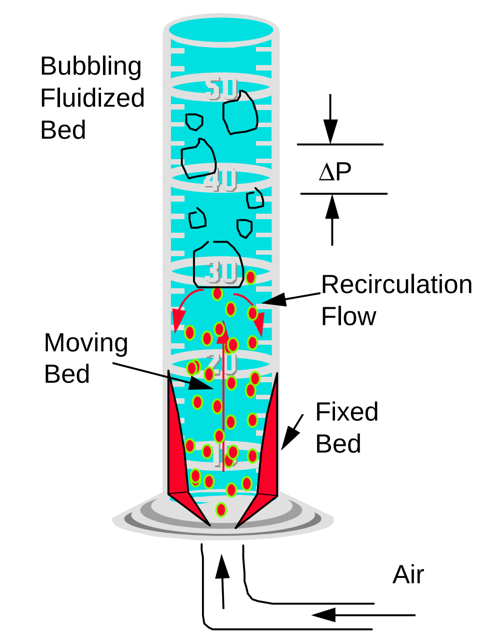


Figure.1 The general setup of the existing cold FBC experiment at METC.

Through our analysis of both the normal and abnormal FBC pressure data, it is evident that they have strange attractors, and both belong to a chaotic system. This means that the prospective control method needs to switch the system from one chaotic state to another chaotic state, which has been achieved before in simple chaotic systems such as the logistic map.[16] A two dimensional projection of attractors from a normal and abnormal FBC time series are shown in Figures 2 and 3, respectively. As it appears in the two figures the normal data attractor looks much more well behaved than the abnormal data. We performed several methods to compute the parameters of both attractors. We computed the correlation dimension, the Kolmogorov entropy, and the Lyapunov exponents of the normal and abnormal attractors. The correlation integral was computed according to the following equation:[17]

$$C(N,\varepsilon)=\frac{2}{N(N-1)}\sum_{j=1}^{N}\sum_{i=j+1}^{N}\theta(\varepsilon-|x_i-x_j|), \tag{1}$$

where $\theta(x)=1$ for $x>0$ and $\theta(x)=0$ for $x<0$. The Kolmogorov entropy was computed according to the relationship:[18]

$$K_{m,d}=\frac{1}{\tau m}\ln\frac{C_d(\varepsilon)}{C_{d+m}(\varepsilon)} \tag{2}$$

And the Lyapunov exponents were computed according to the Sano-Swada technique to compute the Lyapunov spectrum from a chaotic time series.[19]

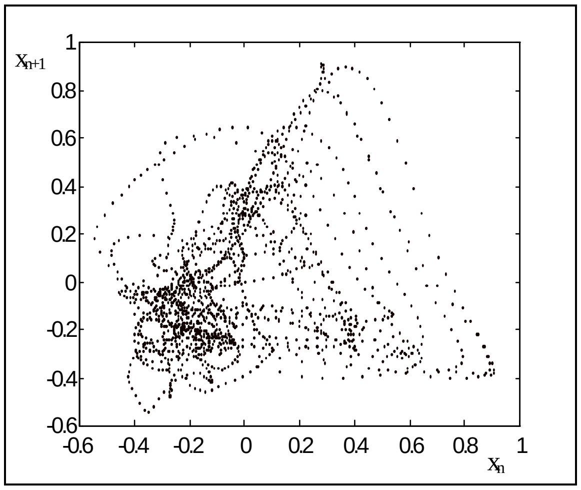


Figure 2 A plot of the FBC normal attractor projected onto two dimensional map.

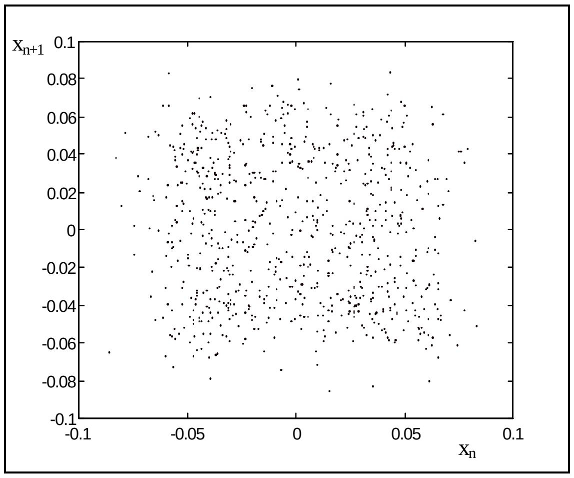


Figure 3 A plot of the FBC abnormal attractor projected onto two dimensional map.

The correlation integral graphs for the normal and abnormal attractors in a range of embedding dimensions are shown in Figures 4 and 5, respectively. The results of the chaos analysis of the FBC data are summarized in Table 1. These analysis show that both the normal and abnormal modes of the system live on a chaotic attractor, because both have fractal dimensions, positive Kolmogorov entropy, and positive Lyapunov exponents.

	Normal Data	Abnormal data
Correlation Dimension	3.07	17.35
Kolmogorov Entropy	8	100

Table 1 Results of chaos analysis of the FBC data.

However, the correlation dimension for the abnormal attractor is much higher than the normal one, which is an indication that when the system changes from its normal to its abnormal behavior, it goes from low order to high order chaos. This situation is a big challenge to any traditional control method. However, we believe that the system parameters can be adjusted through chaos control method to move the system from its high order chaotic behavior to the normal low order chaotic state. A proposed method for the FBC system monitoring and control is discussed below.

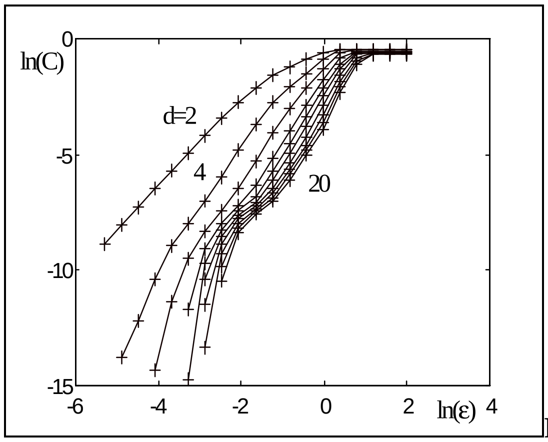


Figure 4 A plot for the correlation integral of the FBC normal attractor, for embedding dimensions 2-20.

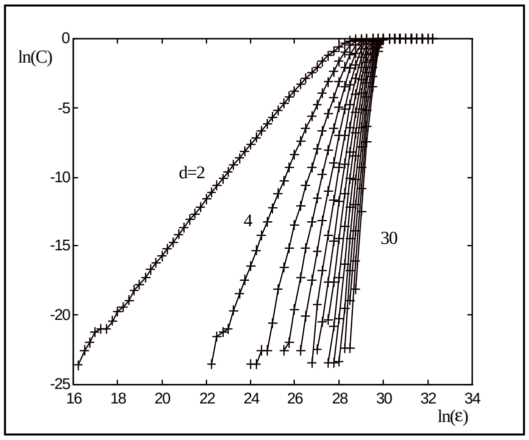


Figure 5 A plot for the correlation integral of the FBC normal attractor, for embedding dimensions 2.30

Even though the chaos analysis methods described above are the best to define the condition of a chaotic system, they are not suitable for on-line monitoring because they need intensive calculations that might run several hours on digital computers. Instead, a chaotic time series predictor technique will be used.

METHODOLOGY

For any control method to function properly with the prescribed system, there should be some monitoring device that will monitor the system state and switch the controller on, in case of detection of any abnormal behavior. We will monitor the system behavior using the difference of time history between the predicted chaotic time series predictor that we developed using the DSI neural network, and the actual time series collected using pressure sensors mounted on the FBC. Testing this chaotic time series predictor, we found that it was able to predict the chaotic behavior of chaotic time series generated by several chaotic systems such as the logistic map, the Henon map, and the cubic map. In all cases the DSI predictor was able to predict the chaotic behavior of the time series for a short time starting from some time history of the signal. It was also able to predict the state space system attractor for the rest of the time. If we train the DSI predictor to predict the normal behavior of the system, starting from some initial measurements, then the average error between the actual and predicted time series over a certain period of time will give us an indication of how much drift did the system make from its normal condition. Once a certain threshold is violated we can switch the controller on. To be able to design a controller for such a system, we need to study the effect of different system parameters on the behavior, and find which parameters are responsible for the system drift from normal. If any of these parameters would be accessible for control, we can implement a control method as illustrated in Figure 6. The only obstacle before executing and testing this control method is to find an appropriate model that will describe the prescribed system behavior in all modes. This model will be used to study the effect of system parameters on the system behavior and to test the performance of the proposed control methodology.

The reason we chose to use the DSI neural network in this situation is the known dynamic characteristics of such a network. The DSI is a fully recurrent neural network that was specially designed to model a wide variety of dynamic systems. It has feedback connections and integrators to form a compact representation of time lags and interactions in real systems. It is suitable for on-line applications due to its fast response and realistic interface with the outside world. It is also equipped with a multidimensional optimization technique and dynamic windowing which allows it learn system dynamics from long time series. The DSI neural network has been used before to control the chaotic behavior in the Lorenz system, and stabilize it into either a fixed point or a periodic orbit. In that application, one state variable was fed back as an input to the DSI, and the output of the DSI was used to control the system. The error between the actual output of the system and a target behavior was used to train the DSI network. State point perturbation control and parameter perturbation control methods have been demonstrated. We wish that a similar technique will work with the FBC system, even though we believe that the FBC system is a much more difficult problem, due to its complex behavior under both normal and abnormal conditions.

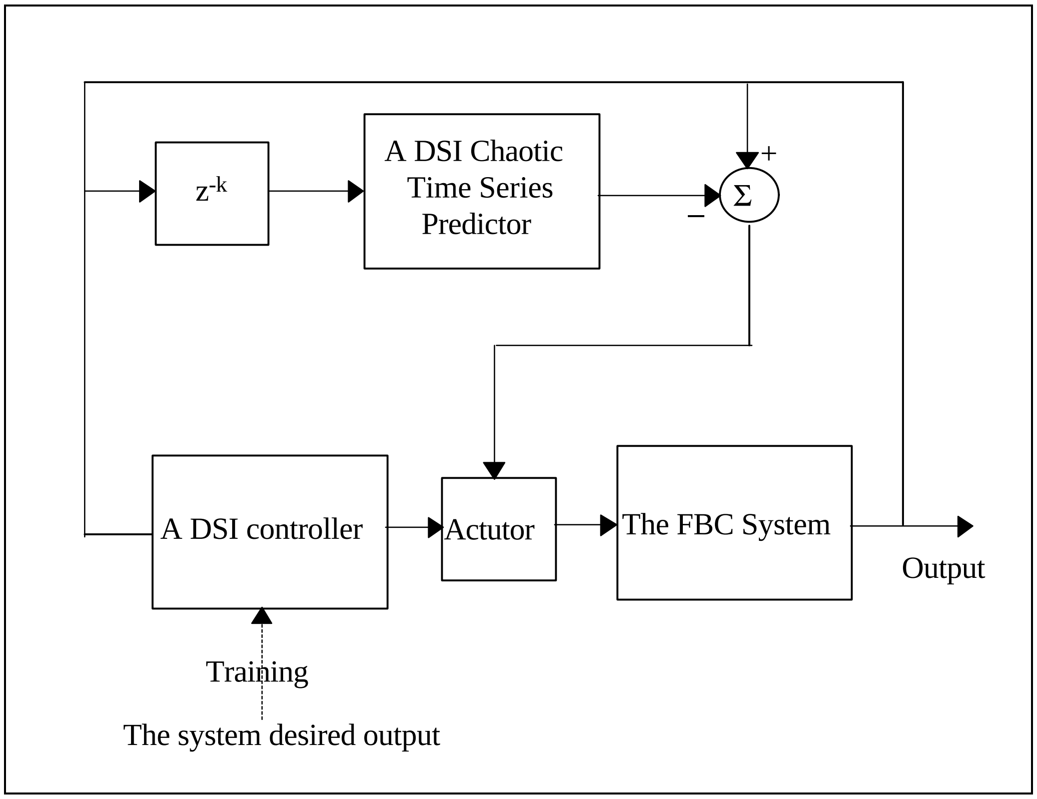


Figure 6 An illustration of a hybrid neural network-based monitoring and control method for the FBC abnormal behavior.

USING THE DSI NEURAL NETWORK FOR CHAOTIC BEHAVIOR CONTROL

In this section we will demonstrate how a special type of a neural network, the Dynamic System Imitator (DSI), can be used for chaotic behavior control. The DSI neural network is specially designed to model a wide variety of dynamic systems.[20] It is a multi-layer recurrent neural network supplied with integrators and extensive feedback connections to model the asynchrony and time lags in a real systems. The DSI neural network was used before for modeling complex dynamic behaviors through very simple configurations due to its temporal and spatial representations. The DSI controller described above was applied to control the chaotic behavior of the Lorenz system to a stable fixed point or a stable periodic orbit. Both system state point control and system parameter perturbation control strategies were implemented.

In this section, we will demonstrate how the DSI can be used to control the chaotic behavior in the Lorenz system, which is a typical chaotic system. The DSI neural network used, has one node in the input layer, three nodes in the hidden layer and one node in the output layer. The general training strategy for the DSI controller is shown in Figure 7. The DSI is trained to generate the necessary control signal to achieve certain system performance. One or more reference values in addition to feedback from the controlled system are used as inputs to the DSI. A pre-specified output behavior of the system is also supplied to the network as a target for the training. During training this pre-specified behavior is continuously compared with the actual behavior of the system to adjust the DSI parameters. Figures 8 and 9 show how the DSI neuro-controller was used to control the chaotic behavior in the Lorenz system, a typical chaotic system.

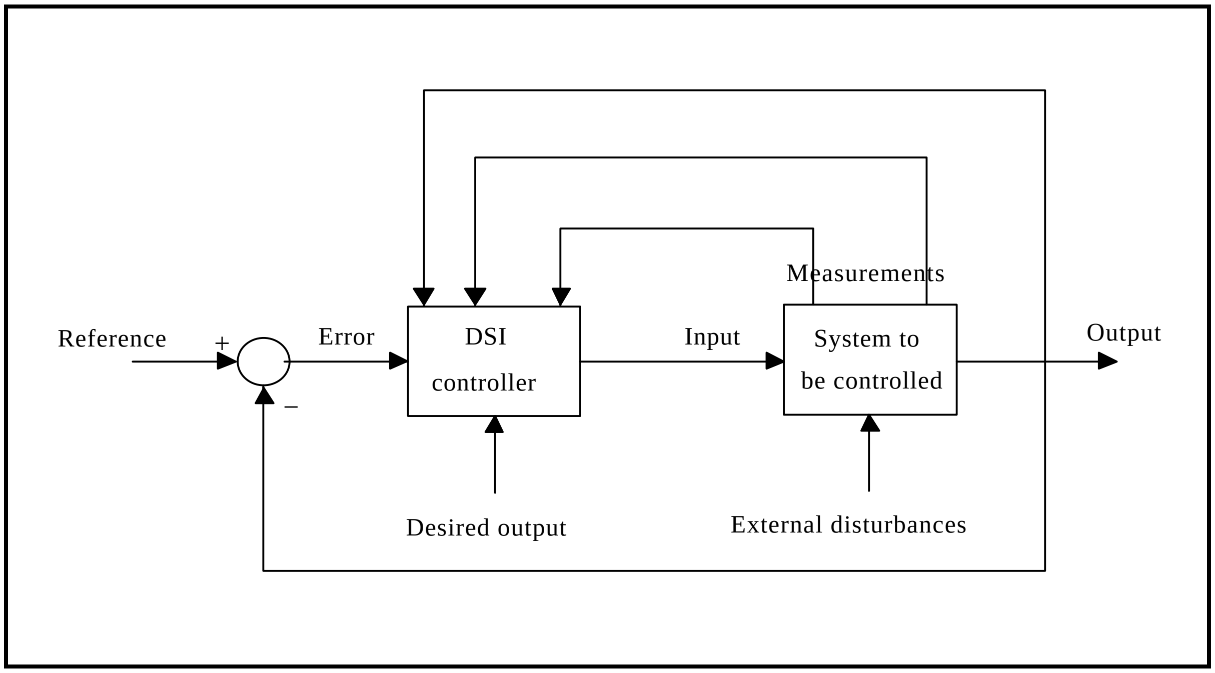


Figure 7 The general training strategy for the DSI controller.

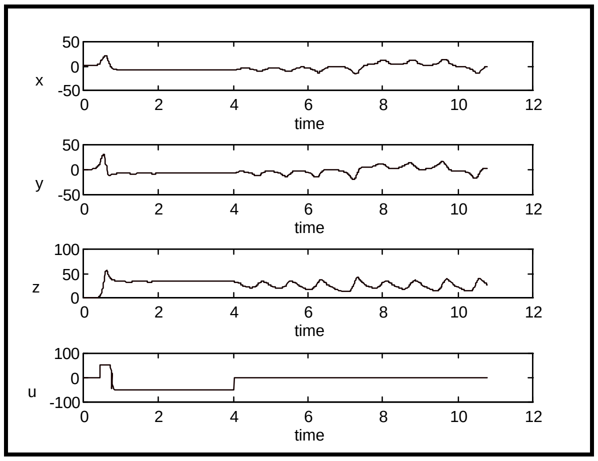


Figure 8 The DSI control signal (u(t)), and the time behavior of the Lorenz system after applying and then removing the control at the stabilized point.

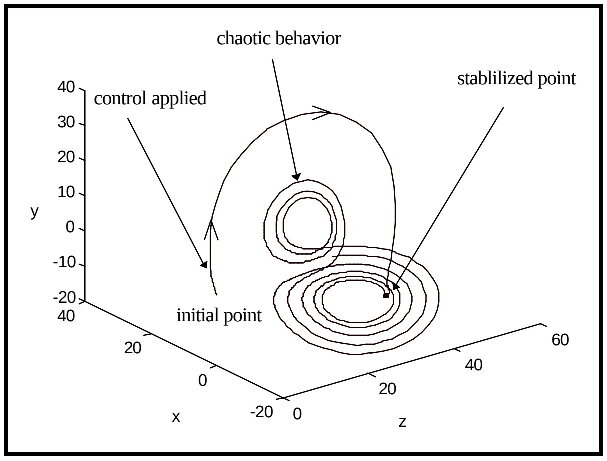


Figure 9 The trajectory of the Lorenz system, after applying and then removing the control action at the stabilized point.

ITERATIVE PREDICTION OF CHAOTIC TIME SERIES USING THE DSI NEURAL NETWORK

Chaotic systems are known for their unpredictability due to their sensitive dependence on initial conditions. When only time series measurements from such systems are available, neural network based models are preferred due to their simplicity, availability, and robustness. However, the type of neural network used should be capable of modeling the highly non-linear behavior and the multi-attractor nature of such systems. The prediction method presented in this paper is based upon predicting one step ahead in the time series, and using that predicted value to iteratively predict the following steps. This method was applied to chaotic time series generated from the logistic, Henon, and the cubic equations, in addition to experimental pressure drop time series measured from an FBC system, which is known to exhibit chaotic behavior.[1-4] The time behavior and state space attractor of the actual and network synthetic chaotic time series were analyzed and compared. The correlation dimension and the Kolmogorov entropy for both the original and network synthetic data were computed. They were found to resemble each other, confirming the success of the DSI based chaotic system modeling.

A simple configuration of the DSI neural network was used for the iterative prediction of a chaotic time series. This configuration has one node in the input layer, three nodes in the hidden layer, and one node in the output layer. The network was trained to predict one point ahead of the time series, using a set of previous values. These values are not explicitly used for prediction, but are implicitly used by adjusting the state of the network from which the prediction is performed. The prediction method is based upon the idea that once the network is trained to predict one point ahead with good accuracy, this same point can be used as an input to the network to predict the next point. This process can be repeated iteratively to predict many points in the time series. Naturally, the accuracy of prediction will deteriorate over time. During training, a time window of 200 points was used to cascade the time series to the network. The algorithm was applied to three types of simulated chaotic time series generated from the logistic, Henon, and cubic equations, in addition to one experimental time series taken from an FBC pressure measurement. The network was able to learn simple one step prediction in a reasonable number of training iterations. After training, the output of the DSI was used iteratively to generate the time series. However, the training initial conditions together with the first actual 25 points of the training time series should be used to start the DSI, in case the time behavior of the training time series must be generated. If not, any network initial conditions and any starting points can be used to generate the state space behavior of the system to which the training time series belongs.

Comparing the predicted time series to the actual time series, we found that the DSI was able to track the training time series time behavior for a short period of time (around 30 points), when started from the training initial conditions , and activated by the first 25 points of the training time series. However, it was able to track the state space attractor to which the training time series belongs, starting from any initial conditions, activated by any arbitrary set of starting points. The only case that fails is zero initial conditions together with zero starting points, which leads to zero solution. To quantitatively compare the results, the correlation dimension and Kolmogorov entropy for the actual and predicted attractors were computed. The results of the correlation dimension and Kolmogorov entropy of the different cases are summarized in Table.2. The correlation dimension is computed, according to the box-counting method, from the slope of the lines representing the correlation integral versus ε (the size of a computing box) on log-log curves for different embedding dimensions.

TABLE 2: COMPARISON OF THE ACTUAL AND DSI SYNTHETIC ATTRACTORS PARAMETERS

Time Series	Correlation Dimension	Kolmogorov Entropy
Logistic Map (actual)	1.0457±0.0057	0.6872±0.0198
Logistic map (synthetic)	1.2316±0.0374	0.6013±0.0435
Henon Map (actual)	1.2607±0.0331	0.3267±0.0135
Henon map (synthetic)	1.3171±0.0725	0.2962±0.0239
Cubic Map (actual)	1.2248±0.0090	0.4245±0.0183
Cubic Map (synthetic)	1.8171±0.01365	0.5340±0.0134
FBC normal (actual)	2.934±.065	5.034±.095
FBC normal (synthetic)	2.12±.07	6.3764±1.26

CONCLUSION

In this paper, analysis to data measured from an FBC facility has been presented. The purpose of this analysis is to build basis for a neuro-controller design that can be used to control some undesirable abnormal behavior in FBC systems. This analysis shows that both the normal and abnormal behavior represented in the data available are chaotic. The system's normal and abnormal attractors have fractal dimensions, some positive Lyapunov exponents and positive Kolmogorov entropy. However, the correlation dimension of the abnormal case is much higher than the normal case. This indicates that when the system switches to its abnormal situation it suffers a very complex behavior, most likely belongs to a higher order chaos. An appropriate controller is desired to control the system abnormal chaotic behavior to its normal chaotic behavior. A general method to monitor and control the chaotic behavior in an FBC system has been outlined. A recurrent neural network called the Dynamic System Imitator (DSI) was adopted. The only missing chain to test the proposed method is an appropriate non-linear model that will describe the FBC in different modes.

A dynamic neural network based model for chaotic time series has been developed. A one step predictor model was used to iteratively generate chaotic time series, using the DSI neural network. After training, the DSI was able to generate the chaotic time series in all test cases. For a short period of time, it was able to generate the same time behavior of the training time series if started with the same initial conditions and the first initial points of the time series. Furthermore, it was able to track the system attractor to which the training time series belongs, for any period of time, for any initial conditions and any initial points, in all test cases. The only case that fails is the zero initial conditions together with zero starting points, which leads to a zero solution. This methodology was applied to three known chaotic models, the logistic, Henon and cubic maps, in addition to one experimental time series taken from differential pressure measurement of an FBC system. The correlation dimension and the Kolmogorov entropy for the actual and DSI network synthetic data were computed and compared. There is a very good match between the actual and synthetic time series parameters in all cases which indicates that the DSI was able to learn the dynamics in those chaotic time series to a very good extent.

Based on the results of FBC data analysis described, the neural network chaotic system controller and neural network chaotic time series predictor developed, a hybrid monitoring and control system for fluidized bed combustors was outlined. However, an appropriate non-linear model for the FBC system is necessary for training and testing the developed neural network algorithms.

ACKNOWLEDGMENT

This work has been supported by the DOE Grant No. DE-FG22-94MT94015.

REFERENCES

1. Fuller, “Interpretation of Pilot Scale, Fluidized Bed Behavior Using Chaotic Time Series Analysis,” 12th international FBC Conference, LA Jolla, CA, May, 1993.
2. S. Daw, “Characterization of Voidage and Pressure Signals from Fluidized Beds, Using Deterministic Chaos Theory,” Fluidized Bed Combustion, ASME Conference 1991, pp 777-785.
3. S. Halow, “Characterizing Fluidized Bed Behavior, By Decomposition of Chaotic Phase Space Trajectories,” 1993 Annual AIChE Meeting, St. Louis, MO, November, 1993.
4. M. Bodruzzaman, and M. Essawy, "Neural-Network Based Monitoring and Control of Fluidized Bed Systems." Final Report to DOE, Pittsburg, PA, March 1997.
5. E. Ott, “Controlling chaos,” Physics Today, pp 34-40, May 1995.
6. J. Singer, “Controlling a Chaotic System,” Physical Review Letters, Vol. 66, pp 1230-1232, 1991.
7. T. Shinbrot, “Using Small Perturbations to Control Chaos,” Nature, Vol. 363, pp 411-417, 3 June, 1993.
8. E. Ott, “Controlling Chaos,” Physical Review Letters, Vol. 64, No 11, pp 1196-1199, 12 March, 1990.
9. W. L. Ditto, “Experimental Control of Chaos,” Physical Review Letters, Vol. 65, No 26, pp 3211-3214, 24 December, 1990.
10. R. Hunt, “Stabilizing High-Period Orbits in a Chaotic System: The Diode Resonator,” Physical Review Letters, Vol. 67, No 15, pp 1953-1955, October 1991.
11. V. Petrov, “Tracking Unstable Periodic Orbits in the Belousov-Zhabotinsky Reaction,” Physical Review Letters, Vol. 72, No 18, pp 2955-2958, May 1994
12. S. Hayes, “Experimental Control of Chaos for Communication,” Physical Review Letters, Vol. 73, No 13, pp 1781-1784, September 1994.
13. R. Roy, “Dynamical Control of a Chaotic Laser: Experimental Stabilization of a Globally Coupled System,” Physical Review Letters, Vol. 68, No 9, pp 1259-1262, March 1992.
14. A. Garfinkel, “Controlling Cardiac Chaos,” Science, Vol. 257, pp 1230-1235, 1992.
15. Magdi A. Essawy, Mohammad Bodruzzaman, *"Iterative Prediction of Chaotic Time Series Using a Recurrent Neural Network,"* Artificial Neural Networks in Engineering (ANNIE) '96, Nov. 10-13, 1996.
16. E. Jackson, “The Entrainment and Migration Controls of Multiple Attractor Systems,” Physics Letters A, Vol. 151, No 9, pp 478-484.
17. M. Ding, C. Grebogi, E. Ott, T. Sauer, J. A. Yorke, “Plateau Onset for Correlation Dimension: When Does It Occur ?”, Physics Review Letters 70, 3872, 1993.
18. Peter Grassberger, “Estimation of the Kolmogorov Entropy from a chaotic Signal”, Physical Review A, Vol. 28, No. 4, pp 2591-2593, October, 1983.
19. M. Sano, “Measurement of the Lyapunov Exponents from a Chaotic Series”, Physical Review Letters, Vol. 55, No. 10, pp 1082-1085, September, 1983.
20. Magdi A. Essawy, Robert E. Uhrig, *" The Dynamic System Imitator: A New Approach for a Dynamic Neural Network Architecture"*, the 9th Power Plant Dynamics, Control & Testing Symposium, Knoxville, Tennessee, May 24-26 1995.