NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

PREDICTION AND GEOMETRY OF CHAOTIC TIME SERIES

by

Mary L. Leonardi

June, 1997

Thesis Advisor: Christopher Frenzen
Thesis Co-Advisor: Philip Beaver

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June, 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE PREDICTION AND GEOMETRY OF CHAOTIC TIME SERIES	5. FUNDING NUMBERS
6. AUTHOR(S) Leonardi, Mary L.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This thesis examines the topic of chaotic time series. An overview of chaos, dynamical systems, and traditional approaches to time series analysis is provided, followed by an examination of state space reconstruction. State space reconstruction is a nonlinear, deterministic approach whose goal is to use the immediate past behavior of the time series to reconstruct the current state of the system. The choice of delay parameter and embedding dimension are crucial to this reconstruction. Once the state space has been properly reconstructed, one can address the issue of whether apparently random data has come from a low-dimensional, chaotic (deterministic) source or from a "random" process. Specific techniques for making this determination include attractor reconstruction,, estimation of fractal dimension and Lyapunov exponents, and short-term prediction.

If the time series data appears to be from a low-dimensional chaotic source, then one can predict the "continuation" of the data in the short-term. This is the "inverse problem" of dynamical systems. In this thesis, the technique of local fitting is used to accomplish the prediction. Finally, the issue of noisy data is treated, with the purpose of highlighting where further research may be beneficial.

14. SUBJECT TERMS Attractor Reconstruction, Chaos, Prediction, State Space Reconstruction, Time Series	15. NUMBER OF PAGES 117
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited.

PREDICTION AND GEOMETRY OF CHAOTIC TIME SERIES

Mary L. Leonardi
Captain, United States Marine Corps
B.A., Northwestern University, 1991

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
June, 1997

Author: Mary L. Leonardi
Mary L. Leonardi

Approved by: Christopher Frenzen
Christopher Frenzen, Thesis Advisor

Philip Beaver, Thesis Co-Advisor

W. M. Woods
Walter Max Woods, Chairman
Department of Mathematics

ABSTRACT

This thesis examines the topic of chaotic time series. An overview of chaos, dynamical systems, and traditional approaches to time series analysis is provided, followed by an examination of the method of state space reconstruction. State space reconstruction is a nonlinear, deterministic approach whose goal is to use the immediate past behavior of the time series to reconstruct the current state of the system. The choice of delay parameter and embedding dimension are crucial to this reconstruction. Once the state space has been properly reconstructed, one can address the issue of whether apparently random data has come from a low-dimensional chaotic (deterministic) source or from a "random" process. Specific techniques for making this determination include attractor reconstruction, estimation of fractal dimension and Lyapunov exponents, and short-term prediction.

If the time series data appears to be from a low-dimensional chaotic source, then one can predict the "continuation" of the data in the short term, exploiting the fact that chaotic systems are fairly predictable in the short term. This is the "inverse problem" of dynamical systems. In this thesis, the technique of local fitting is used to accomplish the prediction. Finally the issue of noisy data is treated, with the purpose of highlighting where further research may be beneficial.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I would like to thank Professor Christopher Frenzen for his guidance in the selection of a thesis topic and in the preparation of this thesis. He developed the curriculum which stimulated my interest in the topic of chaos, and has spent several hours explaining necessary concepts and proofreading. He also suggested several interesting ideas for the development of this thesis.

I would also like to thank Major Phil Beaver, U.S. Army, for his assistance and guidance. He also provided me with many interesting ideas, and he also allowed me to peruse his vast collection of books on chaos for related material, and provided much appreciated word processing advice.

Thanks and appreciation to God, without whom life would be absent or devoid of meaning. Thanks to my mother, whose encouragement have helped me in all endeavors, academic and otherwise, thus far. Thanks also to Ziya and Cheryl, whose support was most appreciated and directly contributed to my ability to succeed at NPS.

I. INTRODUCTION

A. CHAOS AND DYNAMICAL SYSTEMS

1. Definitions and Examples

We start by considering a continuous-time, chaotic dynamical system, where the differential equations giving rise to the observed dynamics are known. Such a system can be expressed as

$$\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}(t))$$

where $\mathbf{x}(t) = [x_1(t),x_2(t),x_3(t),...,x_D(t)]$ represents the vector of dynamical state variables, and where $D \geq 3$. For the discrete time case, the dynamical system can be expressed as an invertible discrete time map of the form: $\mathbf{x}(t+1)=\mathbf{F}(\mathbf{x}(t))$, where $\mathbf{x}(t)$ is again the vector of dynamical state variables, but where $D \geq 2$. The number of degrees of freedom in a continuous time system is equal to the number of first order autonomous ordinary differential equations in the system. In a discrete time system, the number of degrees of freedom is the same as the number of components in the state vector $\mathbf{x}(t)$. The number of degrees of freedom a system has determines whether or not chaos could possibly exist in the system. The Poincare-Bendixson Theorem states that chaos cannot exist in continuous dynamical systems with only one or two dynamical degrees of freedom (Strogatz, 1994). Further, other authors have shown that for an invertible discrete time map, chaos cannot exist in a one-dimensional map. It is noted, however, that noninvertible maps in one dimension can exhibit chaos, as in the case of the logistic map, which will be discussed

later in this chapter. It is also noted that the function **F** must be nonlinear in order for the dynamical system to exhibit chaos.

What does it mean for a dynamical system to exhibit chaos? A commonly used definition is that a system exhibits chaos if the trajectory $\mathbf{x}(t)$ is aperiodic over the long term, and that it exhibits sensitive dependence on initial conditions. Such systems exhibit unpredictable behavior in the sense that for given initial conditions known with finite precision, the long term behavior cannot be predicted, except to say that the states are constrained to a certain finite region of state space, dictated by the existence of a *strange attractor*. Stated another way, two nearby initial conditions will exhibit an exponential divergence of their respective trajectories. A measure used to quantify how fast the two trajectories diverge is the largest *Lyapunov exponent*. Lyapunov exponents, and their calculation will be discussed more fully in Chapter III.

As a class of observable signals, chaos lies between the domain of predictable, regular, or quasi-periodic signals and the totally irregular stochastic signals, commonly called "noise", which are completely unpredictable. As a result, chaos has some interesting properties: namely that it is irregular in time and slightly predictable, and that it has structure in phase space. This structure is exhibited in its strange attractor. (Abarbanel, 1996)

A strange attractor is the geometrical structure formed by the asymptotic states of a chaotic system. Loosely speaking, an attractor is a subset of phase space which "attracts" phase points from other regions of phase space near the attractor. The region of phase space from which it attracts points is called the attractor's *basin of attraction*. Once

a phase point enters an attractor, it does not leave it (Farmer, 1982). On this attractor, the dynamics are characterized by stretching and folding. Stretching occurs because of the divergence of nearby trajectories and folding constrains the dynamics to a finite region in a subspace of dimension greater than or equal to n. This subspace is the smallest space which *embeds* the attractor. In contrast to non-chaotic systems which have attractors of integer dimensions (i.e. points and limit cycles), chaotic systems have strange attractors characterized by a non-integer dimension d (Kugiumtzis,1994). In fact, the term "strange" originally referred to the fact that the attractor has fractal dimension, and is therefore called a fractal set. Many definitions of dimension exist, and this subject will be treated more fully in Chapter III. The dimension of a dynamical system and its Lyapunov exponents are examples of *invariants* of the system: properties which exist that characterize the dynamical system, independent of any particular trajectory.

As an example of a strange attractor, consider the famous Lorenz system of differential equations, which now act as a standard set of equations for testing ideas in nonlinear dynamics. The equations are:

$$\dot{x} = -\sigma(y - x), \tag{1.1}$$
$$\dot{y} = -xz + rx - y, \tag{1.2}$$
$$\dot{z} = xy - bz\ . \tag{1.3}$$

A standard set of parameter values for equations (1.1) - (1.3) is $r = 28$, $b = 8/3$, $\sigma = 10$. With these parameter values, the orbits of the Lorenz system exhibit aperiodic, chaotic motion. Figure 1.1 depicts the phase portrait of the strange attractor upon which the orbits of the Lorenz system reside, for these parameter values. This figure was taken from Strogatz (1994, p. 332).

2. Observed Chaos and Time Series

With this basic understanding of chaotic dynamical systems, we now treat the issue of observed chaotic systems. For example, suppose that we sample a continuous time dynamical system (say the Lorenz equations) at time intervals τ_s starting at some time t_0, and that we take as data the scalar variable x(t). As a finitely sampled evolution, x(t) can be represented in the form:

$$x(t_0 + (n+1)\tau_s) \approx x(t_0 + n\tau_s) + \tau_s(\mathbf{F}(x(t_0 + n\tau_s)) \cdot \mathbf{e}_1),$$

where $\mathbf{e}_1$ is a unit vector in the x direction, and $\mathbf{F}$ in this specific case represents the dynamics of the Lorenz equations. The data composed of these sampled values of x(t) is our scalar-valued, univariate time series, which will be further denoted as {s(t)}, or equivalently, as {s(n)}, n = 1, 2,... ,N. This theory can be made general, i.e., with the choice of any continuous dynamical system and any choice for the state variable being sampled. Obviously, a discrete dynamical system would not require any sampling to be done since the successive iterates would already be in the above form with $\tau_s = 1$. Further, it is noted that one may quite likely obtain an {s(t)}, not through sampling of a known dynamical system, but by simply taking observations of some system of interest where the underlying equations are unknown.

In general, if one were observing some unknown system, it would be typical to observe only one or a few of the dynamical variables which govern the behavior of the system. As we will see, time series data collected by either sampling a dynamical system where the state equations are known, or obtained by simply taking observations of some

(unknown) dynamical system, can go a long way in helping us determine the nature of the system at hand.

In particular, one thing we can hope to accomplish by analyzing a "random looking" time series {s(t)} is to determine the nature of the system's dynamics, i.e., whether or not low-dimensional chaos is present. By low-dimensional we will mean that the underlying attractor's dimension is less than five; there is no explicit definition in the literature of what it means to be low-dimensional. If low-dimensional chaos is detected, we can then calculate the invariants of the attractor, reconstruct a representation of the attractor itself; and make short-term predictions of future values of {s(t)}. We emphasize the possibility of short-term prediction. If the system that we are observing is indeed chaotic, we know that orbits exhibit sensitive dependence on initial conditions. This means that unless we are able to observe and make predictions with an infinite amount of accuracy, our predicted trajectory is certain to diverge from the true trajectory, given enough time. Indeed, the impossibility of long-term prediction of a chaotic orbit is a hallmark of what it means to be chaotic. But a low-dimensional, chaotic system has enough "structure" to allow short term prediction.

In subsequent chapters and sections, we address pertinent questions such as: With regard to the analysis and short-term prediction of such time series, does it matter which state variable is observed? Can the observations of more than one state variable be used? What is the effect of sampling time τ_s? For now, though, we turn to the more general topic of time series and traditional approaches to analyzing them.

B. APPROACHES TO TIME SERIES ANALYSIS

1. Linear Stochastic Approach (AR, MA, ARMA)

This section does not propose to be a comprehensive discussion of time series analysis; rather, it serves to highlight a few of the more common approaches to the subject. The primary reference for this section is Gershenfeld/Weigend (1994).

We first consider a linear, stochastic approach to time series analysis. The approach we will discuss, namely the Box-Jenkins class of models, assumes that the dynamics underlying the time series is stationary and linear, with a stochastic element. This class of models is useful when one has a time series in which there is no apparent seasonal trend, where there is dependence between present and past values of the time series, and where the time order of the observations is taken into account. Clearly, this assumption of dependence, with no apparent trend, would be appropriate for the study of many dynamical systems and their related observed time series. These models, once built, can be used for predictive purposes. The accuracy of the predictions, of course, depends on the appropriateness (goodness of fit) of the type and order of the model to the time series at hand. (Chatfield, 1996)

Linear time series models have some desirable features, namely that they are relatively straightforward to implement and that they can be understood in great detail. However, it is important to note that this class of models may, of course, be completely inappropriate for the time series at hand, i.e., for one that has arisen from a nonlinear source. A basic assumption of the linear time series analysis discussed here is that the dynamics are time invariant, that is, that the system exhibits stationary dynamics.

Let us first discuss one type of linear model, the Moving Average (MA) model. Given the time series {s(t)}, a moving average model implies that the current value of the time series depends on the mean value of the time series, as well as the current and past q values of the Gaussian error terms. Each error term ε_t represents the difference between the time series value s(t) and the mean value μ. The fact that the current and past q values of the error terms appear in the model allows for the modeling of dependence between time series values. The model takes the form:

$$s_t = \mu + \sum_{j=0}^{q} b_j \varepsilon_{t-j}.$$

The ε_t are assumed to be uncorrelated radom variables with mean zero and finite variance σ^2, and the b_j are coefficients. The ε_t are usually scaled so that $b_0 = 1$. The model implies that

$$E(s_t) = 0 \quad \text{and} \quad Var(s_t) = \sigma^2 \sum_{j=o}^{q} b_j^2.$$

The value for q is determined by the user, with the resulting model being called a qth-order moving average model MA(q). Several techniques exist for determining the appropriate order of the model (Chatfield, 1996).

Another type of linear model is called the Autoregressive (AR) Model. Here, the model is represented by

$$s_t = \sum_{i=1}^{p} a_i s_{t-i} + e_t.$$

This is called a pth-order autoregressive model (AR(p)), and it implies that the current value for s(t) depends on the p past values of the {s(t)} and a Gaussian error term. This model is much like a multiple regression model, but s_t is regressed not on independent variables but on past values of s_t; hence the prefix 'auto'.

The next step in complexity, and hence the next type of linear model we consider, is one having both AR and MA parts. This is called an ARMA(p,q) model, with p and q being the orders of the AR and MA parts, respectively. The model takes the following form:

$$s_t = \sum_{i=1}^{p} a_i s_{t-i} + \sum_{j=0}^{q} b_j e_{t-j}.$$

ARMA models have dominated all areas of time series analysis and discrete-time signal processing for more than half a century. If a linear model is good, it transforms the signal into a small number of coefficients plus residual white noise. Fitting the coefficients of a linear model to a given time series and selecting the order of the model are crucial, and involve studying the power spectra and autocorrelation coefficients of the time series. Many good books exist which treat this subject, including Chatfield (1996) and Box/Jenkins (1976).

The important idea to take away from the study of linear time series models is that ARMA coefficients, power spectra, and autocorrelation coefficients contain the same information about a linear system that is driven by uncorrelated white noise. Therefore, only if the power spectrum is a useful characterization of the relevant features of a time series will an ARMA model be a good choice for describing it. However, this type of model can fail to model the dynamics of even simple nonlinearities, if those nonlinearities lead to complicated power spectra. Two time series can have very similar broadband spectra but may be generated from systems with very different properties, such as a linear system that is driven stochasticallly by external noise, and a deterministic (noise-free) nonlinear system with a small number of degrees of freedom. Taking an example from

Gershenfeld/Weigend (1994), let us consider the discrete, one-dimensional logistic map given by

$$x(n+1) = r\,x(n)(1-x(n)) \tag{1.4}$$

with parameter r = 4. They point out that although equation (1.4) is completely deterministic, it can easily generate time series with broadband power spectra. In the context of an ARMA model, a broadband component in a power spectrum of the output must come from external noise input to the system, but here it arises in a one-dimensional system as simple as equation (1.4). Therefore, even simple nonlinearities demonstrate that linear time series analysis can be inappropriate and ineffective for the time series at hand.

2. Nonlinear Stochastic Approach

In 1990, Tong (Tong, 1990) took an important step beyond linear models for prediction. He was the first to suggest the use of two globally linear functions, instead of only one. The resulting model, called a threshold autoregressive model (TAR), is globally nonlinear. Specifically, a TAR model chooses one of two local linear autoregressive models based on the value of the system's state. From here, the next step is to use many local linear models; however, the number of such regions that must be chosen may be very large if the system has even simple nonlinearities (such as equaiton (1.4)). A natural extension of the form of the ARMA model includes quadratic and higher order powers in the model; this is called a Volterra series (Volterra,1959).

TAR models, Volterra models, and their extensions do much to expand the scope of possible methods to model time series, but these come at the expense of the simplicity with which linear models can be understood and fit to data. For nonlinear models to be useful, there must be a process that uses the data to guide and restrict the construction of

the model. Lack of insight into this problem has limited the use of nonlinear, stochastic time series models. Further, this method neglects the fact that the irregular, stochastic component of the time series may be largely deterministic, so predictions would be less accurate using this stochastic approach than they would be if a nonlinear, deterministic approach were used.

3. Multivariate Adaptive Regressive Splines (MARS)

Multivariate Adaptive Regression Splines is a new methodology, due to Friedman, for nonlinear regression modeling. The MARS procedure is based on a generalization of spline methods for function fitting. Splines have been extensively studied and have may desirable properties. The basic underlying assumption is that the function to be estimated is locally relatively smooth, where smoothness is adaptively defined depending on the local characteristics of the function. In this manner, MARS addresses the general problem of modeling and interpreting a general predictive relationhip between "current" value for s(t) and previous values of s(t). This method is designed especially to handle the task of flexible regression modeling of high dimensional, and possible noisy, data. For more details about this method, we direct the reader to Friedman (1991) and Gershenfeld/ Weigend (1994).

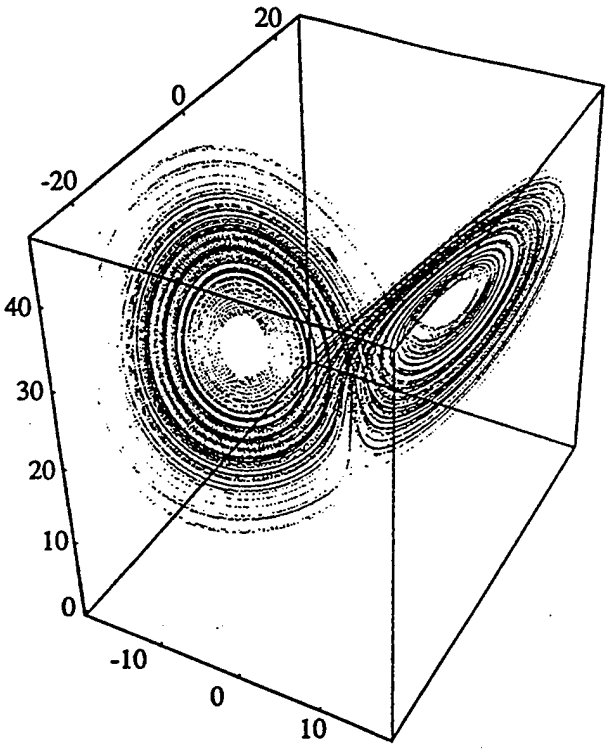

Figure 1.1. Lorenz attractor. From Strogatz (1994, p. 332).

II. METHOD OF STATE SPACE RECONSTRUCTION

A. THEORY OF STATE SPACE RECONSTRUCTION

In contrast to the time series analysis methods discussed in Chapter I, the method of state space reconstruction is a nonlinear, deterministic approach: one that would be fitting for a time series derived from a chaotic process. We are interested, then, in time series that arise from observations of a deterministic dynamical system, such as a set of differential equations. One goal of state space reconstruction is to use the immediate past behavior of the time series to reconstruct the current state of the system (Casdagli, et al., 1992). As with the other methods previously discussed, state space reconstruction assumes that the dynamics producing the time series is stationary.

Why would one want to reconstruct the state space? A good reconstruction can be used to predict future values of the time series, calculate invariants of the dynamics, and reconstruct an attractor which is diffeomorphic to the attractor in the space of the original dynamical variables. One of the main accomplishments of this method is to convert the problem of prediction from a problem of extrapolation of the time series into a problem of interpolation within the state space.

Two methods exist to accomplish state space reconstruction. One method, the principal components technique, will not be discussed or demonstrated in this thesis, except to say that it is a standard procedure in signal processing and was first applied to chaotic dynamical systems by Broomhead and King (Gershenfeld/Weigend, 1994). The method that we will employ in this thesis, and which is currently the most widely used

choice, is the method of delay coordinates. It is implemented as follows. Suppose that we have an underlying (but possibly unknown) dynamical system $\mathbf{x}(n+1) = \mathbf{F}(\mathbf{x}(n))$, where $\mathbf{x}(n)$ is the state vector in a multidimensional phase space. Note that this system is discrete and can represent either a discrete time map or a continuous time system sampled at finite intervals. Suppose we have a time series consisting of discrete scalar observations of the dynamical system which we denote by $\{s_n\} = \{s(n\tau_s)\}$, where n = 0, 1, ..., N. The time series $\{s(n)\}$ is related to the state vector of the underlying dynamical system by $s(n) = h(\mathbf{x}(n\tau_s))$, where h is known as the *measurement function* (Kugiumtzis, 1994). We can create a state vector $\mathbf{y}(n)$ by assigning coordinates

$$y_1(n) = s(n),\ y_2(n) = s(n-\tau),\ldots,\ y_d(n) = s(n-(d-1)\tau),$$

where τ is the *delay parameter*, and d is the *embedding dimension*. Note that τ must be an integer. When $\tau > 1$, we skip samples during the reconstruction. For instance, if $\tau_s = 0.01$, and a delay parameter τ of five is used, then one "skips" over the data corresponding to .02, .03, .04, and .05 when constructing the state vectors. (Casdagli et. al., 1991)

What is accomplished by such an embedding? One answer is that since the dynamics are unknown, we cannot reconstruct the original attractor that gave rise to the observed time series. Instead, we seek an embedding space where we can reconstruct an attractor from the scalar data that preserves the invariant characteristics of the original unknown attractor. This approach to reconstructing the attractor is based on Takens' Theorem, which states that for an infinite amount of noise-free data and a smooth choice of $h(\cdot)$, one can always find an embedding dimension d which preserves the invariants of

the dynamical system. Takens proved that under these conditions it is sufficient to choose $d \geq 2\,d_A+1$, where d_A is the dimension of the attractor and d is an integer. Such a value for d is called a "*sufficient*" d. Takens' theorem guarantees that the attractor embedded in the d-dimensional state space (with sufficient d) is "unfolded" without any self intersections. The condition $d \geq 2\,d_A+1$ is sufficient, but not necessary, and an attractor may be reconstructed successfully with a lower embedding dimension, perhaps as low as $d_A + 1$. This value for d is called the "*necessary*" d, and will vary from system to system. Exactly how to go about reconstructing the attractor will be discussed in Chapter III.

As for the choice of τ, with an infinite amount of noise-free data, the actual value of τ is irrelevant; however, in practice one will never have an infinite amount of noise-free data, so its value is important. With a finite amount of noisy data, the estimates of the invariant measures d_A and the Lyapunov exponents are found to depend on both d and τ. So the choices for d and τ are crucial to the reconstruction. Methods for choosing d and τ will be discussed in subsequent sections.

The state vectors $\mathbf{y}(n)$ serve to replace the scalar data measurements $\{s(n)\}$ with data vectors in Euclidean d-dimensional space, in which the invariants of the sequence of points $\mathbf{x}(n)$ are represented with as much accuracy as in the original system. The new space, and hence the reconstructed attractor in the new space, is related to the original space of the $\mathbf{x}(n)$ by smooth, invertible transformations. Specifically, the reconstructed attractor exhibits the same topological characteristics and geometrical form as the original attractor (Crutchfield et al., 1980). This means that we can utilize the reconstructed state

space to learn as much about the system using the observations {s(n)} as we could if we had been able to use the "true" **x**(n) values. (Abarbanel, 1996)

We emphasize that Takens' Theorem allows any smooth choice of h(·), the measurement function. That is, although we will demonstrate these ideas in this thesis by utilizing a time series consisting of samples of the scalar dynamical variable x(t) of the Lorenz system (equations (1.1) - (1.3)), one could actually utilize any smooth function of the three dynamical variables of this system to comprise the time series. For example, one might try $\cos(x^2(n\tau_s) + y^2(n\tau_s))$, with n = 1, 2, ..., N to generate the time series.

B. CHOOSING THE DELAY PARAMETER

1. Theory

As stated previously, if one had an infinite amount of clean data, the choice of delay parameter τ would be unimportant. However, since we will be dealing with finite-length, possibly noise-contaminated data, the choice of τ is important, so we will need some prescription for determining it. As we will see, poor choices for τ may lead to inaccurate short-term predictions and estimates of invariants, as well as distorted pictures of the attractor. In particular, if τ is chosen too small, the coordinates s(n) and s(n+τ) will not be "independent" enough. This basically means that not enough time has passed from the observation of s(n) to s(n+τ) to capture any "new" information in the coordinate s(n+τ). Thus, we will not see any of the dynamics unfold during this period of time. Further, the plot of the reconstructed attractor will appear "stretched out" in the

$y_1(n) = y_2(n) = y_3(n) =$... direction. This problem with short time intervals can be remedied by taking differences between coordinates and dividing by the time interval (analogous to taking derivatives), but this procedure introduces noise into the phase space picture (Crutchfield et. al., 1981).

If τ is chosen too large, s(n) and s(n+τ) are "too independent"; in other words, we take the risk that they appear "random" with respect to each other. Further, we can no longer make a one-to-one correspondence with points of the time series and points on the original attractor. Therefore, we seek a value for τ which is large enough for s(n) and s(n+τ) to be independent enough so as not to give redundant information, but not so large that they are completely independent, in a statistical sense. Many authors treat this subject in the literature (with many different interesting ideas), but we choose to illustrate the approach based on the concept of looking for the first minimum of the average mutual information function, presented by Abarbanel (1996). This method is related to other popular ideas for determining the delay parameter, including use of the generalized correlation integral (see Liebert/Schuster (1989)).

2. Chaos as a Source of Information

We have already mentioned that a hallmark of chaos in a dynamical system is its "sensitivity to initial conditions." In particular, two nearby initial conditions (points in the state space) move apart at an exponential rate, determined by the most positive Lyapunov exponent λ. Suppose that we have a fixed, finite resolution in our state space. This means that below a certain tolerance α, we cannot tell points apart, i.e., if $\mathbf{x}_1(t_0)$ and $\mathbf{x}_2(t_0)$

are points in the state space, and $0 < |\mathbf{x}_1(t_0) - \mathbf{x}_2(t_0)| < \alpha$, then the two points will appear to us as the same point from our perspective in the finite resolution state space. However, in our chaotic system the orbits $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ will evolve and begin to move apart as time passes. Specifically, at a time $t' > t_0$, the distance between the two points has grown to

$$|\mathbf{x}_1(t') - \mathbf{x}_2(t')| = |\mathbf{x}_1(t_0) - \mathbf{x}_2(t_0)| \exp(\lambda |t' - t_0|), \qquad (2.1)$$

where λ is the largest Lyapunov exponent, and is necessarily positive for a chaotic dynamical system. When this distance exceeds α, we are then able to distinguish between the two points $\mathbf{x}_1(t')$ and $\mathbf{x}_2(t')$. In this sense, we have uncovered information about the population of the state space, thanks to the instability or chaos present in the system. We begin to see even in this simple example how the magnitude of the largest, positive Lyapunov exponent measures the rate at which the system creates or destroys information. For this reason, λ is sometimes measured in bits of information per data sample. When λ is measured in bits of information per data sample (call it λ^*), it is related to the λ given in equation (2.1) in the following way:

$$\lambda^* = \log_2[\exp(\lambda \tau_s)].$$

Here, we have seen that λ, the largest positive Lyapunov exponent, plays an important role in the generation of information. If λ had been negative or zero, no such information would have been uncovered. We take a brief excursion here to discuss and understand Lyapunov exponents more fully. In a nonlinear system with D degrees of freedom, there are D Lyapunov exponents. Sometimes methods exist for finding them analytically (see Strogatz (1994)); we will discuss how to determine them numerically from experimental time series data in Chapter III.

In a system with a chaotic attractor, the sum of these D Lyapunov exponents is negative, which dictates that volumes of phase space contract under the flow determined by the system. Further, a system that is chaotic will have a positive Lyapunov exponent, in order to satisfy the "sensitive dependence on initial conditions" property. Also, in any continuous-time dynamical system (such as a system of one or more differential equations), one Lyapunov exponent must be zero (Wolf et.al., 1984). As a result, we know that we must have at least one negative Lyapunov exponent, in order to get a sum of exponents which is negative. All of this further explains how an attractor is formed as a result of the stretching (explained by the positive Lyapunov exponents) and the folding (explained by the negative Lyapunov exponents and the dissipative nature of phase space volumes) of orbits. This also explains why a minimum of three dimensions is necessary for chaos to exist in a continuous time-dependent system: we need one exponent to be positive, so that chaos is possible, one is zero because we have a differential equation as the source of the dynamics, and a third exponent must exist in order for the sum of all three to be negative. It is noted that for invertible discrete maps, there is no zero exponent, so we need only two dimensions for chaos to exist: one whose Lyapunov exponent is positive and one whose Lyapunov exponent is negative.

We now return to the prescription for choosing τ. The primary reference for this prescription is Abarbanel (1996). We start with a definition. The *mutual information* between measurement a_i drawn from a set $A = \{a_i\}$ and measurement b_j drawn from a set $B = \{b_j\}$, is the amount learned from the measurement of a_i about the measurement of b_j.

Measured in bits, this quantity is

$$\log_2\left[\frac{P_{AB}(a_i,b_j)}{P_A(a_i)P_B(b_j)}\right],$$

where $P_{AB}(a,b)$ is the joint probability density for measurements A and B resulting in values a and b. $P_A(a)$ and $P_B(b)$ are the individual probability densities for the measurements of A and of B. In a deterministic system, these probabilities can be estimated by constructing a histogram of the a_i and the b_j and using the ratio of frequency of occurence to the number of observations as the estimated probability.

Some insight into mutual information may be gained by noting that if the value a_i is independent of b_j, then $P_{AB}(a,b) = P_A(a)\, P_B(b)$, and the amount of information between the two measurements is zero, as it should be. The *average mutual information* between a set A of measurements and a set B of measurements is

$$I_{AB} = \sum_{a_i,b_j} P_{AB}(a_i,b_j)\, \log_2\left[\frac{P_{AB}(a_i,b_j)}{P_A(a_i)P_B(b_j)}\right].$$

This quantity is strictly a set theoretical idea which establishes a criterion for the mutual dependence of two sets of measurements based on the idea of the information connecting them. We can now use this idea to give a more quantitative definition of the independence of s(n) and s(n+τ).

If we take as the set of measurements A the values of the observable {s(n)}, and for the set B we take the values of {s(n+τ)}, then the average mutual information betwen these two measurements is

$$I(\tau) = \sum_{s(n),s(n+\tau)} P(s(n),s(n+\tau))\log_2\left[\frac{P(s(n),s(n+\tau))}{P(s(n))P(s(n+\tau))}\right]. \qquad (2.2)$$

Note that I(τ) is always zero or positive. When τ becomes large, the chaotic behavior of the system makes the measurements s(n) and s(n+τ) become nearly independent in a

practical sense, and $I(\tau)$ will tend to zero. The principle of choosing τ to be the first minimum of $I(\tau)$ seems to correspond well with wanting $s(n)$ and $s(n+\tau)$ to be independent enough, but not too independent. Abarbanel points out that the average mutual information function reveals a different aspect of the system than does the more familiar linear autocorrelation function, and is obviously more appropriate for data arising from nonlinear sources. A nice property of equation (2.2) is its invariance under smooth changes of coordinate system. This means that the quantity $I(\tau)$ is the same whether evaluated in time delay coordinates or in the original (but sometimes unknown) coordinates.

An S-PLUS (Version 3.3) routine for calculating the average mutual information function is given in Appendix A. It was used on a time series of 1000 samples of the x(t) variable from the Lorenz equations, with $\tau_s = 0.01$. The time series was generated in MATLAB (Version 4.2) using *ode45* (a Runge-Kutta 4th and 5th order ordinary differential equation solver), and the initial condition [5, 5, 5]. The result of applying the S-PLUS routine to the Lorenz data is shown in Figure 2.1. We see that a delay parameter of five should be chosen, since it corresponds to the first minimum of the average mutual information function. When running the S-PLUS routine, it is necessary to either specify the breaks for the histograms before invoking the S-PLUS *hist* and *hist2d* functions, or to let the S-PLUS software determine the appropriate breaks. A break in a one-dimensional histogram is a segment of the horizontal axis. The horizontal axis represents possible values of the observable, and the vertical axis is used to plot the number of data points which fall into a particular break. The set of breaks for the

histogram should be mutually exclusive (no overlaps) and should include all possible values for the observable. We found that using small, finely-spaced, breaks gave approximately the same results (within plus or minus one of the value for τ) as when we let the S-PLUS functions decide where the breaks should be. By "small", we mean that we used an individual break size which corresponded to approximately five percent of $|\max(\{s(n)\}) - \min(\{s(n)\})|$.

C. CHOOSING "NECESSARY" EMBEDDING DIMENSION

We already know that choosing an embedding dimension $d \geq 2d_A + 1$ will be sufficient to reconstruct the state space, but we seek the minimum d that will serve this purpose, that is the minimum d that will ensure that the attractor, when reconstructed in this space, is completely "unfolded". Such a d is called the "necessary" embedding dimension. What if d is incorrectly chosen? Takens showed that any d greater than or equal to the sufficient d succeeds in unfolding the attractor, because once the attractor is unfolded, it remains unfolded for all higher embedding dimensions. However, we want to use as small an embedding dimension d as possible to ensure accuracy of predictions. Predictions using a local approximation method will be illustrated in Chapter IV. We will see that predictions get worse as the embedding dimension is increased beyond the necessary embedding dimension (May/Suigihara, 1990).

Many different approaches exist in the literature, but we choose to demonstrate the technique of global false nearest neighbors presented by Abarbanel (1996). It is necessary to point out that the d obtained in this manner, which we further denote as d_E, is a *global*

unfolding dimension and may be different from the local dimension of the underlying dynamics. To illustrate the difference, consider a Mobius strip. The global embedding dimension would be three, but locally the Mobius strip is two-dimensional. We seek the global unfolding dimension when recontructing the state space.

Suppose that we have reconstructed the state space using dimension d, and have created the data vectors $\mathbf{y}(k) = [s(k), s(k-\tau),...,s(k-(d-1)\tau)]$ using the time delay suggested by the average mutual information technique. Using Euclidean distance as our norm, we examine the nearest neighbor in phase space to the vector $\mathbf{y}(k)$. This will be a vector

$$\mathbf{y}^{NN}(\underline{k}) = [s^{NN}(\underline{k}), s^{NN}(\underline{k}-\tau),...,s^{NN}(\underline{k}-(d-1)\tau)],$$

and its time label is not necessarily related to the time k at which the vector $\mathbf{y}(k)$ appears. However, $\underline{k}$ is a function of k. We now reason that if the vector $\mathbf{y}^{NN}(\underline{k})$ is truly a neighbor of $\mathbf{y}(k)$, then it came into the neighborhood of $\mathbf{y}(k)$ because of the dynamics of the system involved. It is either the vector just ahead or just behind $\mathbf{y}(k)$ along the orbit, if the time steps along the orbit are small enough, or it arrived in the neighborhood of $\mathbf{y}(k)$ through evolution along the orbit and around the attractor. Since attactors are generally quite compact in phase space, each phase space point will have numerous neighbors, provided one has enough data points to populate the state space well.

If the vector $\mathbf{y}^{NN}(\underline{k})$ is a false neighbor of $\mathbf{y}(k)$, having arrived in the latter's neighborhood by projection from a higher dimension because the present dimension d does not unfold the attractor, then by going to the next dimension d + 1 we stand a good chance of moving this false neighbor out of the neighborhood of $\mathbf{y}(k)$. In this way, starting with a dimension d, we can look at every data point $\mathbf{y}(k)$ and its nearest neighbor

$\mathbf{y}^{NN}(\underline{k})$ and determine the dimension at which we remove all false neighbors. At that point, we will have determined the minimum dimension d_E at which the attractor is unfolded.

In order to implement this procedure, we need a method for determining change in the distance between a point $\mathbf{y}(k)$ and its nearest neighbor $\mathbf{y}^{NN}(\underline{k})$ when moving to dimension d + 1. As we change from dimension d to d + 1, the additional component of $\mathbf{y}(k)$ is $s(k+d\tau)$ and the additional component of $\mathbf{y}^{NN}(\underline{k})$ is $s^{NN}(\underline{k}+d\tau)$. In order to determine whether $\mathbf{y}(k)$ and $\mathbf{y}^{NN}(\underline{k})$ are near or far in dimension d + 1, we need only compare $|s(k+d\tau) - s^{NN}(\underline{k}+d\tau)|$ with the distance between nearest neighbors in dimension d. If the term occuring in dimension d + 1 is large compared to the distance in dimension d between nearest neighbors, then we have found a false neighbor. If it is not large, we have a true neighbor.

The square of the Euclidean distance between the nearest neighbor points in dimension d is

$$R_d(k)^2 = \sum_{m=1}^{d} [s(k+(m-1)\tau) - s^{NN}(\underline{k}+(m-1)\tau)]^2,$$

while in dimension d + 1 it is

$$R_{d+1}(k)^2 = \sum_{m=1}^{d+1} [s(k+(m-1)\tau) - s^{NN}(\underline{k}+(m-1)\tau)]^2.$$

The distance between points when seen in dimension d + 1 relative to the distance in dimension d is

$$\sqrt{\frac{R_{d+1}^2(k) - R_d(k)^2}{R_d(k)^2}} = \frac{|s(k+d\tau) - s^{NN}(\underline{k}+d\tau)|}{R_d(k)}.$$

When this quantity is larger than some appropriate threshold, we have a false neighbor. If the number of data points is enough to adequately populate the attractor, the

determination of true vs. false neighbor is fairly insensitive to the value of the threshold. We note that the search for nearest neighbors among a large number of data points is computationally cumbersome. Many authors suggest using a kd-tree search scheme, which takes $O(N \log N)$ operations, for N data points (Farmer/Sidorowich, 1987).

If we have clean data from a chaotic system, the percentage of false nearest neighbors will drop from nearly 100% in dimension one to strictly zero when d_E is reached. A MATLAB scheme which implements the above procedure is given in Appendix B. It does not utilize a kd-tree search, however. Figure 2.2 depicts the result of running this program on the Lorenz data series, for N = 1000 and a threshold of ten. Figure 2.3 depicts the result of running this program for N = 500 and a threshold of five. As expected, increasing the number of data points used allows one to use a "less discriminating" threshold. In both cases, though, we see that $d_E = 3$ for the Lorenz system. We note that other effective algorithms, which use different criteria, exist in the literature for accomplishing this same task of determining when neighbors are "near" or "far" in dimension d + 1 (see Fitzgerald et. al., (1996)).

Now that we have a method for choosing the minimum embedding dimension and delay parameter for a state space reconstruction, and assuming that we have a low-dimensional, chaotic time series, we can expect to use our correctly-reconstructed state space to predict future values of the time series, calculate invariants of the dynamics, and reconstruct the underlying attractor. Before we attempt to accomplish these, however, we should address how to determine whether or not the time series we have is actually low-dimensional and chaotic. This will be the topic of Chapter III.

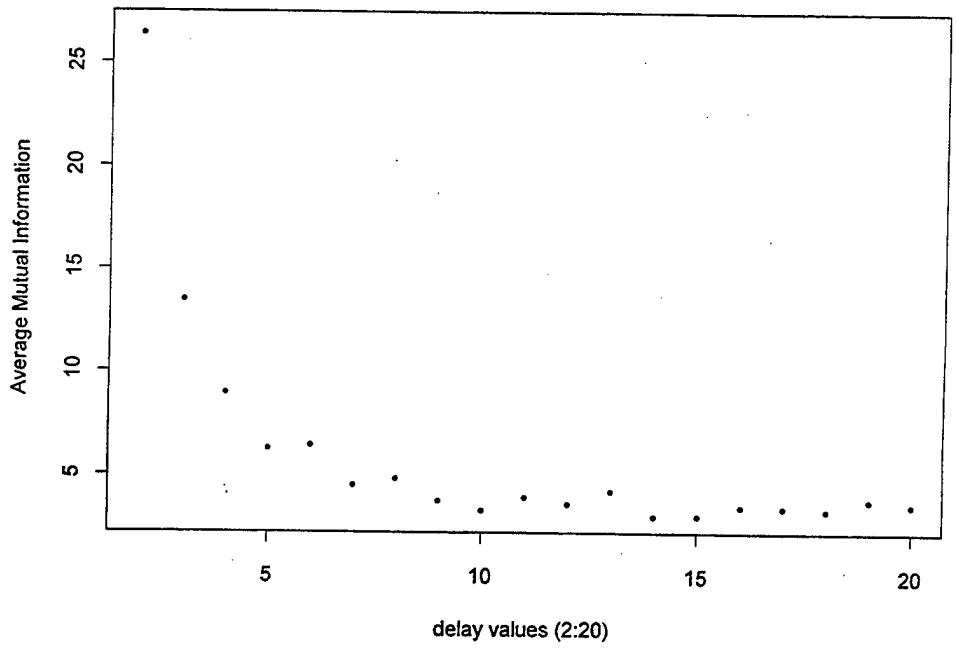

Figure 2.1. Average Mutual Information in S-PLUS (Lorenz)

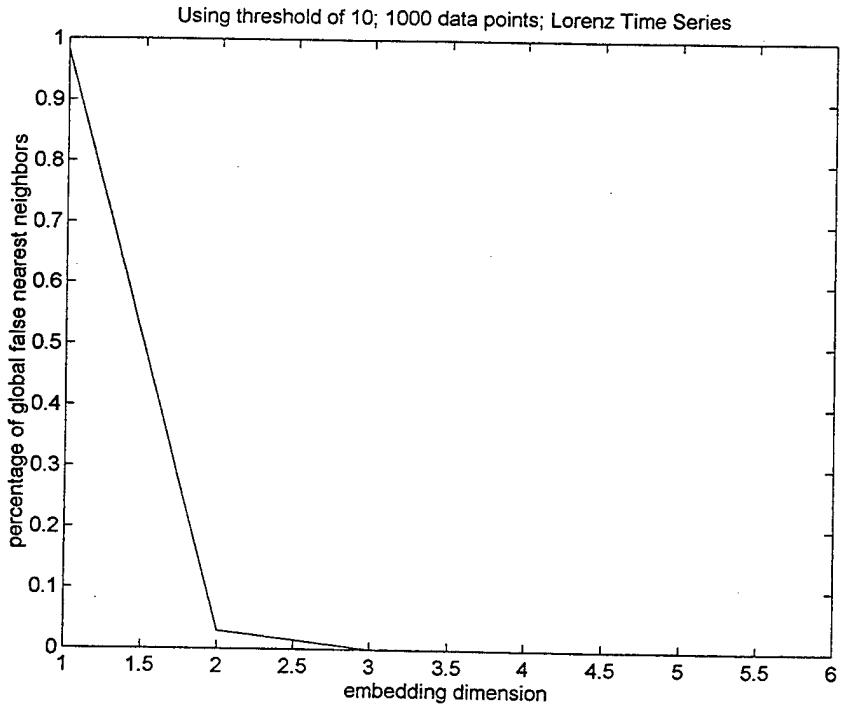

Figure 2.2. Global False Nearest Neighbors in MATLAB (Lorenz)

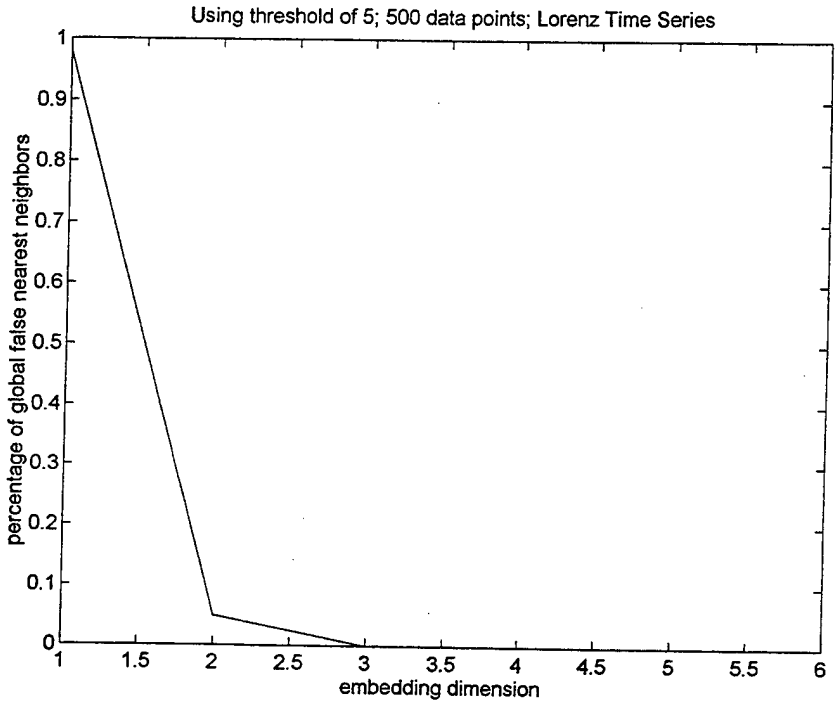

Figure 2.3. Global False Nearet Neighbors in MATLAB (Lorenz)

III. TECHNIQUES FOR DISTINGUISHING BETWEEN LOW-DIMENSIONAL CHAOS AND RANDOMNESS

A. RECONSTRUCTING PHASE PORTRAITS

Suppose we have an aperiodic, irregular-looking time series $\{s(n)\}$, and we want to determine whether the time series arises from a random (stochastic) process or whether the apparent "randomness" is due to the presence of low-dimensional chaos. Recall that chaotic behavior causes orbits to exhibit long-term aperiodicity, so data arising from this type of orbit, though deterministic, may appear "random" to the eye, and perhaps to other more quantitative statistical tests of randomness.

We will now discuss several methods to determine whether the time series arises from a low-dimensional, chaotic process or not. We emphasize, however, that these methods are designed for detecting *low-dimensional* chaos. That is, if the time series arises from a high-dimensional, chaotic source, it will not be possible to distinguish it from a randomly-generated time series using these methods. In fact, most pseudo-random number generators utilize a high-dimensional, chaotic (but still deterministic) dynamical system; and the numbers they produce are therefore called "pseudo-random" (Flandrin/ Michel, 1990).

The first method we discuss involves reconstructing the phase portrait. After one chooses the correct minimal necessary embedding dimension d_E and delay parameter τ, and constructs the state vectors $\mathbf{y}(n)$, one can plot these state vectors as d_E-dimensional points in the phase space (which, recall, is diffeomorphic to the original phase space of dynamical variables). When d_E is three or less, the space is obviously easily viewed.

However, if d_E is greater than or equal to four, one may have to look at "slices" of the multidimensional attractor. The purpose of constructing such a plot is to visually detect the existence of "order". If the time series arises from a low-dimensional, chaotic source, we expect to see evidence of an attractor. However, if the time series is stochastic in nature, then we would expect to see no such indication of "order".

A phase portrait recontruction is demonstrated in Figure 3.1 with 2000 points of the time series generated by the Lorenz equations (1.1) - (1.3). Using $\tau = 5$ and $d = 3$, we see a picture which is clearly diffeomorphic to the familiar "butterfly wings" of the Lorenz strange attractor. To illustrate the fact that using a delay parameter which is either too small or too large will distort the phase portrait, we depict in Figure 3.2, Figure 3.3, and Figure 3.4 the result of using a delay parameter equal to one, ten, and seventeen, respectively. Note that a delay parameter of one is too small, and stretches the attractor in the $x = y = z$ direction, as mentioned in Chapter II. Also note that using a delay parameter much greater than five begins to make the attractor unrecognizable. Recall that an incorrectly chosen delay parameter will make calculations of invariants and short-term prediction less accurate.

As stated in Chapter II, any embedding dimension greater than or equal to the minimal necessary embedding dimension will unfold the attractor. For the Lorenz time series, we try using an embedding dimension of two to illustrate what happens when a lower embedding dimension than the minimal necessary embedding dimension is used. Figure 3.5 depicts the result: a reconstructed attractor which crosses itself. Of course, our "three dimensional" plots look similar (with crossings of the orbits), but that is the result

of representing a three-dimensional phase portrait on a two-dimensional piece of paper. Using an embedding dimension of one gives a plot of points crossing back and forth on a straight line- which also is clearly not representative of the unfolded attractor which exists for $d \geq 3$.

In order to compare the result of reconstructing the phase portrait for a low-dimensional, chaotic system to the result of doing the same on a pseudo-randomly-generated time series, we used S-PLUS to generate a random sequence of 1000 numbers. The Uniform (-15, 15) distribution in S-PLUS was used to generate the sequence. Figure 3.6 shows the resulting phase portrait when using $d = 2$ and $\tau = 1$. In Figure 3.7 we used $d = 3$ and $\tau = 1$, and in Figure 8 we used $d = 3$ and $\tau = 2$. None of these phase portraits exhibit any order, so based on using this technique, we would conclude that we do not have a time series from a low-dimensional, chaotic source.

B. ESTIMATING FRACTAL DIMENSION OF A HYPOTHESIZED STRANGE ATTRACTOR IN A RECONSTRUCTED STATE SPACE

Although the technique of reconstructing the phase portrait provides a good start in determining the presence of an attractor, we should go one step further and actually estimate the fractal dimension of such an attractor (d_A), given the data contained in the time series. For instance, we may be able to detect "order" in the phase portraits, but it may be difficult to determine whether or not we have unfolded the attractor, since it may be difficult to determine whether the orbits are still crossing. We have already seen that we will view "crossings" that may not actually exist when viewing a three-dimensional phase

portrait in two dimensions. One benefit of estimating the fractal dimension of the underlying attractor is that we can ensure we will know the exact minimal embedding dimension: it will be the integer ceiling of d_A. Also, using a popular "rule of thumb," if we have an estimate of d_A which is greater than five, then we might conclude that the time series is not low-dimensional, (Brock, 1986).

Another benefit of estimating the fractal dimension of the attractor is based on the following result given in Flandrin/Michel (1990). If the time series is the result of a low-dimensional chaotic process, then the estimated dimension will converge to the true attractor dimension once the correct embedding dimension is reached. If the time series is the result of a stochastic (or high-dimensional, chaotic) process, then the estimated dimension will continue to increase as the embedding dimension increases, i.e., no such convergence will occur.

In order to discuss the estimation of dimension, it is first necessary to note that there are many different definitions of dimension. Our discussion on dimension is largely summarized from Kugiumtzis et. al. (1994) and Strogatz (1994). The first definition of a fractional dimension was presented in 1919 by Hausdorff, but it is in rather abstract form, and therefore not suitable for practical computation. Closely related to the *Hausdorff dimension* is the box counting dimension. We imagine "covering" the attractor with boxes, cubes, or hypercubes. Let M(l) be the number of hypercubes of a given dimension m with side length l required to cover the attractor. Then from the scaling law

$$M(l) \sim l^{-D_1}$$

the *fractal (box counting) dimension* D is derived to be

$$D_1 = \lim_{l \to 0} \frac{-\log[M(l)]}{\log l}$$

A nice property of the box-counting dimension is that the dimensions of a point, line, and area in two-dimensional space are the usual values of zero, one, and two, respectively. Although efficient algorithms exist to calculate the box counting dimension D_1, it has been shown to have a number of practical limitations. For example, in the definition of the box counting dimension, the distribution of points on the attractor was not taken into consideration, only the geometrical structure of the attractor. This means that all cubes contribute equally to the calculation of dimension even though the frequencies with which they are visited by points on the attractor may be very different.

One way that the distribution of points on the attractor can be taken into account is by use of the *information dimension* which is derived from the minimal information $S(\varepsilon)$ needed to specify a point in a set, such as a hypercube, to an accuracy ε. This information is defined to be

$$S(\varepsilon) = -\sum_{i=1}^{M(\varepsilon)} p_i \log p_i \ ,$$

where p_i is the probability of a point being in the ith set, defined as $p_i = \mu_i / N$ where $N \rightarrow \infty$ and μ_i is the number of points in the ith set. The information dimension D_2 of the attractor is then defined by

$$D_2 = \lim_{\varepsilon \rightarrow 0} \frac{-S(\varepsilon)}{\log \varepsilon} \ .$$

The most popular measure of an attractor's dimension is the *correlation dimension* D_3, first defined by Grassberger and Procaccia. It can be considered as a simplification of the information dimension and is given by

$$D_3 = \lim_{\varepsilon \rightarrow 0} \frac{\log \langle \mu_i \rangle_x}{\log \varepsilon} \ ,$$

where μ_i is defined as before for a ball with center $\mathbf{x}_k$, instead of a cube; and $\langle\cdot\rangle_x$ denotes the average over all points $\mathbf{x}_k$ on the attractor. The correlation dimension also accounts for the distribution of points on the attractor. To estimate D_3, one plots log $\langle\mu_i\rangle_x$ vs. log (ε). We should find an intermediate range of ε over which the graph approximates a straight line. The slope of this line is an estimate of D_3.

The general rule $D_3 \geq D_2 \geq D_1$ holds for the three different dimensions, with equality occurring when the points are distributed uniformly over the attractor. The usefulness of these dimensions depends on the effectiveness of the computational algorithms used to compute them. In this thesis we will use the correlation dimension because it is computationally one of the most robust.

In this thesis we do not write our own code to implement the Grassberger/ Procaccia algorithm. Instead, we rely on the software package Chaos Data Analyzer (Version 1.0, Physics Academic Software), which calculates the correlation dimension of the underlying attractor according to this algorithm. One needs only to input a time series and specify an embedding dimension and delay parameter.

Before we present the results of the Chaos Data Analyzer (or CDA) on the Lorenz data, one other important point about estimating dimension needs to be addressed. Ruelle (1989) proved that the estimated correlation dimension will always be less than or equal to $2 \log_{10}N$, where N is the number of points in the time series. In other words, only dimension estimates which are significantly below $2 \log_{10}N$ should be regarded as "evidence" that one has discovered an underlying low-dimensional chaotic attractor. Said

another way, if we sought evidence of four-dimensional chaos, we estimate that we would need at least 1000 points in order to estimate the correlation dimension accurately.

We now illustrate the use of CDA on the Lorenz data . Tables 3.1, 3.2, and 3.3 contain the results for N = 500, 1000, 5000, respectively. In all three of these tables, $\tau_s = 0.01$ and $\tau = 5$.

Table 3.1 Correlation Dimension Using 500 points (Lorenz)

Embedding Dimension	Correlation Dimension
1.00	1.03
2.00	1.79
3.00	1.83
4.00	1.96
5.00	2.01
6.00	2.03

Table 3.2 Correlation Dimension Using 1000 points (Lorenz)

Embedding Dimension	Correlation Dimension
1.00	1.03
2.00	1.80
3.00	2.01
4.00	2.02
5.00	2.04
6.00	2.04

Table 3.3 Correlation Dimension Using 5000 points (Lorenz)

Embedding Dimension	Correlation Dimension
1.00	1.03
2.00	1.84
3.00	2.04
4.00	2.05
5.00	2.05
6.00	2.06

In each table, we note that the estimates for correlation dimension do indeed converge once the minimal embedding dimension is reached. Also, we see that the number of data points used affects the speed of convergence. For 500 data points, we obtain a dimension estimate of less than two at embedding dimension three. We know that this estimate is incorrect; Strogatz (1994) states that the Lorenz attractor has correlation dimension of approximately 2.06. Strogatz's estimate is confirmed by our results when we use at least 1000 data points. Further, the limiting dimension estimates are significantly less than $2 \log_{10} N$ for $N = 500$, 1000, and 5000. So based on this technique of estimating the dimension of the underlying attractor, and keeping in mind Ruelle's result, we conclude that our Lorenz data is indeed from a low-dimensional, chaotic source.

Next, to illustrate what will happen when using a pseudo-randomly-generated time series, we use the 1000 point time series we generated in S-PLUS. Table 3.4 shows the result of calculating the correlation dimension of this time series.

Table 3.4 Correlation Dimension Using 1000 Points (Pseudo-Random)

Embedding Dimension	Correlation Dimension
1.00	1.03
2.00	2.05
3.00	2.99
4.00	3.80
5.00	4.54
6.00	5.12
7.00	5.72
8.00	6.27
9.00	6.69
10.00	7.07

In contrast to the Lorenz data, we see that the estimated dimension increases as the embedding dimension is increased, and the estimates do not appear to converge. Keeping in mind that we are actually using a pseudo-randomly generated time series (which most likely is the result of a high-dimensional dynamical system), it may be that the estimated dimensions will eventually taper off at some value. CDA only allows us to increase the embedding dimension to ten, though, so for this time series we are not able to observe whether or not the dimension estimates eventually converge. For a truly "random" time series, no such tapering off will ever occur. True "noise" always seeks to be unfolded in higher and higher dimensions, because the unfolding never takes place.

C. ESTIMATING LYAPUNOV EXPONENTS

Taking our discussion in Chapter II as an introduction to the subject of Lyapunov exponents, we proceed from there and emphasize that an important signature of low-dimensional chaos in a time series is the existence of a positive Lyapunov exponent. Given a time series {s(n)}, how do we go about estimating the largest Lyapunov exponent

associated with the underlying dynamical system? As stated in Chapter II, if we knew the underlying system of equations which generated the time series, one might be able to use straight-forward, analytical methods to estimate the complete spectrum of Lyapunov exponents (Strogatz, 1996). However, not knowing the underlying system necessitates a computational method for estimation.

Many authors have addressed this topic in the literature, and the most commonly used method/algorithm is the one presented by Wolf et. al. (1985). Their technique for estimating the non-negative Lyapunov exponents from a finite amount of experimental data involves examining the divergence of orbits in the space of the reconstructed attractor. In order to estimate the largest Lyapunov exponent, called λ_1, we monitor the long-term evolution of a single pair of nearby initial conditions. Note that although the attractor was actually reconstructed from a single trajectory, it can provide pairs of points that may be considered to lie on different trajectories.

To estimate the largest positive Lyapunov exponent of a time series {s(n)}, we first construct the state space using an appropriate delay parameter τ and embedding dimension d_E. After the state vectors $\mathbf{y}(t)$ have been created, we locate the nearest neighbor (using Euclidean distance) to the initial point $\mathbf{y}(t_0)$ and denote the distance between these two points as $L(t_0)$. At a later time t_1, the initial length will have evolved to length $L'(t_1)$. The length element is propagated through the attractor for a time short enough so that only small scale attractor structure is likely to be examined. If the evolution time is too large, we may see L' shrink as the two trajectories which define it pass through a folding region of the attractor. This would lead to an underestimation of

λ_1. We then look for a new data point that satisfies two criteria reasonably well: its separation $L(t_1)$ from the evolved original point is small, and the angular separation between the evolved and replacement elements is small. If an adequate replacement point cannot be found, we retain the points first used. This procedure is repeated until the original trajectory has traversed the entire data file, at which point we estimate the largest positive Lyapunov exponent λ_1 as

$$\lambda_1 = \frac{1}{t_M - t_0} \sum_{k=1}^{M} \log_2 \frac{L'(t_k)}{L(t_{k-1})} ,$$

where M is the total number of replacement steps. The time step $(t_{k+1} - t_k)$ is held constant. This algorithm is adversely affected by noisy data, too few data points, and inappropriate choices for d and the delay parameter τ.

In this thesis, we again use the Chaos Data Analyzer to implement the algorithm, noting that it calculates the largest Lyapunov exponent in bits of information per data sample. Tables 3.5, 3.6, and 3.7 show the result of using CDA on the Lorenz time series with N = 500, 1000, 5000, respectively, and with $\tau = 5$.

Table 3.5 Lyapunov Exponent Using 500 Points (Lorenz)

Embedding Dimension	Largest Lyapunov Exponent
1.00	1.23
2.00	0.14
3.00	0.12
4.00	0.11
5.00	0.09
6.00	0.09

Table 3.6 Lyapunov Exponent Using 1000 Points (Lorenz)

Embedding Dimension	Largest Lyapunov Exponent
1.00	1.18
2.00	0.14
3.00	0.10
4.00	0.10
5.00	0.09
6.00	0.09

Table 3.7 Lyapunov Exponent Using 5000 Points (Lorenz)

Embedding Dimension	Largest Lyapunov Exponent
1.00	1.18
2.00	0.09
3.00	0.07
4.00	0.07
5.00	0.07
6.00	0.07

From these results, we would conclude that the largest Lyapunov exponent is approximately 0.07, and we would conclude that the time series is chaotic, since this estimate is positive. As with estimating the correlation dimension, we see that a greater number of data points allows for faster convergence to the "true" estimate, once the minimal embedding dimension is reached. This estimate of λ_1 corresponds to known results for the largest Lyapunov exponent of the Lorenz system. We also note that the estimation of invariants, such as Lyapunov exponents, gives another way to determine the correct minimal embedding dimension. It will be the embedding dimension at which we have convergence of estimates, provided we are using enough data points.

For the pseudo-randomly generated time series, Table 3.8 shows the result of using CDA to calculate the largest Lyapunov exponent of the S-PLUS pseudo-random

data. The most interesting aspect of this table is that there does not seem to be an embedding dimension at which convergence occurs, as with the Lorenz time series. Also, with a truly random time series, we would expect to see an estimated Lyapunov exponent of zero, indicating that on the *average*, distance between initially nearby orbits neither grows nor decays. What should actually happen is that the orbit separation distance should sometimes grow and sometimes decay, at random. The CDA estimates for the largest Lyapunov exponent do get close to zero as the embedding dimension is increased, but perhaps do not actually reach zero because of the "pseudo" random nature of the time series. Possibly, we are again observing the fact that a high-dimensional, chaotic dynamical system was used to generate this time series.

Table 3.8 Lyapunov Exponent Using 1000 Points (Pseudo-Random)

Embedding Dimension	Largest Lyapunov Exponent
1.00	1.39
2.00	0.80
3.00	0.52
4.00	0.39
5.00	0.07
6.00	0.03
7.00	0.02
8.00	0.01
9.00	0.02
10.00	0.03

D. PREDICTOR ERROR

The last criterion we will discuss for determining the existence of low-dimensional chaos is predictor error. Only a very general overview is provided here, since short-term prediction of a time series is the subject of the next chapter. What we intend to show, though, is that short-term prediction will be possible for a chaotic time series and not for a randomly-generated time series (Casdagli et. al., 1992).

First, we need a few definitions. Let $\{s(n)\}$ be the time series, with $n = 1, ...,N$. Let T be the prediction interval, i.e., how many time steps ahead we wish to predict. We call $s_{true}(N + T)$ the true value of the time series at step $N + T$, and we call $s_{pred}(N + T)$ the predicted value. The predictor error measures how far apart these two values are. Predictor error, and how one determines $s_{pred}(N + T)$ will be explained in the next chapter.

For now, though, it is sufficient to note that if we have a chaotic time series, we should expect to see predictor error starting out small for a small prediction interval, and increasing as the prediction interval increases. No such relationship between predictor error and prediction interval will exist for a randomly-generated time series because the predictor error will always be large.

For a chaotic system and for a given short prediction interval, we should also see predictor errors decrease to a value near zero as d is increased to the correct minimal embedding dimension d_E, then increase as d is increased beyond d_E. This will not occur for a randomly-generated time series, because the predictor error will be large no matter

what embedding dimension is used (Casdagli, 1989). These results will be illustrated for the Lorenz time series and for the pseudo-randomly generated time series in Chapter IV.

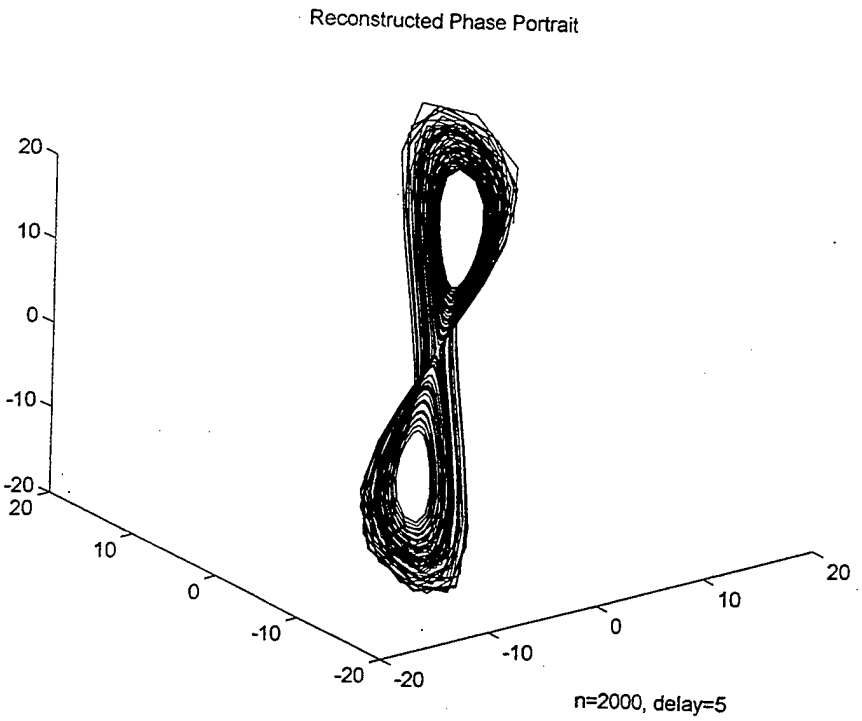

Figure 3.1. Reconstructed Attractor (Lorenz)

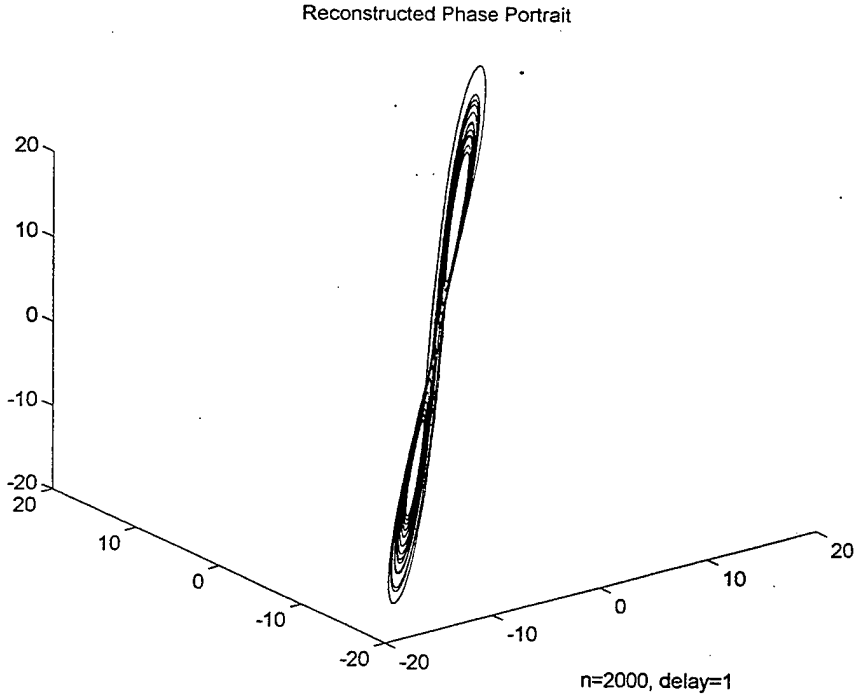

Figure 3.2. Reconstructed Attractor (Lorenz)

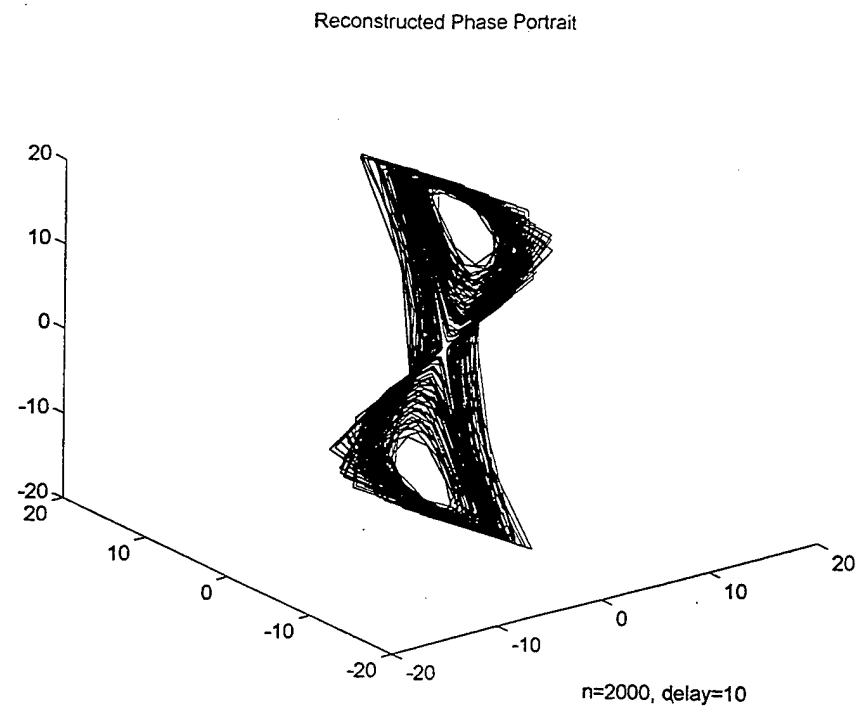

Figure 3.3. Reconstructed Attractor (Lorenz)

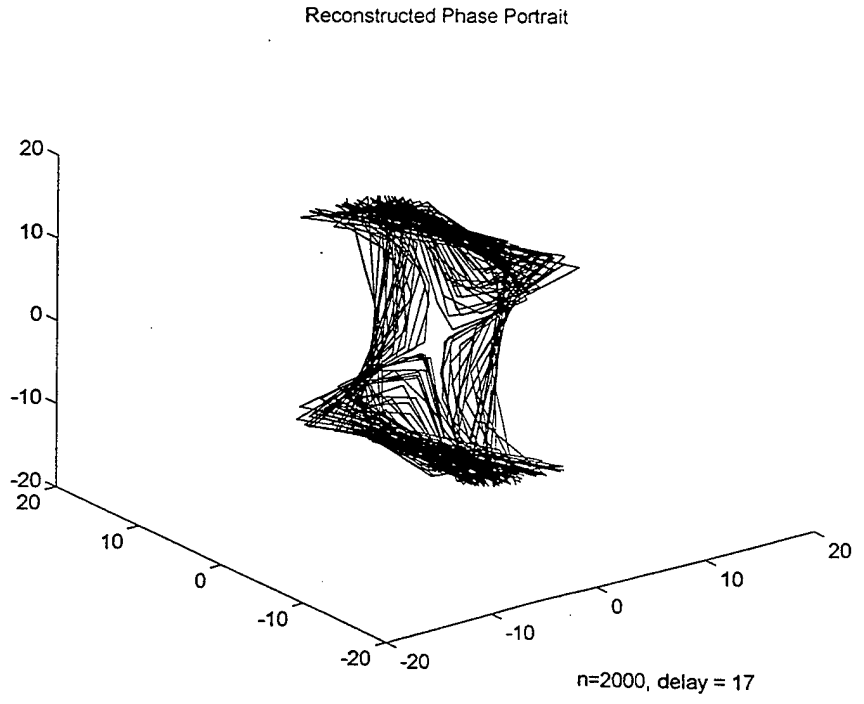

Figure 3.4. Reconstructed Attractor (Lorenz)

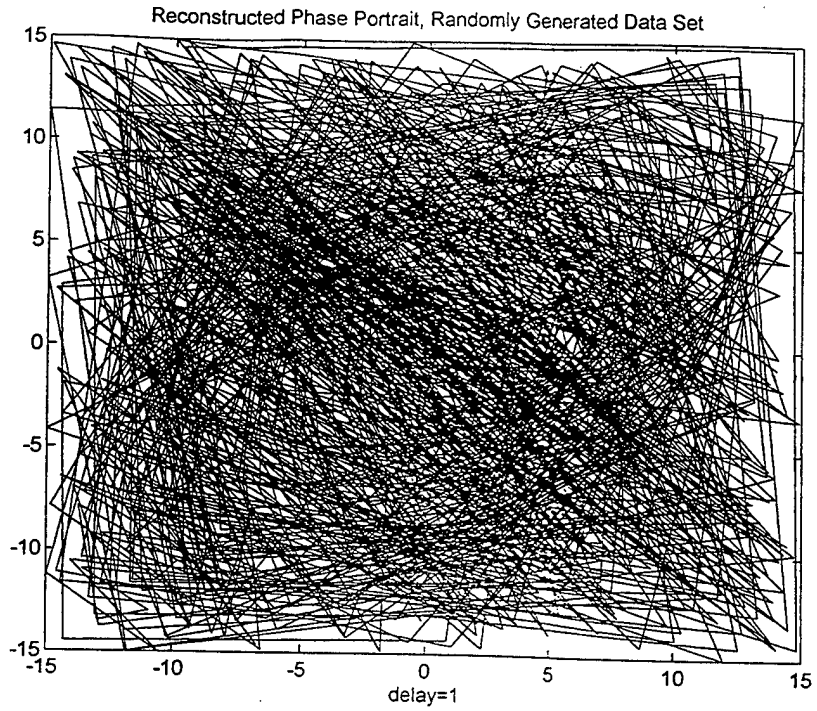

Figure 3.5. Reconstructed Attractor (Pseudo-Random)

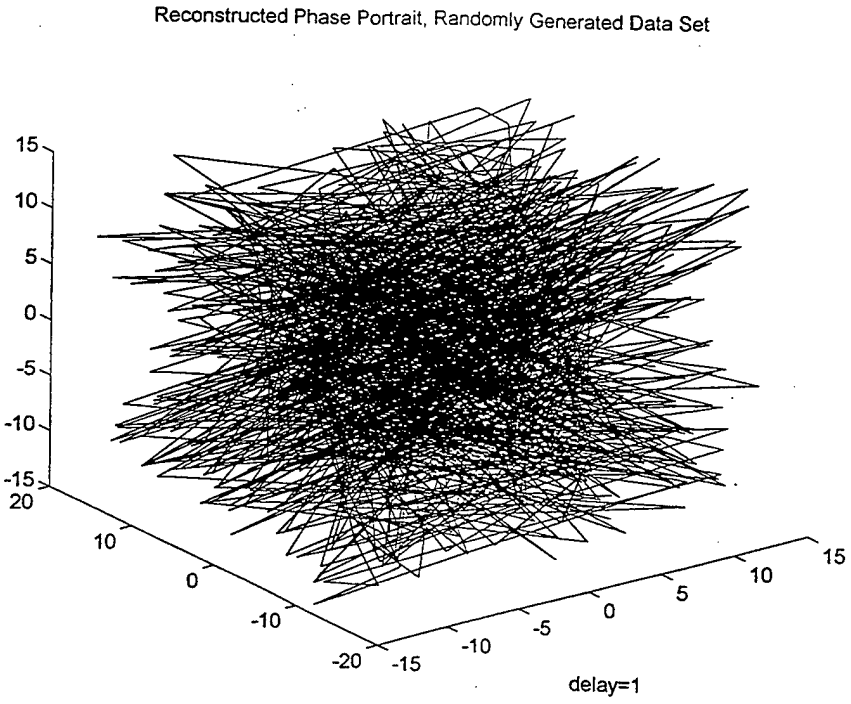

Figure 3.6. Reconstructed Attractor (Pseudo-Random)

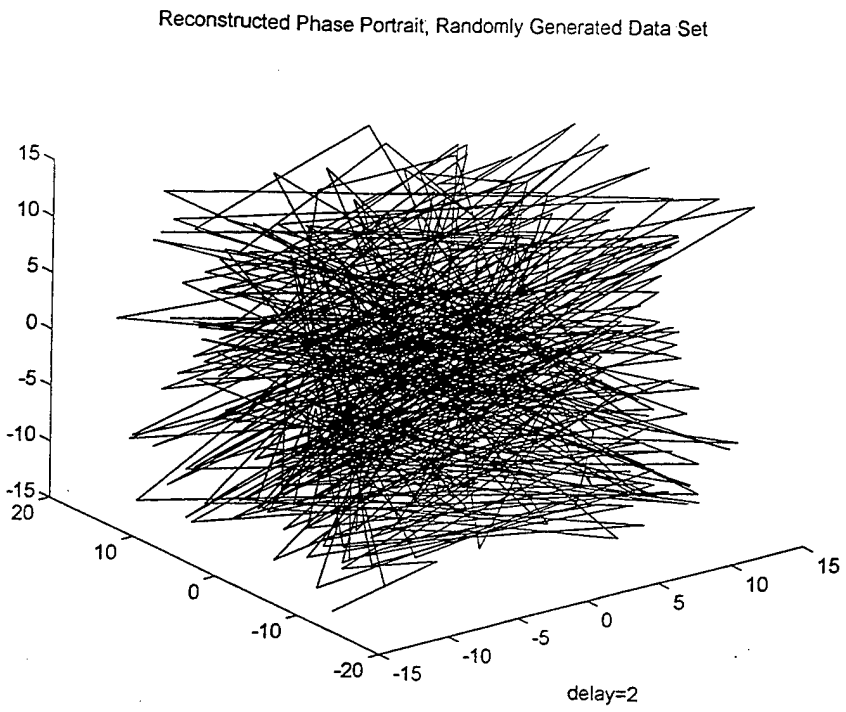

Figure 3.7. Reconstructed Attractor (Pseudo-Random)

IV. SHORT-TERM PREDICTION OF CHAOTIC TIME SERIES

A. GLOBAL METHODS

We have claimed that short term prediction can be accomplished on a chaotic time series $\{s(n)\}$, $n = 1, 2,..., N$; in fact, we mentioned that this is a feature of chaotic time series which distinguishes it from a randomly-generated one. Prediction is called the "inverse problem" in dynamical systems. That is, given a sequence of iterates from a time series, we want to construct a nonlinear map that gives rise to them. Such a map would be a candidate for a predictive model. The consideration of a nonlinear map is essential, since linear maps do not produce chaotic time series.

Several methods exist for predicting time series. The primary references for this section are Casdagli (1989) and Casdagli et. al. (1992). The methods fall into the categories of global function representation and local function approximation. The following is a list of functions that have been employed in global function representation problems.

- *Polynomials:* These have the advantage that their parameters can be determined by a linear least squares algorithm, which is fast and gives a unique solution. However, polynomials have the disadvantage that they blow up as their argument approaches infinity and consequently; they usually do not extrapolate well outside the domain of the data set. Furthermore, they generally behave poorly under iteration.

- *Rational Functions:* These are ratios of polynomials of the same order, and have the advantage that they approach a constant as the argument approaches infinity.

- *Wavelets* are a localized generalization of Fourier series. Their most immediate use is as a means of signal decomposition and hence, state space reconstruction. They present an interesting possibility for function representation within the state space.

- *Neural nets* are currently very popular. They represent the "connectionist" or "brain-style computation" and are frequently used in "pattern recognition" types of problems. Their use in prediction of time series is in the attempted pattern recognition of past iterates of the time series.

- *Radial basis functions* are of the form

$$f(s) = \sum_i a_i \Phi(\|s - s_i\|),$$

where Φ is an arbitrary function, s is the point being predicted from, s_i is the ith value of the time series, and $\|s - s_i\|$ is the distance from s to the ith data point. This functional form has the advantage that the least squares solution of a_i is a linear problem. Radial basis functions are widely used in the fields of computer graphics and vision and have also been used by several authors in time series prediction. (Gershenfeld/Weigend, 1994)

Global function representations have the appeal of representing the entire data set with one form, but they have the disadvantage that it may be difficult or impossible to model the intricate structure of a strange attractor with one global function representation. Local function approximation extends the global function representation methods by weighting the contributions of the points in a data set in proportion to their nearness to the point being predicted from, instead of attempting to take into account the entire data set when making a prediction. Another benefit of local approximation is that there are a priori scaling laws available for the accuracy of approximation.

B. LOCAL FITTING

There are several ways to accomplish local approximation. One is *kernel density estimation*, which is a method for estimating probability density functions from continuous data. The basic idea is to assign to each point an "influence function" or kernel Φ that decays with distance, giving an estimator of the form

$$f(s) = \sum_i \Phi(\|s - s_i\|),$$

where $\|\cdot\|$ denotes a norm. The kernel Φ may be a step function or a monotonically decreasing function, such as an exponential. Kernel density estimation is related to radial basis functions, the difference being that for radial basis functions the contribution of the kernel from each point depends on s and is computed for each s based on least squares, whereas for kernel density estimation, the kernels are ususally fixed or are changed in a uniform manner; for example, for an exponential kernel the decay parameter may be adjusted.

The second way to do local approximation is by *local fitting*. This approach allows one to build a globally nonlinear model while fitting only a few parameters in each local patch. Local fitting is generally quick and accurate, but it has the disadvantage that the approximations are discontinuous. Also, depending on the structure of the underlying attractor, this method may require large amounts of data in order to adequately sample the subregions. Continuity may be achieved by weighted local fitting or by choosing the neighboring points for the local approximation so that they form appropriate simplices. Casdagli (1989) points out that in some cases one may want to select parameters using more robust criteria than least squares. Local function approximation is similar to Tong's

threshold autoregressive models. However, Tong usually constructs two or three different linear models depending on the state of the system, while local function approximation requires building a different model for each state (Tong, 1990).

In this thesis, we demonstrate the method of local fitting, using the ideas contained in Farmer/Sidorowich (1987) and Casdagli et. al. (1992) as our guide. The first step, which we have already discussed and demonstrated, is to embed the time series $\{s(n)\}$, $n = 1, 2,...N$, in a state space with embedding dimension d_E and delay parameter τ chosen as previously discussed. Next, we assume a functional relationship between the current state vector $\mathbf{y}(n)$ and the future state $\mathbf{y}(n+T)$, where T is the number of steps ahead one wants to predict. If this functional relationship is $\mathbf{y}(n+T)=f_T(\mathbf{y}(n))$, then we want to find a predictor F_T which approximates f_T. As mentioned earlier, if the time series is chaotic, f_T is necessarily nonlinear.

To predict $s(n+T)$, we first define a metric $\|\cdot\|$ on the state space, and find the k nearest neighbors of $\mathbf{y}(n)$, i.e. the k states $\mathbf{y}(n')$ with $n' < n$ that minimize $\|\mathbf{y}(n)-\mathbf{y}(n')\|$. We can then construct a local predictor, regarding each neighbor $\mathbf{y}(n')$ as a d_E-dimensional point in the domain and $s(n'+T)$ as the corresponding point in the range. A common approach for constructing this local predictor is to fit a linear polynomial to the pairs $(\mathbf{y}(n'), x(n'+T))$. Note that the range is a scalar, and is not d_E-dimensional like the domain. For some purposes, it may be desirable to let the range be d_E-dimensional as well. When $k = d +1$ this scheme is equivalent to linear interpolation, but Farmer/Sidorowich (1987) propose that a choice of $k > d +1$ ensures greater stability of the solution. Of course, one may want to fit a higher-order polynomial for the local map, but in this thesis we shall only

demonstrate the use of the linear map. Figure 4.1, taken from Sidorowich (1987, p. 122), graphically depicts this idea. In the figure, the current state **y**(t) and the unknown future state **y**(t+T) are represented by open circles and the black dots inside the dashed circle are the neighbors of **y**(t). To make a prediction, a local chart is fit with the neighbors in its domain and the states they evolve into a time T later in its range.

We will also need some measure of how accurate our predictions are. We use the normalized mean-square-error, which will be further denoted as E. In order to test our predictions we will need to treat part of the data we have as the training data set (the part we assume we know) and the other part as the test data set, against which we plan to test our predictions. The normalized error E is given by

$$E = \frac{\sum_{j=1}^{m} (s_{true}(j) - s_{pred}(j))^2}{\sum_{j=1}^{m} (s_{true}(j) - s_{mean})^2},$$

where $s_{true}(j)$ is the true value of the time series, $s_{pred}(j)$ is the computed predicted value, s_{mean} is the average of the true values over the range of predictions, and m is the number of data points in the test data set. This formula assumes the calculation of predictions which are compared to the (assumed known) time series values. Predictions are perfect if E = 0, while E = 1 indicates that the accuracy of predictions is no better than a constant predictor equal to the average of previous time series values. A value for E which is greater than one indicates even worse performance then the average of previous time series values.

As mentioned earlier, an advantage of local approximation is the existence of a priori scaling laws. Provided that the correct embedding dimension has been chosen and that the time series is relatively noise-free, Farmer/Sidorowich derive the following as an error estimate:

$$E \approx C\, e^{(m+1)\,\lambda_{max}\,T} N^{-(m+1)/D} \qquad (4.1)$$

where m is the order of the approximating polynomial used, C is a constant, λ_{max} is the largest Lyapunov exponent when m = 0, and equals the metric entropy otherwise. The metric entropy is equal to the sum of the positive Lyapunov exponents. From equation (4.1) we are able to observe that the error decreases as N increases, and increases when λ_{max}, m, D, and T increase. Farmer/Sidorowich (1987) claim that this prediction scheme can be quite effective on low- to moderate-dimensionality time series. We note that other interesting ideas about the performance of a predictor exist in the literature. For example, May/Sugihara (1990) compute the traditional linear correlation coefficient of observed and predicted values. A correlation coefficient of one indicates perfect predictions while a value close to zero indicates no correlation between observed and predicted values. Kuguimtzis et. al. (1994) propose a predictor which, instead of minimizing E, would reproduce dynamic invariants such as Lyapunov exponents and attractor dimension in the original data.

Another important decision when making predictions for more than one time step ahead is whether they should be constructed directly or iteratively. A direct prediction would be accomplished by making one long term prediction over the full interval T. In contrast, an iterative prediction would be accomplished by splitting the long interval T into

smaller increments, and then concatenating short term forecasts to produce a long term prediction. Several authors argue that iterative predictors are generally more stable and accurate than direct predictors. In fact, Farmer proposes that with clean data the error E scales like

$$E \approx Ce^{kT}N^{-(m+1)/D}.$$

The extra factor of e^{m+1} has disappeared, indicating that significant improvement is possible by increasing the degree of the approximating polynomial (Sidorowich, 1992).

The MATLAB code we wrote to accomplish the local fitting is given in Appendix C. We include a code which accomplishes a one-step prediction and a code which accomplishes an iterative, multi-step prediction. Both of these codes assume an embedding dimension of three, but they are (and were) easily changed to produce results using different embedding dimensions. Both codes allow different choices of k (number of nearest neighbors), N (number of data points in the training data set), τ (delay), and the multi-step prediction code allows for different choices of T (prediction interval).

For the Lorenz time series, we started by doing the one-step predictions while varying d, τ, k, and N. We let d take on the values 2, 3, 4, 5; τ take on the values 1, 2,..., 10; k take on the values 2, 3,...,10; and N take on the values 500, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000. For each of the 3600 different possible combinations, we calculated the absolute error $|s_{true}(N+1) - s_{pred}(N+1)|$. The resulting tables are too numerous to present here; instead we give a brief synopsis of what we discovered.

Of the embedding dimensions used, the value of three corresponded to the most accurate predictions. While embedding dimensions of four and five gave more accurate predictions than did embedding dimension two, they gave less accurate predictions than did embedding dimension three. This seemed to make sense to us, given that we had already determined that the appropriate minimal embedding dimension d_E was three. This also agreed with Abarbanel's claim that an embedding dimension used for predictive purposes should be high enough to unfold the attractor, but no higher (Abarbanel, 1996).

Of the delay parameters τ used, the value of one seemed to result in the most accurate predicions. Predictions got steadily worse as the delay was increased. We were at first surprised by this result, as we were expecting a delay of five to correspond to the most accurate predicitions, since that was the optimal delay we found using the technique of average mutual information. However, we realized that using smaller delays would result in smaller absolute errors, but would not necessarily result in smaller relative errors (relative to the delay, or time step) being used. We later saw the result we were expecting when computing iterative predictions.

As for number of data points used in the training set, we saw improved predictions when the number of data points was increased. This makes sense, since we are, in effect, better populating the attractor when we use more data points. However, there was no obvious best choice for the number of nearest neighbors to use, except that two was too few and seven or more was too many. Farmer/Sidorowich propose using $k > d + 1$, for stability. We decided to use five nearest neighbors when computing the iterative predictions.

Next, we attempted predictions 200 steps ahead, iteratively, using 1000 data points in the training data set, five nearest neighbors, and an embedding dimension of three. Again we let the delay parameter take on the values 1, 2,..., 10. For each choice of τ, we calculated the prediction step at which the normalized error E reached the value of one. We call this value of the prediction step the prediction horizon. Table 4.1 depicts the result of these calculations. A delay parameter of five corresponds to the largest prediction horizon, a result which agrees with our previous determination of the delay parameter.

Table 4.1 Iterative (100 Step) Predictions With Lorenz Data

Delay Parameter	Prediction Horizon
1	30
2	49
3	54
4	84
5	110
6	90
7	75
8	50
9	41
10	35

To further illustrate the iterative prediction, Figure 4.2 depicts the actual time series values vs. the predictions for a delay of five. As can be seen by the graph, predictions break down after about 100 steps. But, more importantly, we see that short-term predictions are, indeed, possible for a low-dimensional chaotic time series. In contrast, we display in Figure 4.3 the result of attempting to predict the S-PLUS pseudo-random time series, using an embedding dimension of three. Similar graphs were

obtained for increased embedding dimensions. The prediction horizon was either one or two for all combinations of d and τ used.

One issue not yet addressed is how predictions, attractor reconstruction, and invariant estimation changes as a result of noise in the time series. To illustrate how prediction can fail on a noisy time series, we took the Lorenz time series and added Uniform (-1,1) noise to it. We can see the effect of the noise in the phase portrait in Figure 4.4. This noise will affect the accuracy of invariant estimations and predictions. The result of attempting to predict this noisy time series is displayed in Figure 4.5. With the noise added, we are now not able to predict accurately for even a few steps.

Tables 4.2 and 4.3 contain the results of estimating correlation dimension and largest Lyapunov exponent, respectively. We seem to have identified a positive Lyapunov exponent, but it is much closer to zero and our estimates do not converge. Also, the estimation of correlation dimension is significantly inaccurate.

Table 4.2 Correlation Dimension using using 5000 points (Noisy Lorenz)

Embedding Dimension	Correlation Dimension
1	1.03
2	2.04
3	2.87
4	3.58
5	4.25
6	4.67
7	5.16
8	5.11
9	5.43
10	5.71

Table 4.3 Lyapunov Exponent using using 5000 points (Noisy Lorenz)

Embedding Dimension	Largest Lyapunov Exponent
1	1.73
2	0.09
3	0.07
4	0.04
5	0.04
6	0.02
7	0.01
8	0.03
9	0.02
10	0.03

The issue of noise has only recently been treated in the literature, and there is considerable room for further development. Farmer and Sidorowich have done much of the work in this area. They focus on methods such as nonlinear smoothing to separate the "noise" from the "signal" in deterministic chaos.

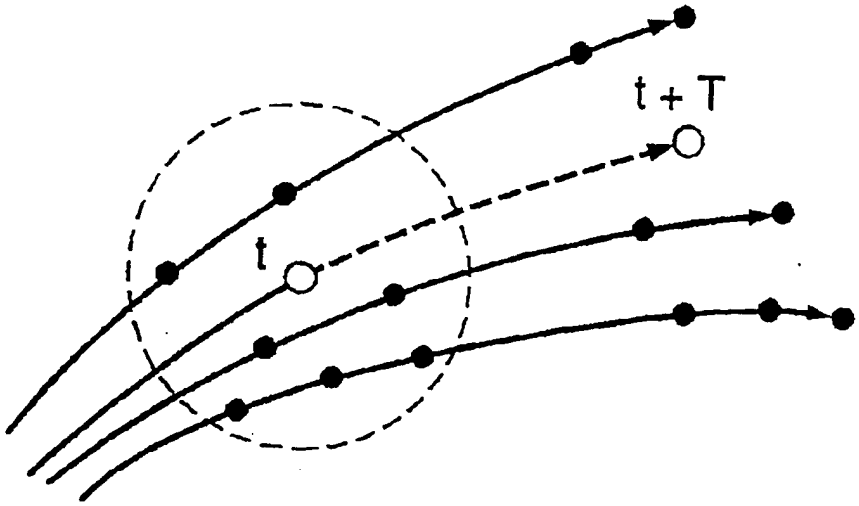

Figure 4.1. Theory of Local Fitting

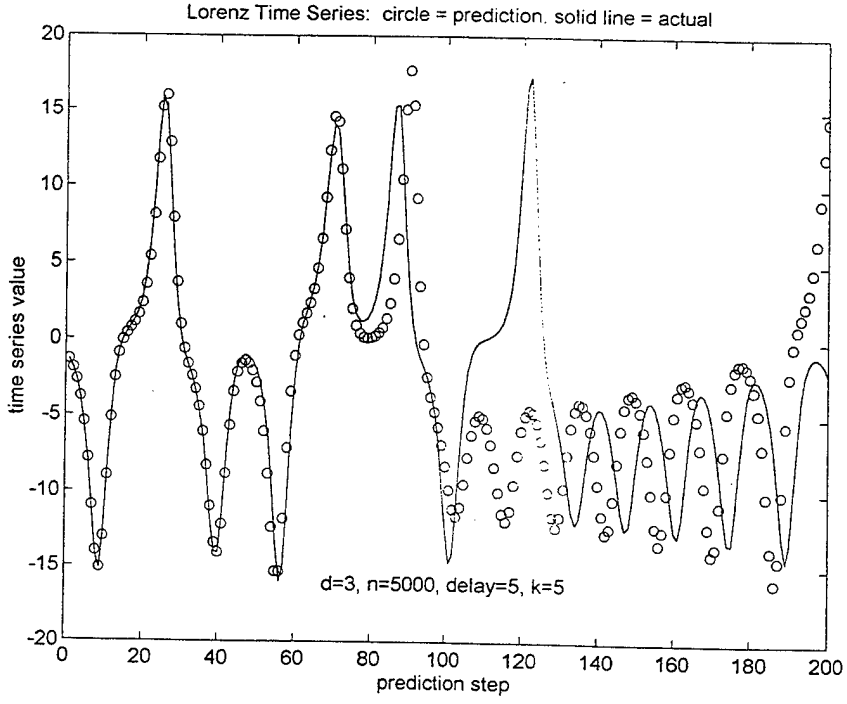

Figure 4.2. Local Fitting Predictions (Lorenz)

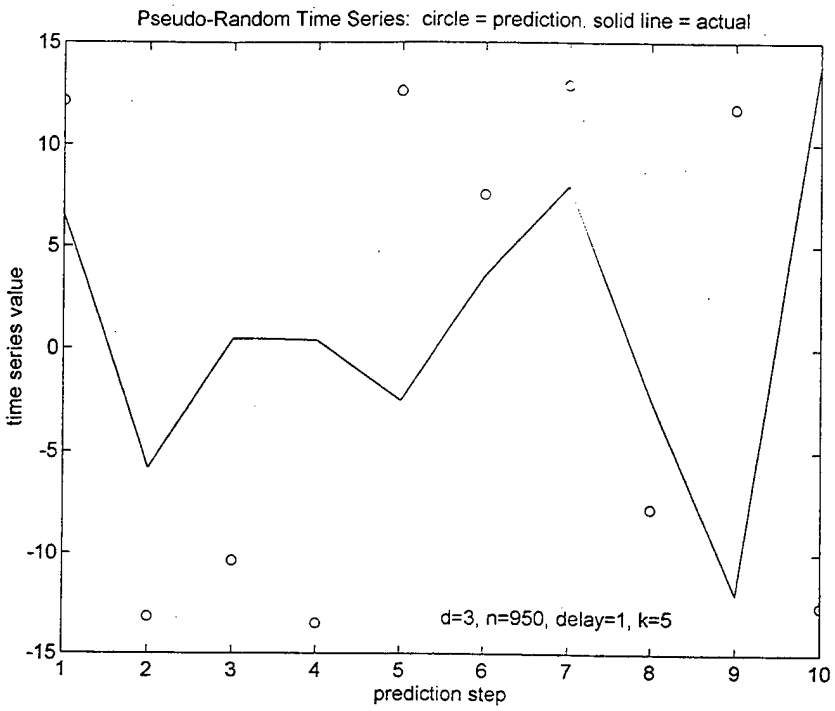

Figure 4.3. Local Fitting Predictions (Pseudo-Random)

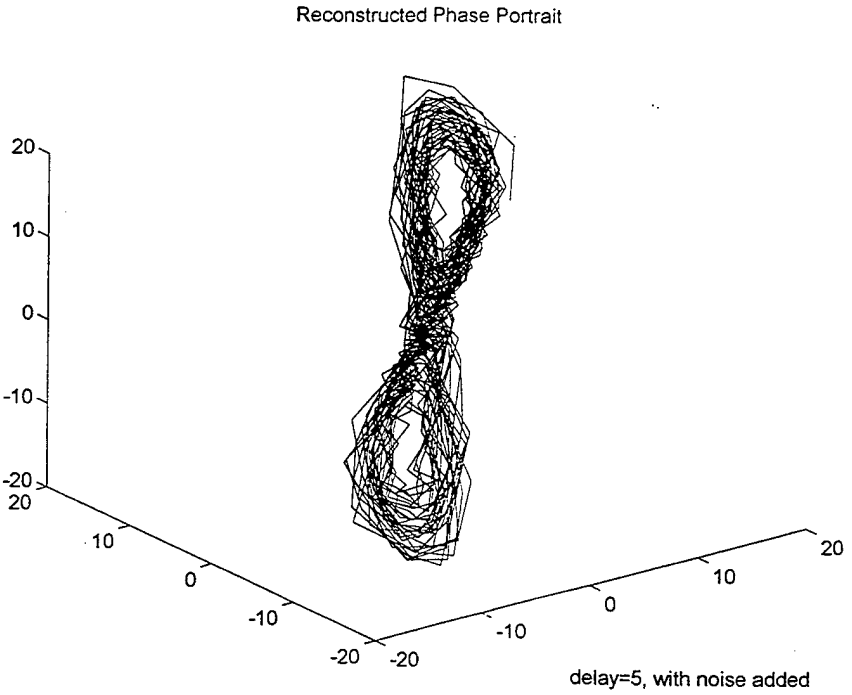

Figure 4.4. Reconstructed Attractor (Lorenz With Noise)

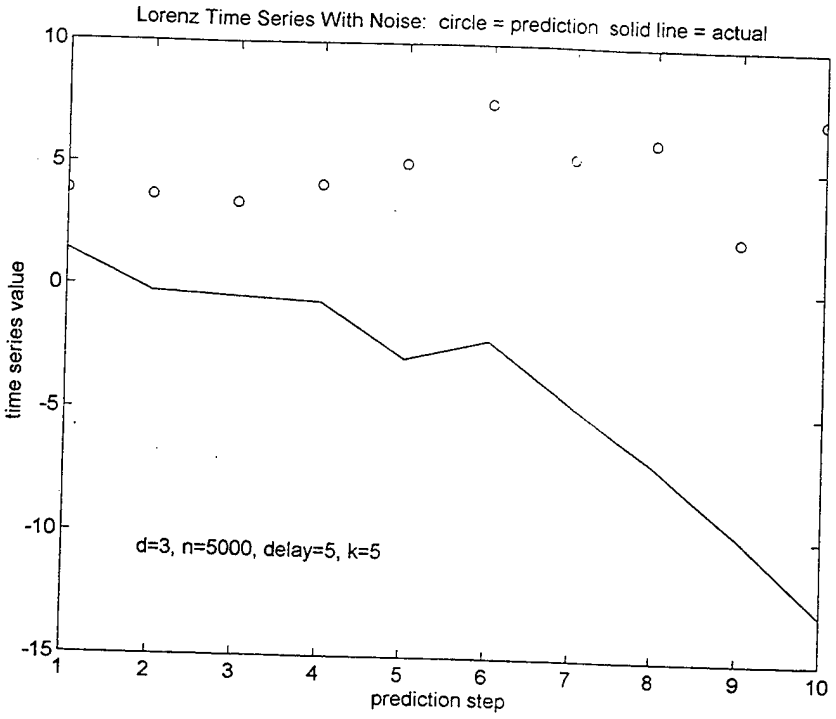

Figure 4.5. Local Fitting Predictions (Lorenz With Noise)

V. DEMONSTRATION OF TECHNIQUES ON DATA SETS

A. FAR-INFRARED LASER DATA

Having explained the theory behind state space reconstruction, techniques for detecting low-dimensional chaos, and prediction, we now intend to demonstrate these methods on some time series to illustrate some of their strengths and weaknesses.

The first time series that we consider is discussed in Gershenfeld/Weigend (1994), and was used for a time series competition held by the Santa Fe Institute (SFI) in 1992. The time series, referred to in Gershenfeld/Weigend (1994) as Data Set A, arises from a physics laboratory experiment which provided very clean data: 1000 points measuring fluctuations in a far-infrared laser which are approximately described by three coupled nonlinear ordinary differential equations. Attributes of this data set, as prescribed by SFI, are stationary dynamics, low-dimensional, low-noise, and short.

Before reconstructing the state space, we use our S-PLUS code (in Appendix A) to determine the first minimum of the average mutual information function on this time series. The result is shown in Figure 5.1. It indicates that we need to use a delay parameter τ of two. Next, we determine the minimal necessary embedding dimension d_E using the MATLAB code (in Appendix B). Figure 5.2 shows the result of using a threshold of ten and 1000 points; we need an embedding dimension of three. Having the tools now to reconstruct the state space, we attempt to detect the presence of low-dimensional chaos, using the methods presented in Chapter III. First, we reconstruct the phase portrait. The result of using τ and d_E as chosen above is shown in Figures 5.3,

5.4, 5.5, and 5.6. These figures are the result of using delay parameters equal to one, two, three, and four, respectively. We note that, judging by the phase portraits, we would decide to use a delay of one, since it corresponds to the most "structure" in phase space. This does not agree with the first minimum of the average mutual information function, which is two. We disregard this discrepancy for now, noting that perhaps the "breaks" we chose for the histogram may have something to do with this discrepancy. A picture is worth a thousand computations, so we decide on using $\tau = 1$. Increasing τ only distorts the phase space portrait.

We next calculate the correlation dimension of the underlying attractor, using Chaos Data Analyzer. Table 5.1 shows the result. We see that the correlation dimension estimates converge (to one decimal place) to 2.1. We suspect that we may need more than 1000 data points in order to get convergence to two decimal places.

Table 5.1 Correlation Dimension Using 1000 Points (Data Set A)

Embedding Dimension	Correlation Dimension
1	3.88
2	1.82
3	2.05
4	2.14
5	2.09
6	2.12

Next, we estimate the largest positive Lyapunov exponent using CDA. The results are contained in Table 5.2. We see that Lyapunov exponent estimates converge to 0.07. So far, it appears that we have detected low-dimensional chaos in this time series.

Table 5.2 Lyapunov Exponent Using 1000 Points (Data Set A)

Embedding Dimension	Largest Lyapunov Exponent
1	0.95
2	0.13
3	0.11
4	0.08
5	0.07
6	0.07

We now attempt to predict, using our MATLAB code (in Appendix C), by fitting a local linear map to the k nearest neighbors of the point from which the prediction is made. Figure 5.7 shows the result of using embedding dimension of three, delay parameter of one, and five nearest neighbors. We see that we are able to predict reasonably well until time step 22. This is where the normalized error E approaches one. For comparison, we show in Figure 5.8 the result of using a delay of two, and all other parameters unchanged. Predictions are not too bad, but E approaches one at time step 15. Recalling our earlier conflict of optimal delay of either one or two, we now decide that $\tau = 1$ corresponds to the optimal delay for this time series.

We conclude that we have a time series which has arisen from a low-dimensional, chaotic source. We have found evidence of a low-dimensional underlying attractor, both graphically and computationally. We have detected a positive Lyapunov exponent, and we have been able to accomplish short-term prediction for this time series.

B. VOLUME-PRESERVING CHAOTIC MAP

Our next example is taken from a pre-print of a paper written by Carroll and Pecora (1997). It is an example of a hyperchaotic, volume-preserving map. The term *hyperchaotic* indicates the presence of more than one positive Lyapunov exponent; there

are two in the case of the map we will use. A *volume-preserving* map does not have an attractor, i.e., the chaotic motion of the trajectories may cover a large part of the phase space. A volume-preserving map exhibits expanding in some directions and contracting in other directions at the same rates, so that the volume of a ball of initial conditions remains unchanged through time. This is in direct contrast to the Lorenz system of equations, which is *dissipatave*. This means that the volume of a ball of initial conditions approaches zero as time progresses. In fact, volumes in phase space shrink exponentially fast in the Lorenz system. The map studied by Carroll and Pecora has applications in private and secure communication systems, where two chaotic time series with no attractor is a desired entity.

Recall that chaotic motion involves the "stretching" and "folding" of orbits. The "folding" in the volume-preserving map chaotic map is accomplished by a modulus shift, while the "stretching" is accomplished by a linear transformation, i.e., by multiplying an initial vector by a compatible matrix. Carroll and Pecora give the following as an example of a hyperchaotic, volume preserving map:

$$\{x_{n+1} = -(4/3)x_n + z_n\} mod\, 2 - 4, \quad (5.1)$$
$$\{y_{n+1} = (1/3)y_n + z_n\} mod\, 2 - 4, \quad (5.2)$$
$$\{z_{n+1} = x_n + y_n\} mod\, 2 - 4, \quad (5.3)$$

where $\{\}\ mod\ 2$ - 4 means take the result modulus ± 2 and then subtract 4. In their paper, they determine the Lyapunov exponents analytically. They are 0.683, 0.300, and -0.986. They note that the exponents do not add exactly to zero because of round-off error.

We generated 4,000 values of the x variable of this dynamical system to comprise our time series. Using the technique of average mutual information, we determine that the

optimal value for τ is two. This result is displayed in Figure 5.9. Using global false nearest neighbors, we determine that the minimal embedding dimension d_E is eight. This result is displayed in Figure 5.10. Figures 5.11 through 5.13 show the resulting phase portraits in two dimensions, for delays of one, five, and ten, respectively. Noteworthy is the fact that we saw no evidence of an attractor for any of these delays; however, the phase portrait does not cover the entire cube represented by $-6 < y_1 < -2$, $-6 < y_2 < -2$, $-6 < y_3 < -2$. This seems to indicate that the data is deterministic, although we cannot detect the presence of an attractor.

Next, we calculated correlation dimension using CDA. The results are contained in Table 5.3. The estimates do not converge; consequently, it would be difficult to determine thus far whether chaos or a stochastic process is representing itself in the time series.

Table 5.3 Correlation Dimension (Volume Preserving Map)

Embedding Dimension	Correlation Dimension
1	1.02
2	2.03
3	2.78
4	2.76
5	2.79
6	3.31
7	3.61
8	4.89
9	5.26
10	6.25

We next use CDA to estimate the most positive Lyapunov exponent. This result is located in Table 5.4. The estimate agreed with Carroll/Pecora's estimate for the largest positive Lyapunov exponent. This result is noteworthy, since a positive Lyapunov exponent indicates that a chaotic process is being represented in the time series. Unfortunately, though, CDA only allows us to estimate the largest positive Lyapunov exponent; so we are unable to experimentally confirm the presence of a second positive Lyapunov exponent.

Table 5.4 Lyapunov Exponent (Volume Preserving Map)

Embedding Dimension	Largest Lyapunov Exponent
1	1.49
2	1.29
3	0.87
4	0.82
5	0.75
6	0.73
7	0.71
8	0.69
9	0.68
10	0.69

Finally, we attempt short-term prediction. After analyzing the result of using different values for nearest neighbors used on the one-step predictions, we decide that using four nearest neighbors gives the most accurate results. We then performed a ten-step, iterative prediction. The result is shown is Figure 5.14. The prediction for the first step is nearly perfect, but subsequent predictions noticeably fail around step five, when E reaches the value of one.

So what would we conclude from the information we have gathered from this time series? This map has provided us with a good example of something between low-dimensional chaos (like the Lorenz time series) and a stochastic process (like the pseudo-random time series). The absence of an attractor explains the mostly unenlightening phase space portraits and the lack of convergence of correlation dimension estimates. However, the presence of at least one Lyapunov exponent indicates that the time series arises from a chaotic source. Finally, although prediction was not nearly as successful as it was with the Lorenz time series, we were able to predict a few iterates fairly well; in fact, the first prediction was near perfect. So from all this we would conclude, if we had no knowledge of the system from which this time series came, that we had a time series which arose from a deterministic, chaotic, but perhaps high-dimensional, source. Although we would not conclude that we have detected low-dimensional chaos, we certainly have detected chaos.

C. STAGGERED TIME SERIES

As our last example, we take two known low-dimensional, chaotic time series and stagger them. We take as one time series 3000 points of the familiar Lorenz time series we have been discussing. Recall that this time series had an optimal delay value of five. To obtain the other time series we introduce the Rossler system of equations, which is another example of a low-dimensional, chaotic system. The equations are:

$$\dot{x} = -y - z, \tag{5.4}$$
$$\dot{y} = x + ay, \tag{5.5}$$
$$\dot{z} = b + z(x - c), \tag{5.6}$$

where a, b, and c are parameters. This system has a strange attractor for a = 0.2, b = 0.2,

and c = 5.7 (Strogatz, 1994). This strange attractor, like the Lorenz attractor, has fractal dimension between two and three. We obtained a 3000-point time series from this system by extracting values of the x variable, using $\tau_s = 0.01$. We note that a delay of 30 corresponds to the most accurate reconstruction of the Rossler attractor, seen in Figure 5.15. However, the time series we used for the construction of the staggered time series was the Rossler system variable {x(t)}, with a delay of one.

To obtain the staggered time series, we merely staggered components from each of the above two time series, i.e. took the first entry the of Lorenz series, then took the first entry of the Rossler series, took the second component of the Lorenz series , etc. We can represent this time series as {L1, R1, L2, R2, ...,L3000, R3000}, where the L points represent the Lorenz data and the R points represent the Rossler data.

We pause to address the reason for staggering the two time series, using a delay of one on each time series. In other words, why did we not take the Lorenz data (with delay of five already encorporated) and the Rossler data (with delay of 30 already encorporated) and stagger these two? We wanted to start with the original (untested, with respect to choice of delay) data and test for optimal delay once the staggered time series is created.

We next used average mutual information to determine the optimal delay, and the result was that the optimal delay was two. However, a delay of two corresponds to only using the Rossler points R1, R2, etc. We must not accept this recommendation, since it defeats the purpose of staggering the two time series. In fact, the use of any even delay is unacceptable. However, we are pleased that the algorithm for average mutual information apears to have correctly identified the fact that a delay of two corresponded to the most

structure in the time series. A priori, we may have thought that this time series would have stumped the algorithm, but we see that it seems fairly robust. We also used average mutual information to calculate the optimal delay on the time series {R1, L1, R2,...,R3000, L3000}, and the result was still two. This means that, by the nature of the order of the components in the time series, it identified structure in the Lorenz points. We postpone the choice of delay, for now, and calculate the necessary embedding dimension d_E using global false nearest neighbors and using a delay of one. The result was that $d_E = 4$.

The resulting phase portrait of the staggered time series is shown in Figure 5.16, using a delay of one. This plot presents a very interesting sight. One may wonder why we do not see the Rossler attractor and the Lorenz attractor superimposed on the same plot. The reason is as follows. The reconstructed phase portrait of the Lorenz attractor consisted of a plot of the state space "points" [L1, L2, L3], [L2, L3, L4], [L3, L4, L5], etc. The reconstructed phase portrait of the Rossler attractor consisted of a plot of the state space points [R1, R2, R3], [R2, R3, R4], [R3, R4, R5], etc. However, the reconstructed phase portrait of the staggered "attractor" consists of a plot of the state space points [L1, R1, L2], [R1, L2, R2], [L2, R2, L3], etc. Therefore the staggered "attractor" does not consist of the union of the Rossler state space points and the Lorenz state space points. One may also wonder why, then, we see any structure at all, let alone structure as delicate as appears in Figure 5.16. We do not have an answer to this question. Perhaps the topic of staggered time series could be an area for further research.

We proceed by calculating the correlation dimension; the results are in Table 5.5. Although we do not have convergent estimates, the results seem to indicate a fractal dimension between three and four. This confirms the choice of four as the minimal embedding dimension.

Table 5.5 Correlation Dimension Using 6000 points (Staggered)

Embedding Dimension	Correlation Dimension
1	1.02
2	2.05
3	2.51
4	2.72
5	2.92
6	3.12
7	3.25
8	3.31
9	3.38
10	3.45

In Table 5.6 we display the Lyapunov exponent calculation. We seem to have evidence of a largest positive Lyapunov exponent approximately equal to 0.08. It seems then that this structure we discovered exhibits sensitive dependence on initial conditions, with the distance between two initially nearby points growing exponentially with time.

Table 5.6 Lyapunov Exponent Using 6000 points (Staggered)

Embedding Dimension	Largest Lyapunov Exponent
1	1.23
2	0.92
3	0.36
4	0.12
5	0.11
6	0.09
7	0.08
8	0.09
9	0.08
10	0.08

Finally, we attempt prediction using embedding dimension four. Figure 5.17 shows the result of using a delay of one, embedding dimension of four, ten nearest neighbors, 5000 data points, and predicting 15 steps. We determined the number of nearest neighbors to use by varying this parameter from five to fifteen and calculating a one-step prediction, as we did with the Lorenz time series. The most accurate one-step prediction corresponded to using ten nearest neighbors. By step 15, the predictions (at least the ones corresponding to the Rossler points) are significantly inaccurate. An interesting thing to note is that predictions corresponding to the Rossler points fail quicker than do the predictions corresponding to the Lorenz points.

Summarizing our results: we found structure in the phase space portraits; the correlation dimension estimates and global false nearest neighbors suggested a fractal dimension of between three and four; we determined the existence of a positive Lyapunov exponent, and we were able to accomplish short-term prediction. In other words, we have all the signs of a low-dimensional, chaotic time series. However, it would be errant to

conclude that this time series arose from a single dynamical source, i.e. a single set of equations, since we know that we staggered two time series which had no dependence on each other. We believe that this time series, at a minimum, presents an interesting problem possibly deserving of further research.

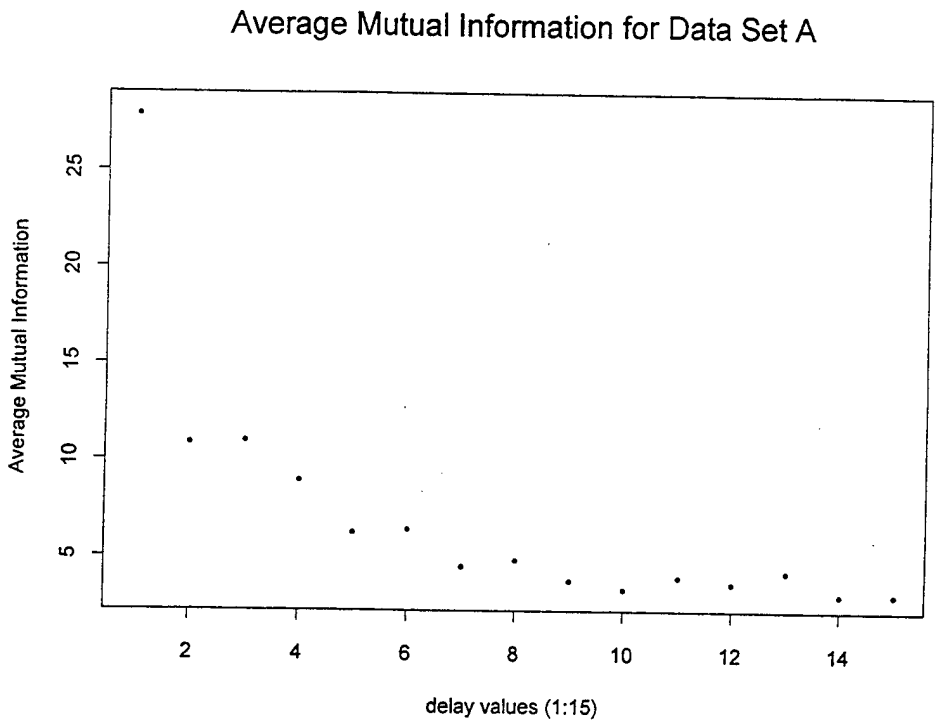

Figure 5.1. Average Mutual Information in S-PLUS (Data Set A)

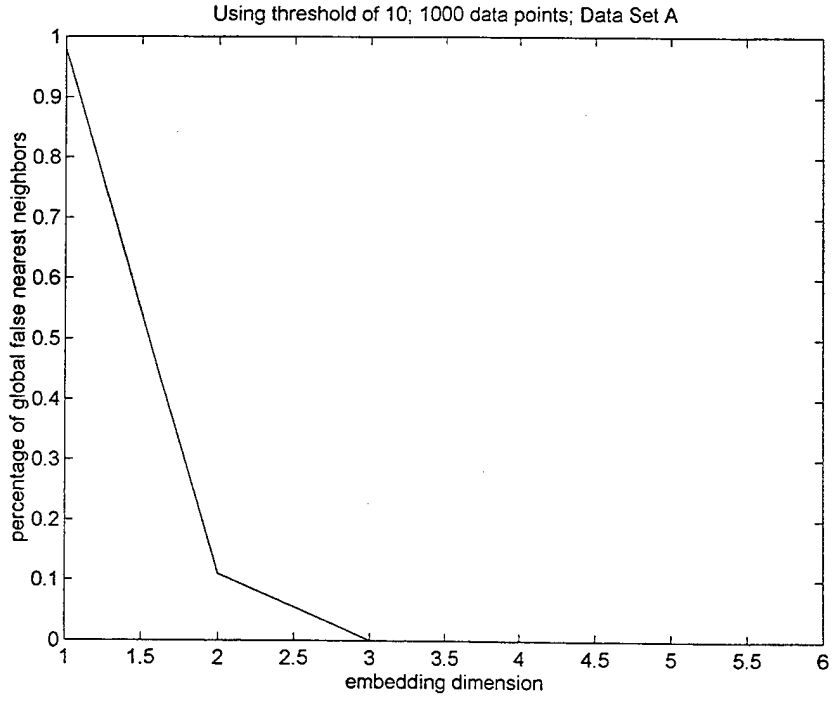

Figure 5.2. Global False Nearest Neighbors in MATLAB (Data Set A)

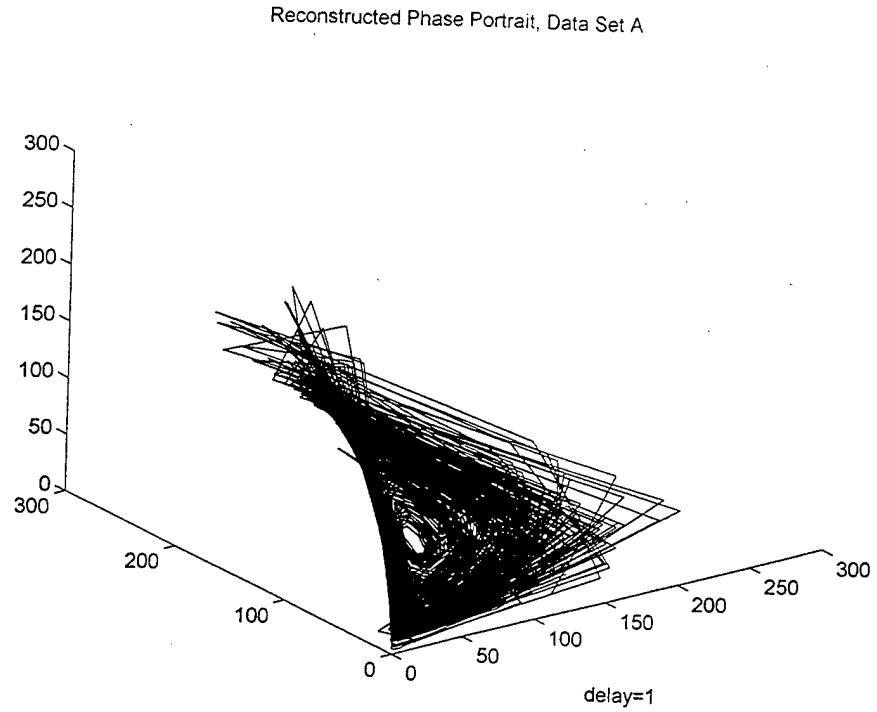

Figure 5.3. Reconstructed Attractor (Data Set A)

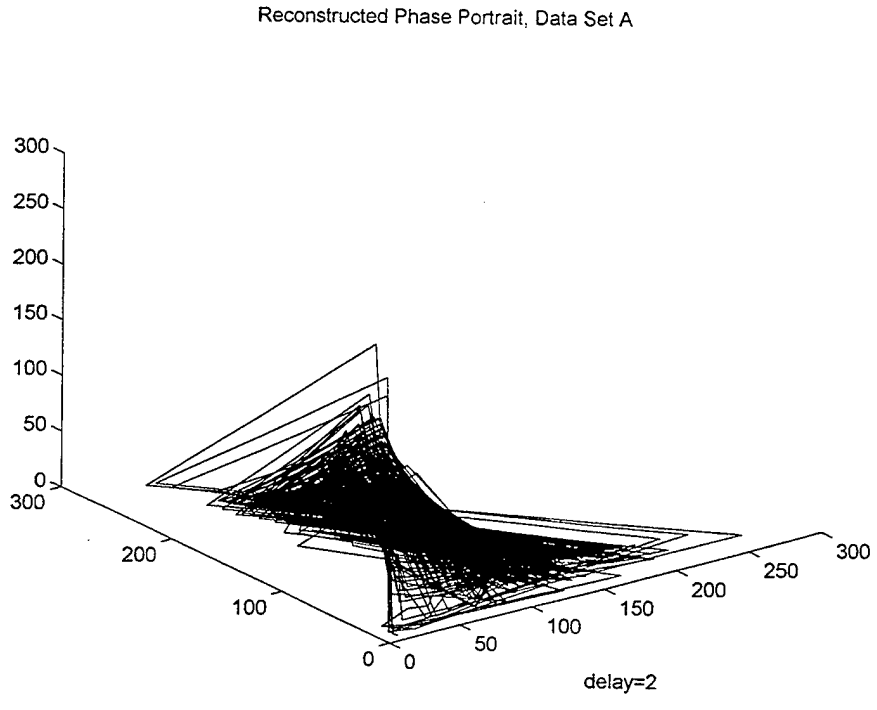

Figure 5.4. Reconstructed Attractor (Data Set A)

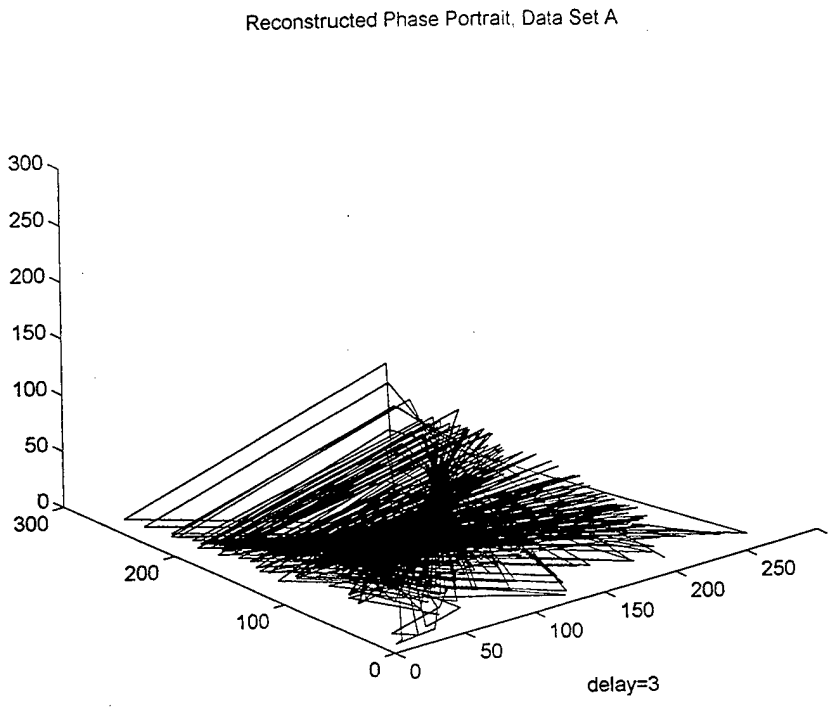

Figure 5.5. Reconstructed Attractor (Data Set A)

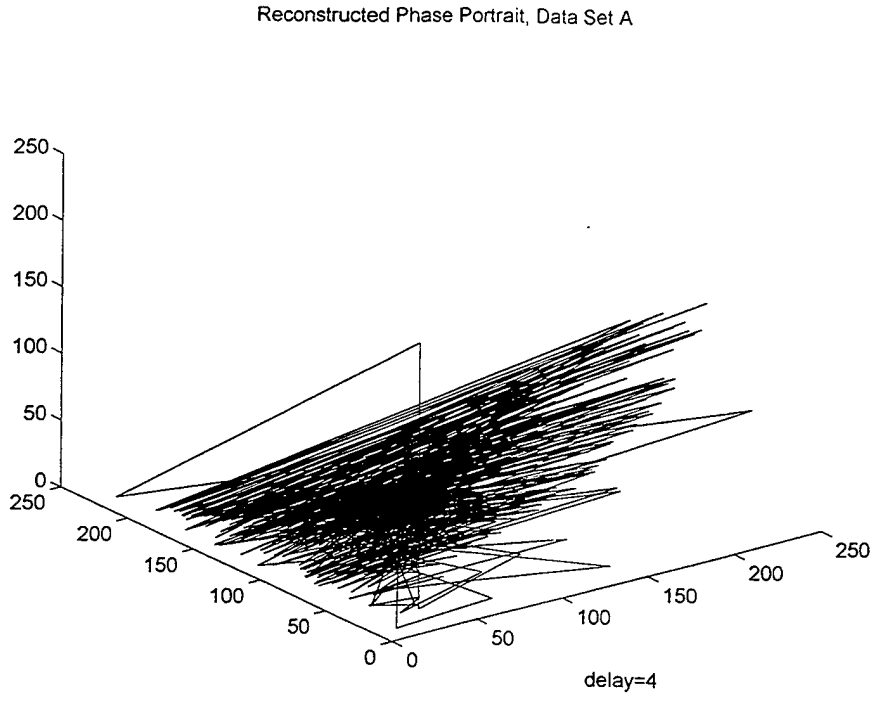

Figure 5.6. Reconstructed Attractor (Data Set A)

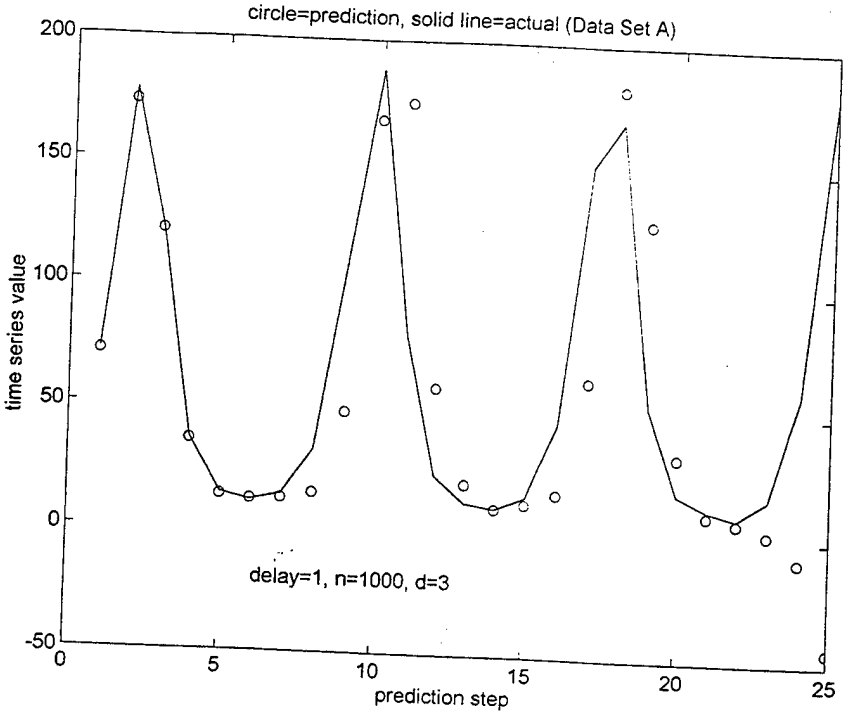

Figure 5.7. Local Fitting Predictions (Data Set A)

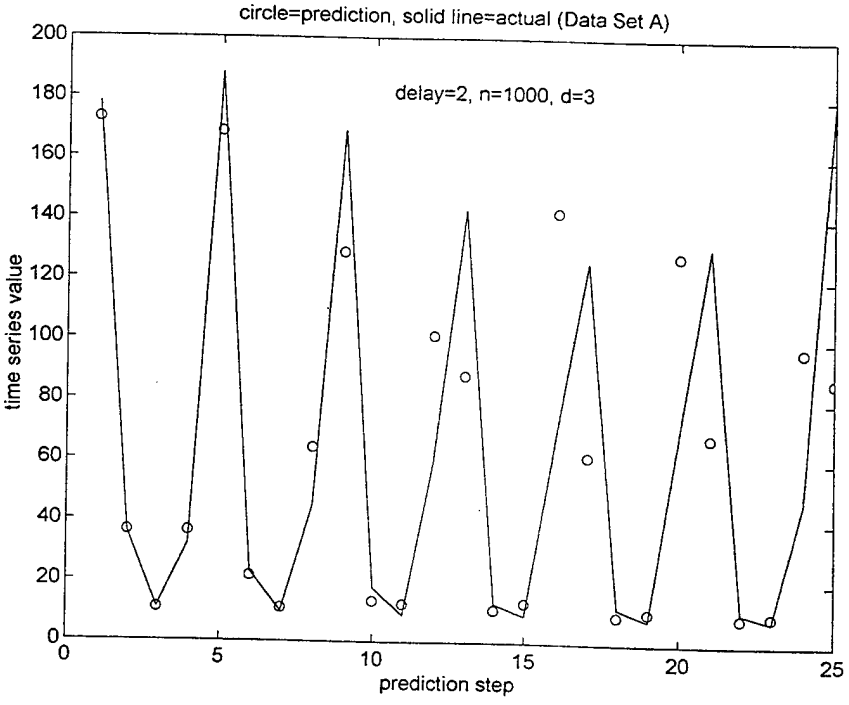

Figure 5.8. Local Fitting Predictions (Data Set A)

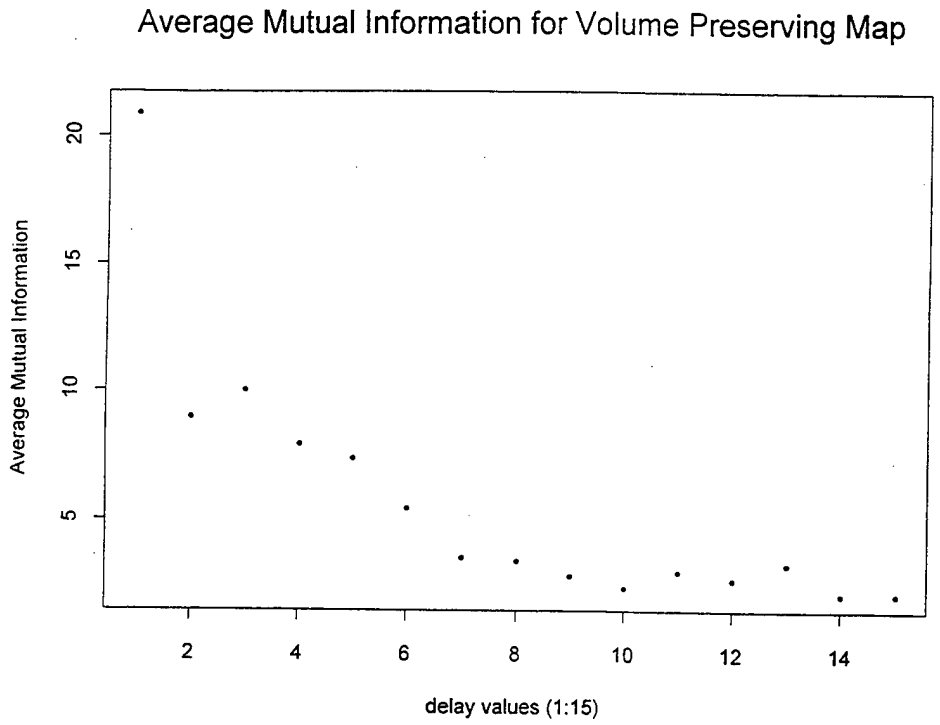

Figure 5.9. Average Mutual Information in S-PLUS (Volume Preserving Map)

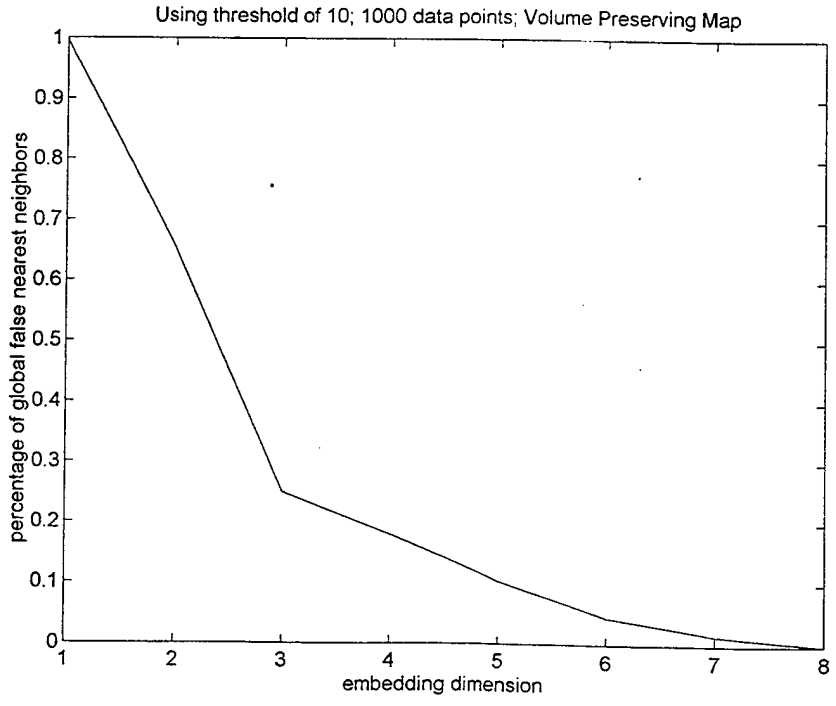

Figure 5.10. Global False Nearest Neighbors in MATLAB (Volume Preserving Map)

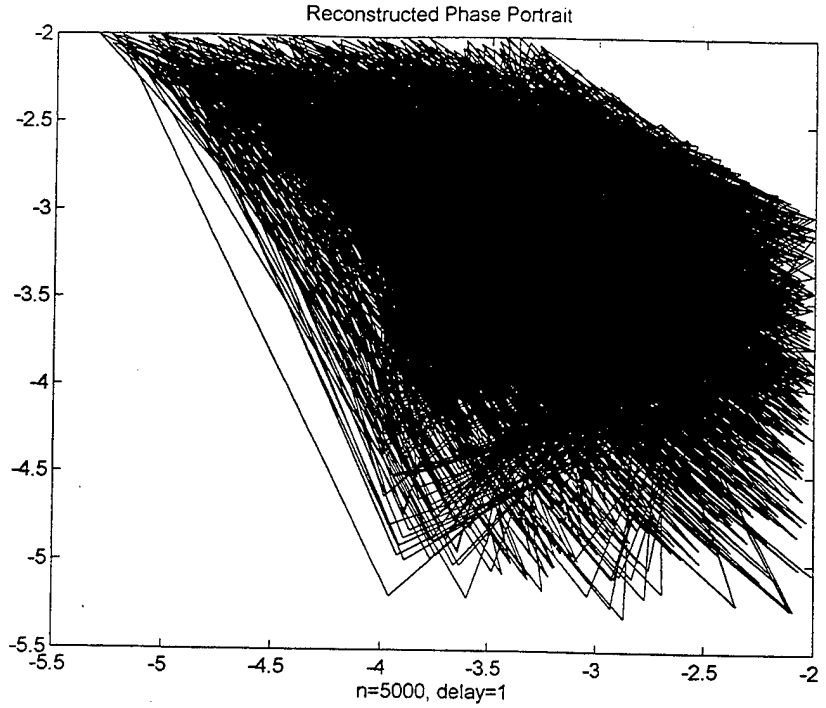

Figure 5.11. Reconstructed Attractor (Volume Preserving Map)

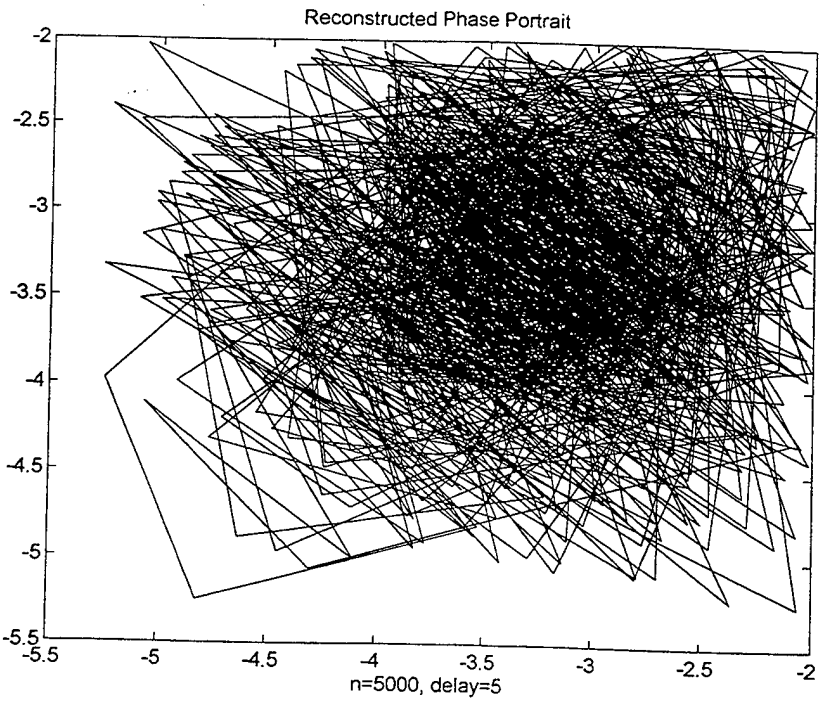

Figure 5.12. Reconstructed Attractor (Volume Preserving Map)

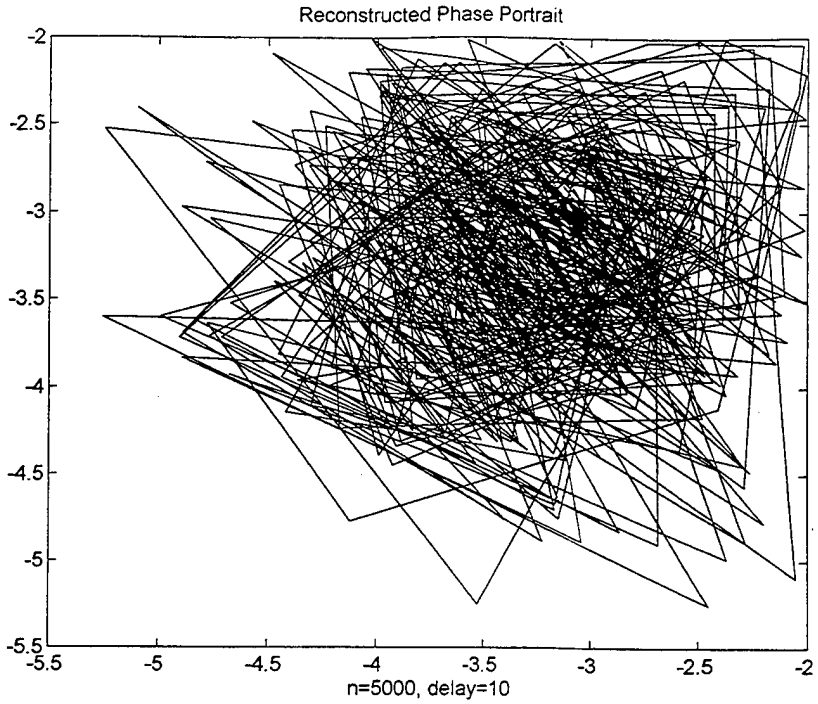

Figure 5.13. Reconstructed Attractor (Volume Preserving Map)

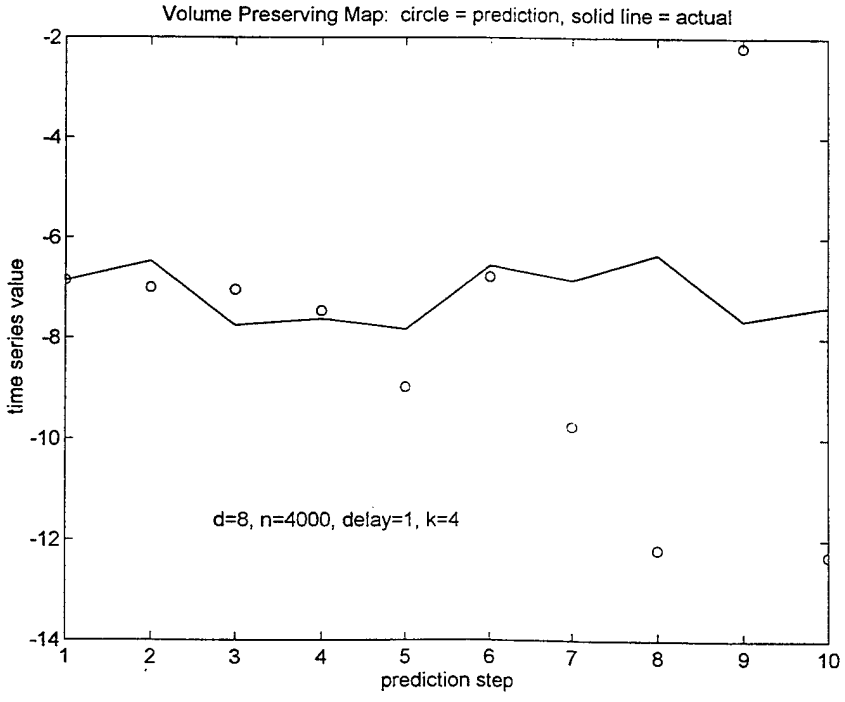

Figure 5.14. Local Fitting Predictions (Volume Preserving Map)

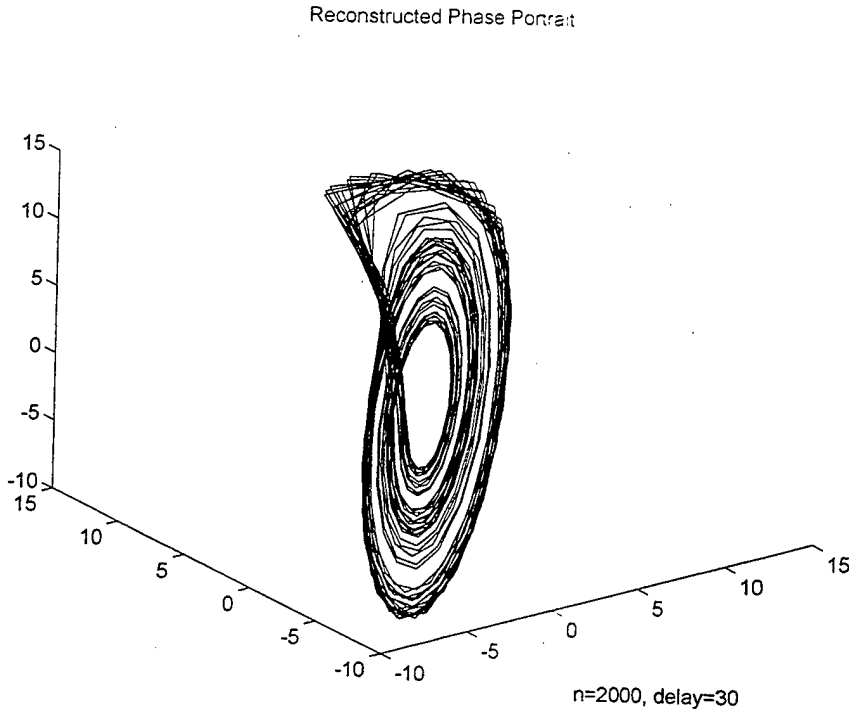

Figure 5.15. Reconstructed Attractor (Rossler)

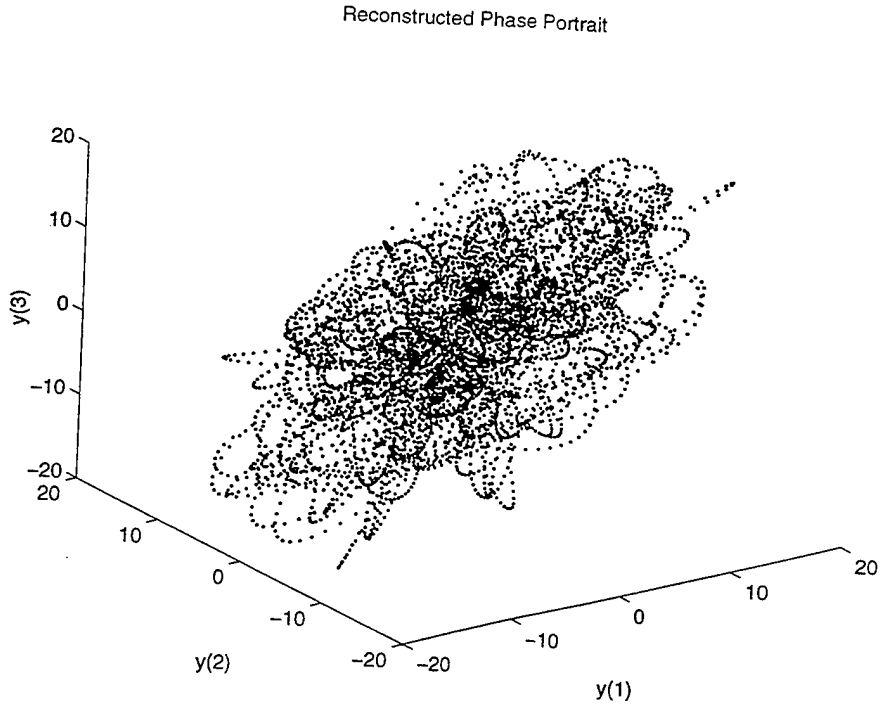

Figure 5.16. Reconstructed Attractor (Staggered)

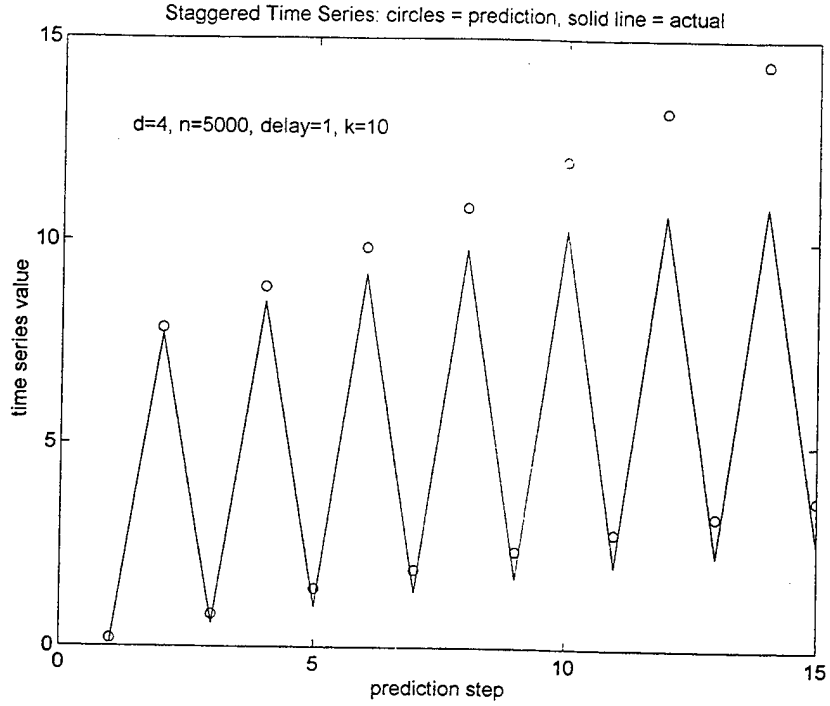

Figure 5.17. Local Fitting Predictions (Staggered)

VI. CONCLUSIONS

In this thesis we have demonstrated the method of state space reconstruction to embed a time series in a state space, and then use that reconstruction to accomplish several tasks. First, one can use the reconstruction to detect low-dimensional chaos in apparently random data. If low-dimensional chaos is detected, we can accomplish short-term prediction using a local linear technique. We discussed issues crucial to the reconstruction, namely, choice of delay parameter and embedding dimension.

We feel that there is still room for research and development in many areas. For instance, there seems to be no prescription or theory in existence yet for choosing the optimal value of nearest neighbors to use. For most of our examples, the value of five seemed to work reasonably well, but we have no real explanation for this. Also, we do not know of an "optimal" way to determine the breaks for the histograms used in the technique of average mutual information.

There is also room for research in the areas of noisy time series and number of data points needed to calculate invariants. Clearly, more noise is bad and more data points is good, but it seems that these ideas might be made more precise than they have been so far in the literature. Finally, the staggered time series brought up many interesting questions such as why do we see structure and what does it mean?

Overall, we conclude that the methods we discussed worked well when applied, i.e., we were mostly able to determine which time series arose from low-dimensional, chaotic processes and which did not. For the chaotic time series, we were able to

accomplish some type of short-term prediction. However, when distinguishing between low-dimensional chaos and noise, we stress that one needs to take into account the results of all methods discussed, not just one or two. Especially for "tricky" cases like the volume preserving map we discussed, one may reach a false conclusion if the results of only one or two methods are considered. In general though, we are satisfied with the application and results of the time series methods we discussed. We feel that these techniques provide a valuable method for analyzing time series, expecially those arising from low-dimensional, chaotic dynamical systems.

APPENDIX A. AVERAGE MUTUAL INFORMATION (S-PLUS)

```
# Read in data file:
series_read.table("a:smx5000.dat")

# Transform data to accomodate changing delay parameter:
q <- t(series)
        n_length(q)
      for (delay in 1:10){
                x_runif(floor(n/delay))
                for(i in 1:floor(n/delay))
                { x[i]_q[delay*i]}

# Initialize:
                n1_length(x)
                R_runif(floor(n/delay))
                S_runif(floor(n/delay))
                Q_runif(floor(n/delay))
                summand_runif(n1-1)
                Muinf_runif(10)

# Apply 1-D and 2-D histograms:
                breaks1_c(-15:15)
                xbreaks1_breaks1
                ybreaks1_breaks1
                len_length(breaks1)
                hm_hist(x,breaks=breaks1 ,plot=F,probability=T)
                cts_hm$counts
                hm2_hist2d(x,x,xbreaks=xbreaks1,ybreaks=ybreaks1,scale=T)
                cts2_hm2$z

# Calculate average mutual information:
                for (j in 1:(n1-1)){
                 for (m in 1:(len-1))
                        {if (x[j] > breaks1[m] & x[j] < breaks1[m+1])
                                R[j]_cts[m]}

                 for (m in 1:(len-1))
                        {if (x[j+1] > breaks1[m] & x[j] < breaks1[m+1])
                                S[j]_cts[m]}

                 for (a in 1:(len-1)) {for (b in 1:(len-1)) {
                        if (x[j] > xbreaks[a] & x[j] < xbreaks[a+1]
```

```
                & x[j+1] > ybreaks[b] & x[j+1] < ybreaks[b+1])
                    Q[j]_cts2[a,b] }}}


            summand[j]_Q[i]*(ln(Q[i]/(R[i]*S[i]))/ln(2))          }

    l=cumsum(summand)
    Muinf[delay]_l[n1-1] }

# Plot average mutual inforamation vs. delay:
win.graph()
plot(delay,Muinf)
```

APPENDIX B. GLOBAL FALSE NEAREST NEIGHBORS (MATLAB)

```
% Specify time series:
x=vect(1:1000);
n=length(x);
%-------------------------------------------------------------------------------------
% embedding dimension = 1
space1=x';
size1=zeros(n,n);
for i=1:n
for j=1:n
 if i==j, size1(i,j)=NaN;
  else if i>j, size1(i,j)=size1(j,i);
 else
   size1(i,j)=norm(space1(i)-space1(j));
 end
end
end
end

nearindex1=zeros(1,n);
neardist1=zeros(1,n);
for i=1:n
  [Y,I]=sort(size1(i,:));
  nearindex1(i)=I(1);  % index of nearest neighbor
  neardist1(i)=Y(1);   % distance to nearest neighbor
end
%-----------------------------------------------------------------
% go to embedding dimension  2
d2 = 2
space2=zeros(2,n-d2+1);
for i=1:n-d2+1
 space2(:,i)=[x(n-i+1),x(n-i)]';
end

size2=zeros(n-d2+1,n-d2+1);
for i=1:n-d2+1
for j=1:n-d2+1
 if i==j, size2(i,j)=NaN;
 else if i>j, size2(i,j)=size2(j,i);
 else
   size2(i,j)=norm(space2(:,i)-space2(:,j));
```

```
 end
 end
 end
 end

 nearindex2=zeros(1,n-d2+1);
 neardist2=zeros(1,n-d2+1);
 for i=1:n-d2+1
  [Y,I]=sort(size2(i,:));
  nearindex2(i)=I(1);  % index of nearest neighbor
  neardist2(i)=Y(1);   % distance to nearest neighbor
 end

% Is d=1 the correct embedding dimension?
 count=zeros(1,n-d2+1);
  j=1;
  for i=2:n
   if nearindex1(i) == 1, count(j)=NaN;
    elseif abs(space2(2,n-d2+3-i)-space2(2,(n-d2+1)-(nearindex1(i)-2))/(neardist1(i))) >
10
   count(j)=1;
   j=j+1;
   end
  end

percent1= (sum(count(find(count>0))))/(length(count)-sum(isnan(count)))
% this is percentage of global false nearest neighbors in dimension 1
%-------------------------------------------------------------------------------
% go to embedding dimension 3
d3=3
space3=zeros(d3,n-d3+1);
for i=1:n-d3+1
 space3(:,i)=[x(n-i+1),x(n-i),x(n-i-1)]';
end

size3=zeros(n-d3+1,n-d3+1);
for i=1:n-d3+1
for j=1:n-d3+1
 if i==j, size3(i,j)=NaN;
 else if i>j, size3(i,j)=size3(j,i);
 else
  size3(i,j)=norm(space3(:,i)-space3(:,j));
 end
end
end
```

```
 end

 nearindex3=zeros(1,n-d3+1);
 neardist3=zeros(1,n-d3+1);
 for i=1:n-d3+1
  [Y,I]=sort(size3(i,:));
  nearindex3(i)=I(1);  % index of nearest neighbor
  neardist3(i)=Y(1);   % distance to nearest neighbor
 end

 % Is d=2 the correct embedding dimension?
  count=zeros(1,n-d3+1);
   for i=1:n-d3+1
     if nearindex2(i) == n-d2+1, count(i)=NaN;
     elseif (abs(space3(3,i)-space3(3,nearindex2(i)))/(neardist2(i))) > 10
     count(i)=1;
     end
   end

 percent2=(sum(count(find(count>0))))/(length(count)-sum(isnan(count)))
 % this is percentage of global false nearest neighbors in dimension 2
 %-------------------------------------------------------------------------------
 % go to embedding dimension 4
 d4=4
 space4=zeros(d4,n-d4+1);
 for i=1:n-d4+1
  space4(:,i)=[x(n-i+1),x(n-i),x(n-i-1),x(n-i-2)]';
 end

 size4=zeros(n-d4+1,n-d4+1);
 for i=1:n-d4+1
 for j=1:n-d4+1
  if i==j, size4(i,j)=NaN;
  else if i>j, size4(i,j)=size4(j,i);
  else
    size4(i,j)=norm(space4(:,i)-space4(:,j));
  end
 end
 end
 end

 nearindex4=zeros(1,n-d4+1);
 neardist4=zeros(1,n-d4+1);
 for i=1:n-d4+1
  [Y,I]=sort(size4(i,:));
```

```
  nearindex4(i)=I(1);  % index of nearest neighbor
  neardist4(i)=Y(1);   % distance to nearest neighbor
 end

 % Is d=3 the correct embedding dimension?
  count=zeros(1,n-d4+1);
   for i=1:n-d4+1
     if nearindex3(i) == n-d3+1, count(i) = NaN;
     elseif (abs(space4(4,i)-space4(4,nearindex3(i)))/(neardist3(i))) > 10
     count(i)=1;
     end
   end

 percent3= (sum(count(find(count>0))))/(length(count)-sum(isnan(count)))
 % this is percentage of global false nearest neighbors in dimension 3
 %-------------------------------------------------------------------------------
 % go to embedding dimension 5
 d5=5
 space5=zeros(d5,n-d5+1);
 for i=1:n-d5+1
  space5(:,i)=[x(n-i+1),x(n-i),x(n-i-1),x(n-i-2),x(n-i-3)]';
 end

 size5=zeros(n-d5+1,n-d5+1);
 for i=1:n-d5+1
 for j=1:n-d5+1
  if i==j, size5(i,j)=NaN;
  else if i>j, size5(i,j)=size5(j,i);
  else
    size5(i,j)=norm(space5(:,i)-space5(:,j));
  end
 end
 end
 end

 nearindex5=zeros(1,n-d5+1);
 neardist5=zeros(1,n-d5+1);
 for i=1:n-d5+1
  [Y,I]=sort(size5(i,:));
  nearindex5(i)=I(1);  % index of nearest neighbor
  neardist5(i)=Y(1);   % distance to nearest neighbor
 end

 % Is d=4 the correct embedding dimension?
  count=zeros(1,n-d5+1);
```

```
  for i=1:n-d5+1
   if nearindex4(i) == n-d4+1, count(i)= NaN;
   elseif (abs(space5(5,i)-space5(5,nearindex4(i)))/(neardist4(i))) > 10
   count(i)=1;
   end
  end

percent4= (sum(count(find(count>0))))/(length(count)-sum(isnan(count)))
% this is percentage of global  false nearest neighbors in dimension 4
%-------------------------------------------------------------------------------
% go to embedding dimension 6
d6=6
space6=zeros(d6,n-d6+1);
for i=1:n-d6+1
 space6(:,i)=[x(n-i+1),x(n-i),x(n-i-1),x(n-i-2),x(n-i-3),x(n-i-4)]';
end

size6=zeros(n-d6+1,n-d6+1);
for i=1:n-d6+1
for j=1:n-d6+1
 if i==j, size6(i,j)=NaN;
 else if i>j, size6(i,j)=size6(j,i);
 else
  size6(i,j)=norm(space6(:,i)-space6(:,j));
 end
end
end
end

nearindex6=zeros(1,n-d6+1);
neardist6=zeros(1,n-d6+1);
for i=1:n-d6+1
 [Y,I]=sort(size6(i,:));
 nearindex6(i)=I(1);  % index of nearest neighbor
 neardist6(i)=Y(1);   % distance to nearest neighbor
end

% Is d=5 the correct embedding dimension?
 count=zeros(1,n-d6+1);
  for i=1:n-d6+1
   if nearindex5(i) == n-d5+1, count(i)=NaN;
   elseif (abs(space6(6,i)-space6(6,nearindex5(i)))/(neardist5(i))) > 10
   count(i)=1;
   end
  end
```

```
percent5= (sum(count(find(count>0))))/(length(count)-sum(isnan(count)))
% this is percentage of global false nearest neighbors in dimension 5
%-------------------------------------------------------------------------------
% go to embedding dimension 7
d7=7
space7=zeros(d7,n-d7+1);
for i=1:n-d7+1
 space7(:,i)=[x(n-i+1),x(n-i),x(n-i-1),x(n-i-2),x(n-i-3),x(n-i-4),x(n-i-5)]';
end

size7=zeros(n-d7+1,n-d7+1);
for i=1:n-d7+1
for j=1:n-d7+1
 if i==j, size7(i,j)=NaN;
 else if i>j, size7(i,j)=size7(j,i);
 else
   size7(i,j)=norm(space7(:,i)-space7(:,j));
 end
end
end
end

nearindex7=zeros(1,n-d7+1);
neardist7=zeros(1,n-d7+1);
for i=1:n-d7+1
  [Y,I]=sort(size7(i,:));
  nearindex7(i)=I(1);  % index of nearest neighbor
  neardist7(i)=Y(1);   % distance to nearest neighbor
end

% Is d=6 the correct embedding dimension?
 count=zeros(1,n-d7+1);
  for i=1:n-d7+1
   if nearindex6(i)== n-d6+1, count(i)=NaN;
   elseif (abs(space7(7,i)-space7(7,nearindex6(i)))/(neardist6(i))) > 10
    count(i)=1;
    end
  end

percent6= (sum(count(find(count>0))))/(length(count)-sum(isnan(count)))
% this is percentage of global false nearest neighbors in dimension 6

% plot results:
dim=[1:6];
```

```
perc=[percent1,percent2,percent3,percent4,percent5,percent6]
plot(dim,perc)
xlabel('embedding dimension')
ylabel('percentage of global false nearest neighbors')
```

APPENDIX C. PREDICTION ALGORITHMS (MATLAB)

One-step prediction is accomplished using a local linear technique:

```
load smx20000                    % contains previously generated data
delay=10                         % delay parameter
d=3                              % embedding dimension
r=1;                             % row index for results matrix
c=1;                             % column index for results matrix
results=zeros(10,9);

for n=[500,1000,2000,3000,4000,5000,6000,7000,8000,9000]
for k=[2,3,4,5,6,7,8,9,10]

% Create delayed time series from original time series:
y=smx20000(1:n);
x=zeros(1,floor(n/delay));
 for q=1:floor(n/delay)
  x(q)=y(delay*q);
 end
n1=length(x);


% Create state space: there are n-d+1 delay vectors in space
space=zeros(d,n1-d+1);
 for i=1:n1-d+1;
  space(:,i)=[x(n1-i+1),x(n1-i),x(n1-i-1)]';
 end


% Find k nearest neighbors:
size=zeros(1,n1-d);
q=1;
 for m=2:n1-d+1
  size(q)=norm(space(:,1)-space(:,m));
  q=q+1;
 end

% Get the k nearest neighbors out of the state space
[Y,I]=sort(size);
set=I(1:k);
covar=space(:,set+1);
```

```
% Do multiple regression:
h=zeros(1,k)';
j=1;
 for i=set+1
  h(j)=space(1,i-1); % for each nearest neighbor, get the 1st component
                    % of where that state went to next
 j=j+1;
 end

e=ones(k,1);
A=[e covar'];                    % prepares matrix of covariates for regression
beta= A\h;                       % computes regression coefficients

% Compute the prediction:
predict=beta'*[1; space(:,1)];
true=smx20000((delay*floor(n/delay))+delay);
onesteperror=true-predict;

% Display the results:
results(r,c)=abs(onesteperror);

if c==9,  c=1;
else  c=c+1;
end

end
r=r+1;
end

% Display results:
results
meanbycol=mean(results)          % for each column
stdbycol=std(results)
minbycol=min(results)
maxbycol=max(results)

meanbyrow=mean(results')         % for each row
stdbyrow=std(results')
minbyrow=min(results')
maxbyrow=max(results')

overallmax=max(max(results))
overallmin=min(min(results))
```

Multi-step, iterative, prediction is accomplished using a local-linear technique:

```
delay=2                          % delay parameter
d=3;                             % embedding dimension
n=1000;                          % number of data points in time series
k= 5;                            % number of nearest neighbors
T= 200 ;                         % number of steps to predict ahead

predict=zeros(1,T);
true=zeros(1,T);
onesteperror=zeros(1,T);

% Create delayed time series from original time series:
y=stagger2(1:n);                 % must specify existing time series
x=zeros(1,floor(n/delay));
 for q=1:floor(n/delay)
  x(q)=y(delay*q);
 end
n1=length(x);

t=1;
while t < (T+1)

% Create state space: there are n-d+1 delay vectors in space
space=zeros(d,n1-d+1);
 for i=1:n1-d+1;
  space(:,i)=[x(n1-i+1),x(n1-i),x(n1-i-1)]';
 end


% Find k nearest neighbors:
size=zeros(1,n1-d);
q=1;
 for m=2:n1-d+1
  size(q)=norm(space(:,1)-space(:,m));
  q=q+1;
 end

% Get the k nearest neighbors out of the state space
[Y,I]=sort(size);
set=I(1:k);
covar=space(:,set+1);

% Do multiple regression:
h=zeros(1,k)';
```

```
j=1;
 for i=set+1
  h(j)=space(1,i-1);                % for each nearest neighbor, get the 1st
                                    % component of where that state went to next
  j=j+1;
 end

e=ones(k,1);
A=[e covar'];                       % prepares matrix of covariates for regression
beta= A\h;                          % computes regression coefficients

% Compute the prediction:
predict(t)=beta'*[1; space(:,1)];
true(t)=stagger2((delay*floor(n/delay))+t*delay);    % must specify existing time series
onesteperror(t)=true(t)-predict(t);
x(floor(n/delay)+t)=predict(t);

t=t+1;
n1=length(x);

end

% Mean square error calculation:
summand1=zeros(1,T);
summand2=zeros(1,T);
E=zeros(1,T);

avgx=mean(true);
 for j=1:T
   summand1(j)=(true(j)-predict(j))^2;
   summand2(j)=((true(j)-avgx))^2;
 end

cumsum1=cumsum(summand1);
cumsum2=cumsum(summand2);

% Calculate prediction horizon:
phoriz=zeros(1,T);
 for m=1:T
  E(m)=(cumsum1(m))/(cumsum2(m));
   if E(m) >= 1, phoriz(m)=1;, end
 end

horiz=min(find(phoriz==1))
end
```

```
% Plot results:
plot(1:T,predict,'o',1:T,true)
xlabel('prediction step')
ylabel('time series value')
pause

figure
plot(1:T,E,1:T,E,'*')
xlabel('prediction step')
ylabel('normalized mean square error')
```

LIST OF REFERENCES

Abarbanel, Henry D.I., *Analysis of Observed Chaotic Data*, Springer-Verlag, 1995.

Box, G. E., and Jenkins, G. M., *Time Series Analysis, Forecasting and Control*, Holden-Day, 1976.

Brock, W. A., "Distinguishing Random and Deterministic Systems: Abridged Version," *Journal of Economic Theory*, Vol 40, 1986.

Casdagli, Martin, "Nonlinear Prediction of Chaotic Time Series," *Physica D*, Vol 35, 1989.

Casdagli, Martin, Stephen Eubank, J. Doyne Farmer, and John F. Gibson, "An Analytic Approach to Practical State Space Reconstruction," *Physica D*, Vol 57, 1992.

Casdagli, Martin, and Stephen Eubank, *Nonlinear Modeling and Forecasting*, Addison-Wesley Publishing Company, 1992.

Chatfield, C., *The Analysis of Time Series*, Chapman and Hall, 1989.

Crutchfield, J. P., J. D. Farmer, H. Froehling, N. H. Packard, and R. S. Shaw, "On Determining the Dimension of Chaotic Flows," *Physica D*, Vol 3, 1981.

Crutchfield, J. P., J. D. Farmer, N. H. Packard, and R. S. Shaw, "Geometry from a Time Series," *Physical Review Letters*, Vol 45 No 9, 1980.

Farmer, J. D., "Chaotic Attractors of an Infinite-Dimensional Dynamical System," *Physica D*, Vol 4, 1982.

Farmer, J. D., and John J. Sidorowich, "Predicting Chaotic Time Series," *Physical Review Letters*, Vol 59, No 8, American Physical Society, 1987.

Fitzgerald, William J., Christopher Molina, Mahesan Niranjan, and Nick Sampson, "Geometrical Techniques for Finding the Embedding Dimension of Time Series", University of Cambridge, Cambridge, England, 1996.

Flandrin, Patrick, and Olivier Michel, "Introduction of Higher Order Statistics for Estimating the Dimension of Chaotic Time Series", *Ecole Normale Superieure de Lyon*, 1990.

Friedman, Jerome H., "Multivariate Adaptive Reression Splines," *The Annals of Statistics*, Vol 19 No 1, 1991.

Gershenfeld, Neil A., and Andreas S. Weigend, *Time Series Prediction: Forecasting the Future and Understanding the Past*, Addison-Wesley Publishing Company, 1994.

Kuguimtzis, D., B. Lillekjendlie, and N. Christophersen, "Chaotic time Series Part I: Estimation of Invariant Properties in State Space," Department of Informatics, University of Oslo, Pb. 1080 Blindern, N-0316 Oslo, Norway, 1994.

Kuguimtzis, D., B. Lillekjendlie, and N. Christophersen, "Chaotic time Series Part II: System Identification and Prediction," Department of Informatics, University of Oslo, Pb. 1080 Blindern, N-0316 Oslo, Norway, 1994.

Liebert, W., and H. G. Schuster, "Proper Choice of the Time Delay for the Analysis of Chaotic Time Series," *Physics Letters A*, Vol 142 No 2, 1989.

May, Robert M., and George Sugihara, "Nonlinear Forecasting as a Way of Distinguishing Chaos From Measurement Error in Time Series," *Nature*, Vol 344, 1990.

Ruelle, D., "Deterministic Chaos: the Science and the Fiction," *Proc. R. Soc. Lond. A*, Vol 427, 1990.

Sidorowich, John J., "Modeling of Chaotic Time Series for Prediction, Interpolation, and Smoothing," Institute for Nonlinear Science, University of California at San Diego, La Jolla, California, 1992.

Strogatz, Steven H., *Nonlinear Dynamics and Chaos*, Addison-Wesley Publishing Company, 1994.

Tong, Howell, *Nonlinear Time Series: A Dynamical Systems Approach*, Clarendon Press, 1990.

Volterra, V., *Theory of Functionals and of Integral and Integro-Differential Equations*, Dover Publishing, 1959.

Wolf, Alan, Jack B. Swift, Harry L. Swinney, and John A. Vastano, "Determining Lyapunov Exponents from a Time Series," *Physica D*, Vol 16, 1985.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Director, Training and Education 1
   MCCDC, Code C46
   1019 Elliot Rd.
   Quantico, Virginia 22134-5027

4. Director, Marine Corps Research Center 2
   MCCDC, Code C40RC
   2040 Broadway Street
   Quantico, Virginia 22134-5107

5. Director, Studies and Analysis Division 1
   MCCDC, Code C45
   3300 Russell Road
   Quantico, Virginia 22134-5130

6. Marine Corps Representative 1
   Naval Postgraduate School
   Code 037, Bldg. 234, HA-220
   699 Dyer Road
   Monterey, California 93940

7. Marine Corps Tactical Systems Support Activity 1
   Technical Advisory Branch
   Attn: Maj J. C. Cummiskey
   Box 555171
   Camp Pendleton, California 92055-5080

8. Chairman 4
   Code MA/Wo
   Naval Postgraduate School
   Monterey, California 93943-5000

9. Professor Christopher Frenzen 3
Code MA/Fr
Naval Postgraduate School
Monterey, California 93943-5000

10. Phil Beaver 1
Code MA/Be
Naval Postgraduate School
Monterey, California 93943-5000

11. Capt. Mary L. Leonardi 2
1526 W. Belle Plaine
Chicago, Illinois 60613

12. Patricia Leonardi 1
1526 W. Belle Plaine
Chicago, Illinois 60613