Nonlinear Modeling and Identification for Process Control

Thesis by

Carl Rhodes

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

1998

(Submitted October 14, 1997)

Acknowledgements

There are a number of people I would like to thank for their support and encouragement throughout the development of this thesis. My Ph.D. advisor Manfred Morari has been a constant source of wisdom. His high standards are something which I will always admire and which have helped to shape this work in many positive ways. I am thankful that he allowed me a great deal of intellectual freedom during my research and granted me the opportunity to present my work at many different conferences.

The faculty at Caltech should be acknowledged for providing me with wonderful instruction in the areas of chemical engineering and control and dynamical systems. I would especially like to thank Richard Murray, John Brady, John Doyle, and Stephen Wiggins for their work in making those first two years of classwork so valuable. I am also grateful for the hardworking staff in the Chemical Engineering department at Caltech. Kathy Bubash was always willing to go the extra mile for me when I needed paperwork or information.

During my research I was fortunate to have the chance to work closely with several excellent researchers including Lev Tsimring, Henry Abarbanel, and Stephen Wiggins. Their insight from fields outside of control helped me to examine my research problems from a different perspective. Franta Kraus and Frank Allgöwer have also been great sources of knowledge throughout my research.

My colleagues at Caltech and ETH provided me with both intellectual stimulation as well as friendship. Some members of the Morari group that I have been lucky enough to work with are Richard Braatz, Tyler Holcomb, Nikos Bekiaris, George Meski, Alex Zheng, Simone de Oliveira, Matt Tyler, Mayuresh Kothare, Iftikhar Huq, Thomas Güttinger, Cornelius Dorn, Hiroya Seki, Jimmy Wells, Daniel Schreiber, and Domenico Mignone.

I would like to thank the following agencies who funded this research: the Chevron Foundation, the Shell Foundation, the Landau Foundation, the Department of Energy, and the Swiss Federal Institute of Technology.

Spencer, the pug dog, gave me lots of love over the past year. All he asked for in return was two walks a day, a bowl of food at night, and a place to curl up and sleep.

The last person I would like to thank is certainly the most important one, my wife Alexandra. She is the one who kept me going during the past five years. Not only did she stand by me as my best friend and support me throughout my studies and research, she also sacrificed a great deal by agreeing to move to Switzerland. Without her, none of this would have ever been possible.

Abstract

Ideally, processes to be controlled would behave in a linear manner so that well-developed methods of linear control could be applied directly. However, environmental regulations and increased competition are forcing these processes to operate in regions where the assumptions of linearity tend to break down. There has been a great deal of recent academic interest in the control of nonlinear systems, but there are relatively few applications of these methods in industry. One major reason may be the lack of tools for developing models suitable for nonlinear control schemes.

A number of tools that can be used in the modeling of nonlinear systems for process control are presented in this thesis. In the first section, the problem of determining the proper regression vector size for black-box modeling is examined. The false nearest neighbors algorithm (FNN) is suggested as a solution for this problem. Extensions, analysis, and numerous applications of the FNN algorithm are given and the algorithm is seen to be a useful tool in the identification of nonlinear models.

In the second section of the thesis, the problem of nonlinear model reduction for systems exhibiting large time-scale separations is examined. A method of determining the reduced order manifold of slow dynamics is outlined and it is proved that this algorithm identifies the proper manifold. Some thoughts on how the results of the algorithm can be used for developing reduced models are presented.

In the third section, the concept of data-based control is introduced. This method of control attempts to utilize process data directly through local modeling techniques. Some preliminary work in this area is given for trajectory tracking and computing controllable sets and data-based control is successfully applied to an experimental electrical circuit. Finally, some thoughts on possible future work in this field are presented.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1 Introduction

In order to apply any method of control design more advanced than trial and error, a suitable mathematical model of the process to be controlled is needed. Preferably, the model of the process to be used in controller design should be both accurate and compact. The model needs to accurately describe the dynamics of the system because the controller will be designed to meet certain performance specifications using this model as a reference. If the model does not capture the dynamics of the process correctly, there may be problems when the controller is implemented on the physical process. While robust control methods can allow controllers to deal with small amounts of plant/model mismatch, better performance can be achieved if a more accurate model of the process is developed.

On the other hand, controller design may become difficult or computationally impossible for mathematical models which are too large or too complicated. This is especially true for nonlinear systems where controllers are commonly developed using algebraic manipulations or nonlinear optimizations. Models developed for control purposes have very different requirements than those used for simulation studies. Models developed for simulation studies are designed to be highly accurate, often sacrificing computational simulation speed for dynamic accuracy. Models developed for control purposes need to be of a size and complexity such that they are useful for control design algorithms.

Notice that a tradeoff exists between the two requirements of accuracy and compactness. A large model, if designed correctly, will always be as accurate or more accurate than a smaller model. Models which are best suited for controller design are known as *parsimonious* models (Söderström and Stoica, 1989). This means that the model should be the smallest one which accurately describes the dynamic behavior of the system.

Mathematical models of physical processes are commonly developed in two different ways: first-principles modeling and black-box identification. First-principles models are developed by modeling the fundamental physics of the process (conservation laws, chemistry, etc.) in order to describe the dynamics of the system. In order to develop first-principle models, an expert's understanding of the physical behavior of the process and the fundamental laws needed to model the system is needed. When properly designed, these models tend to represent the dynamics of a system extremely well. However, these models can be very large and reducing the model based solely on the physical understanding of the problem can be difficult or impossible.

On the other hand, black-box models are developed using identification techniques which don't require any physical understanding of the process. Instead, experiments are performed to produce dynamical data for the system of interest. After these identification experiments, a model is found directly from the experimental data. The first systematic method of identification for linear time-series was proposed by Box and Jenkins (1970). Current method of identification for input/output systems build directly on these methods.

The first step in black-box modeling is determining the model structure to be used. Once the structure is determined, the parameters associated with that specific structure are found and a model is developed. By restricting the model structure to be of a certain size, the best model associated with that structure and complexity can be found. For this reason, restricting the complexity of black-box models is relatively easy.

Black-box modeling also has a number of drawbacks. Black-box models have a limited range of validity. For many systems, it is impossible to describe the dynamics outside of the limited region where dynamical data from the process is available. The data may also contain significant amounts of noise which must be accounted for during the identification process. In addition, little physical insight into the process dynamics is gained from black-box modeling when compared to building models based on first-principles. This may be important since a physical understanding of process dynamics may be helpful in controller design.

The black-box identification of linear models is a well-studied problem, and there are a number of good detailed references which provide a systematic framework for developing linear models (Ljung, 1987; Söderström and Stoica, 1989). There are also a number of computational packages available for performing linear identification (e.g. Ljung (1986)). Linear identification methods have traditionally worked well in conjunction with linear control methods. However, there is a drawback to these linear methods.

While a linear model may be accurate for a certain operating region, the majority of real world processes exhibit nonlinear dynamics. In the past, processes were able to operate a small operating region where the system behaves in a linear fashion. However, increasingly strict environmental regulations and economic pressures are forcing processes into operating regimes where the assumption of linearity break down and tools for nonlinear identification and control are needed. While there has been a great deal of work in the control of nonlinear systems, relatively few applications of these methods have been published in the literature. One major reason may be the lack of suitable models for nonlinear control.

A possible solution to two common problems encountered when forming nonlinear models for control purposes is presented in this thesis. Additionally, a new method for the control of nonlinear systems which utilizes local modeling is introduced. A list of the important questions addressed by this thesis follows:

- **What is the proper "model order" of a nonlinear black-box model?** When performing system identification, the proper model structure needs to be determined. Part of determining the proper model structure is determining the number of terms to be included in the regression vector. Only after the model structure has been specified can the parameters of the given model be determined.

  In order to be certain that the model is parsimonious, the smallest number of terms should be included in the regression vector that allow an accurate model to be developed. The false nearest neighbors (FNN) algorithm will be used as a tool for determining the proper dimension of a regression vector when forming models for process control. The algorithm uses a simple geometric criterion for determining the proper embedding dimension, and as a result a global model of the dynamics is not needed for solving this problem.

In this thesis, the FNN algorithm will be applied for the first time in the analysis of input/output time-series. The FNN algorithm has also been extended for inferential measurement selection in nonlinear systems. Problems associated with the analysis of noise corrupted time-series have also been discovered. These problems are illustrated and an extension to the algorithm is proposed to correct these problems. Numerous examples are presented in order to show how the FNN algorithm can play a useful role in nonlinear identification.

- **How can an accurate, but complex nonlinear model be reduced in a way such that the "important" dynamics are preserved?** While first-principle models can provide an accurate dynamical descriptions of a process, often models designed in this manner will have a description which is too large for nonlinear control design tools to be utilized. For linear systems, a number of methods exist for performing model reduction. The goal of model reduction is to form a model which matches dynamics of the complete model as closely as possible in a smaller description.

  For nonlinear systems, little work in the field of model reduction has been published. A method of model reduction will be proposed for nonlinear systems which exhibit dynamics on different time-scales. For control purposes, the fast dynamics of such system may not be important if they are stable. By using model reduction to "truncate" the fast dynamics, a smaller description of the dynamics which are important for control purposes can be developed.

  In this thesis, a method of identifying the reduced order manifold of slow dynamics in the state-space will be presented. The method, which was originally developed for combustion problems, will be justified rigorously for the first time. This justification results from a change of coordinates known as Fenichel coordinates. In addition, some new ideas on how the results of this algorithm can be used for forming reduced models will be introduced.

- **Is it possible to utilize local models to perform "data-based control"?** Traditionally, controllers are designed by using a global model of the process dynamics. First, an identification experiment is performed where data are collected. With these data, a model of the system is built using the black-box methods described earlier. Once a model of the system is found, a controller is built to meet certain performance specifications for that model. The last step involves the implementation of the controller on the actual process.

  However, a new class of "local models" have been successfully applied to chaotic time-series prediction. This new class of models does not utilize a single global model. Instead, a number of locally valid models are built using data which are in the "local neighborhood" of interest. In order to build a model, the identification data are searched for data which are close to the dynamical region of interest. After the neighboring data are found, simple models which are valid only locally can be built and used for prediction.

  In this thesis, the first steps in developing a control scheme based on these local models will be introduced. Instead of a global model, the control scheme relies upon the identification data and local models to determine the appropriate control move. In this new framework, an identification experiment would be performed and the controller would use the identification data directly. Forming a global nonlinear model from the identification data is not necessary. By utilizing large amount of identification data rather than a global nonlinear model which approximates the identification data, it is hoped that better results can be achieved. Some preliminary studies and thoughts for the future of this area are presented.

An overview of this thesis

The thesis is organized into 3 major parts. In Chapters 2-5, the problem of model order determination in nonlinear black-box identification is discussed and the false nearest neighbors (FNN) algorithm is presented as a solution. Chapter 2 gives an introduction to the problem and Chapter 3 reviews the major theoretical work in the area of time-delay embedding which motivates the FNN algorithm. Problems which can arise when using the FNN algorithm due to noise corruption are illustrated in Chapter 4 and a new threshold test is presented to solve these problems. The application of the FNN algorithm to the problems of input/output time-series identification and inferential measurement selection is presented in Chapter 5.

In Chapter 6, the problem of nonlinear model reduction is introduced. Systems which exhibit time-scale separation are discussed in this section, and singular perturbation theory is used to motivate the specific model reduction problem that will be considered. A method of identifying the reduced dimensional invariant manifold associated with time-scale separation is presented and it is shown rigorously that this method identifies the proper manifold. Examples are presented, and some thoughts on how this method can be used to determine low-order models are given.

Chapters 7-10 introduce the concept of data-based control. An introduction to some background material which motivates these data-based methods is presented in Chapter 7. A method of computing controllable sets using a data-based framework is presented in Chapter 8. A simple algorithm which uses data-based control ideas is given in Chapter 9, and the algorithm is applied to a number of examples in an offline manner. An overview of research for the future in data-based control is given in Chapter 10, along with a sketch of how these methods could be used for control.

Finally, an overall summary and list of suggested future work in the general area of modeling and identification is presented in Chapter 11.

Part II

Model order determination for nonlinear systems

Chapter 2 Introduction

2.1 Motivation

In order to use standard tools for controller design, first a suitable model of the process needs to be developed. While it is always preferable to use physical knowledge of the process, in some cases building models from first-principles is not practical. One reason is that first-principle models of chemical processes can be very large. Complete models based on physics consisting of hundreds of states are not uncommon for many chemical processes. For example, the simplest model of a distillation column consists of a single ODE for each tray. More complex models consist of numerous ODEs per tray. Since it is not uncommon to have distillation columns with hundreds of trays in industry, it is easy to see that models of columns developed in this way will have a high dimension. While models of this size may not be problematic for simulation, current methods of nonlinear control are not equipped to deal with such large descriptions. A second reason that physical models may not be sufficient for control purposes is that there may be effects which are either difficult or impossible to account for in the physical modeling process. Examples of this for a distillation column example include unmodeled reactions, incomplete mixing, and unmodeled thermodynamic effects.

For these reasons, in some situations it is necessary to use nonlinear "black box" identification methods to develop a model of the process dynamics. Recently, there has been a great deal of work examining the black-box identification of nonlinear systems using input/output data. A good overview of the work in this field can be found in Sjöberg, Zhang, Ljung, Benveniste, Delyon, Glorennec, Hjalmarsson and Juditsky (1995). While most of the recent work has focused on utilizing neural networks, other recently published works have examined nonlinear ARMAX model structures (Chen and Billings, 1989), radial basis functions, wavelets, hinging hyperplanes (Beiman, 1993), and multivariate adaptive regression splines (MARS)

(Friedman, 1991).

The common focus of all these works is determining and approximating the functional relationship between a regression vector and an output vector. In other words, a set of observed regressors $\psi(t)$ for $t = 1 \ldots N$ and observed outputs $y(t)$ related to that regression vector are given. All these methods attempt to find a functional relationship

$$y(t) = G[\psi(t)] \tag{2.1}$$

which minimizes some error function E which is often of the form

$$E = \sum_{i=1}^{N} \mathrm{Err}[y(i) - G[\psi(i)]] \tag{2.2}$$

where Err is typically some norm operator. Commonly, the regression vector for single-input/single-output systems consists of delayed versions of the input and output.

$$\psi(t) = [y(t-\tau), y(t-2\tau), \ldots, y(t-l\tau), u(t-\tau), \ldots, u(t-m\tau)] \tag{2.3}$$

While many works focus on ways of parameterizing and determining the optimal function G, there is relatively little work on determining the proper form of the regressor ψ for nonlinear systems. Specifically, in nearly all of these studies the number of delayed terms in the regression vector (l and m) is assumed to be known. The function G is then estimated using the observed regressors and outputs of the time-series. If the differences between the values of the approximated function $G[\psi(t)]$ and the actual output $y(t)$ are large, there could be two sources of the error. The first is an estimated function G which does not do a good job of representing the relationship between the regressor and the output. The second possibility is the regressors $\psi(t)$ simply do not contain enough information to accurately predict the future outputs. If the prediction error is large, it is impossible to tell from the output of these commonly utilized algorithms whether the source of the error is incorrect functional approximation or a regression vector which does not contain enough terms

[1].

In the identification of linear systems, determining the exact source of the error is not a problem. Due to the linearity of the system, choosing the regression vector is the only critical step since the functional relationship is defined by linearity as

$$y(t) = a_1 y(t-\tau) + \cdots + a_l y(t-l\tau) + b_1 u(t-\tau) + \cdots + b_m u(t-m\tau). \tag{2.4}$$

Since computing the unknown parameters in these linear models involves relatively little computational time (an ordinary least-squares problem), determining the proper regression vector involves computing the optimal linear model for various regression vectors and using some function (such as AIC or F-Test) to determine when the error no longer significantly decreases with respect to the complexity resulting from additional terms (Söderström and Stoica, 1989).

For linear models, the error associated with a given model will decrease with an increasing number of parameters. The decrease in error is then weighed against the complexity associated with a larger model. The Akaike information criterion (AIC) is a well-known method of model order determination for linear identification (Söderström and Stoica, 1989) defined as

$$\text{AIC} = N \log V(\theta) + 2p \tag{2.5}$$

where N is the number of training data, $V(\theta)$ is the error associated with a given model, and p is the number of parameters in the model. The optimal model order is then determined by minimizing the AIC.

While there has been some work in utilizing these types of information criteria for nonlinear systems, these types of methods are often infeasible since nonlinear optimizations typically require large amounts of computational effort. In addition, it is very difficult to guarantee that the parameters of the nonlinear model will converge to an optimal solution. The proper function G from Equation (2.1) needs to deter-

[1]The MARS modeling scheme is the one exception. The algorithm "trims" away terms in the regressor which do not significantly reduce the prediction error.

mined. If G is not of the proper form, it is impossible to determine the minimum error associated with a given model order. It is also difficult to determine the minimal error associated with a given model order in the case where computing the parameters associated with a given structure is a nonconvex problem. In that case, it may be impossible to ensure that the optimal solution has been found.

For nonlinear systems, it can be argued that it would be preferable to break the identification process into two distinct parts. First, the proper regression vector should be determined. The proper regression vector should contain only those terms needed to accurately predict the output. Once the proper regression vector is determined, then the functional relationship between the regressor and the output can be estimated.

While there is extensive work in determining the proper regression vector for linear systems, relatively little work exists in the field of nonlinear systems. A method based on geometric ideas for determining the proper dimension for the regression vector called the false nearest neighbor algorithm (FNN) will be presented here. This method was first developed for the analysis of self-driven chaotic time-series (Abarbanel, Brown, Sidorowich and Tsimring, 1993; Abarbanel, 1996), but in this thesis extensions of these methods for input/output data, inferential systems, and noise-corrupted time series will be presented.

Another approach to the model order determination problem for nonlinear systems is the statistical approach of (Poncet and Moschytz, 1994). In this method, an estimate of the optimal prediction error is found. This is done by forming regression vectors $\Psi_l(t) = [y(t-\tau), \ldots, y(t-l\tau)]$ from the data for a given dimension l. The optimal prediction for $y(t)$ from the regressor $\Psi_l(t)$ is given by the conditional expectation $g_0(\Psi_l(t)) = E(y \mid \Psi_l = \Psi_l(t))$. The lower bound of the prediction error in this framework is then $\sigma_l^2 = \mathrm{Var}(Y \mid \Psi_l)$ which is the conditional variance. This variance can be estimated by the quantity $\delta_l^2(\epsilon) = E[\frac{1}{2}(y(t) - y(t'))^2 \mid \|\Psi_l(t) - \Psi_l(t')\| \leq \epsilon]$. This is simply the variance of the outputs for regression vectors which are within some prespecified distance in the regressor space. By analyzing the average of δ_l^2 over the regression space, an estimate of σ_l^2 is found. When $\delta_l^2(\epsilon)$ fails to decrease significantly as a function of increasing dimension l, the prediction error no longer decreases with increasing dimension and the proper regression vector is found.

A third method is similar to the AIC, however it is used to identify the proper dimension of hidden Markov models (HMM) (Finesso, 1990). This is a special type of model which the output is assumed to be a function of a Markov sequence. However in order to compute the information criterion off this method, a model of the process needs to be computed for each dimension of interest. This may be problematic because of the computational time associated with forming these models.

2.2 Theoretical background

Mathematical models of processes to be control often take the following form

$$\begin{aligned} \dot{x} &= f(x,u) && (2.6) \\ y &= h(x) && (2.7) \end{aligned}$$

where $x \in \Re^n$ is the state vector of the system, u is a controlled input, and y is a measured output. While models arrived at by first-principles can do a good job of representing the major features of the dynamics of physical systems, the dynamics of the mathematical model and the actual physical system can be quite different. This can be the result of unmodeled dynamics and/or errors in the parameters contained in functions f and h. Since it may not be possible to measure all the states because of either economic or physical limitations, determining the unknown parameters of the model (Equation 2.6) can be difficult. In addition, there may be unmodeled dynamics of the system.

For these reasons, it may be desirable to compute a model of the system dynamics directly from input/output data of the physical system. In mathematical terms, a model of the form

$$y(t) = G[y(t-\tau), y(t-2\tau), \ldots, y(t-l\tau), u(t-\tau), \ldots, u(t-m\tau)] \qquad (2.8)$$

should be found where τ is the sampling time of the system. A few interesting questions arise naturally from this discussion. Assume the system to be modeled is known exactly (say it is Equation 2.6) and the controlled input (u) only changes at the sampling times of the system.

1. Does a representation in the form of Equation 2.8 exist?
2. How many delayed terms (l and m) are needed to represent the input/output dynamics of Equation 2.6 if the size of the state-space is known ($x \in \Re^n$)?
3. Can the function G be determined directly from the functions f and h?

If the functions f and h are linear, the answers to these questions are known. An equivalent representation of the input/output dynamics does exist, and a number of terms $l = m = n$ is sufficient to represent the input/output dynamics of the system described by Equation 2.6. In addition, for linear systems the function G can be determined exactly from the functions f, h, and the time delay τ using the z-transformation (Franklin, Powell and Emami-Naeini, 1986).

When the functions f and h are nonlinear, the answers to these questions are not as simple. The well-known results for linear systems are based strongly on the linearity of the system. Specifically, the term G can be calculated because the dynamics are invariant with regards to the location of the trajectory in the state-space of the system. For nonlinear systems, computing the function G from f and h is impossible (using current methods) except in trivial cases. For this reason, different methods must be utilized. While the methods used to solve these problems will be discussed in detail in Chapter 3, a short introduction will be given here.

To get some insight into the problem, first the case of autonomous (self-driven) systems will be examined. The following question was first studied in the analysis of chaotic time-series. Given the following system

$$\dot{x} = f(x) \tag{2.9}$$

$$y = h(x) \tag{2.10}$$

where $x \in \Re^n$, it was shown by Takens that the output at any point in time can be generically written as a non-linear function of time-delayed versions of that same output (Takens, 1981). In other words, there exists some G such that

$$y(t) = G[y(t-\tau), y(t-2\tau), \ldots, y(t-l\tau)]. \tag{2.11}$$

Additionally, it was shown by Takens that $l > 2n$ is a sufficient condition for this relationship to exist. The proof is based on the Whitney embedding theorem of differential geometry which states that any k dimensional manifold can be embedded in the space $\Re^{2k+1}$ (Guillemin and Pollack, 1974). Takens showed that an embedding exists between the state-space of the original system and the time-delayed version of the system. Since an embedding is a nonlinear one-to-one relationship, the original state-space dynamics also exist in the delayed coordinates. These ideas were later extended by another set of authors (Sauer, Yorke and Casdagli, 1991) to a slightly more general result and a separate result relevant to inferential prediction which will be discussed in Section 5.5.

For input/output systems, a similar result to that of Takens was first suggested by Casdagli (1992). In this paper, a brief outline of the methods needed to prove an embedding theorem for nonlinear systems was given. For discrete time systems, these results were recently formalized by Poncet, Poncet and Moschytz (1995) using the methods outlined by Casdagli. In this paper, it was shown that $n+1$ delayed versions of the input and output is a sufficient condition to represent the dynamics of a system with n states under standard assumptions such as state observability. While a formal proof does not yet exist for continuous time systems, the results should be similar to the discrete-time result.

While these results are interesting from a mathematical perspective, they are little help when it comes to identifying the number of delay terms needed to model a system from input/output data. For reasons of parsimony, it is preferable to determine the smallest possible number of delay terms needed to represent the dynamics. In most identification problems, the dimension of the system generating the time-series is unknown. In addition even when the dimension of the state-space system is known *a priori*, there may exist a smaller representation of the dynamics (since the condition is sufficient and not necessary). For these reasons, there is a need to determine the proper "dimensionality" of the system in the time-delay description directly from input/output data.

Chapter 3 A review of embedding theorems for nonlinear dynamical systems

3.1 Introduction

As the study of chaotic systems progressed, scientists wished to perform experiments to validate numerical simulations which showed the existence of strange attractors in order to be certain that these phenomena occur in physical systems. However, one major difficulty encountered when experimental studies were performed is that it is impossible to accurately measure all the states of the physical system. Many of the experimental studies designed to validate chaotic simulations were in the field of fluid mechanics. In order to measure all the states for a system, numerous physical properties of the fluid at a single point would have to be accurately determined which is physically impossible. On the other hand, in simulations this is not an issue since all of the states are computed as a function of time during the simulation process.

In order to solve this problem, experimental researchers needed to find a method of "recreating" the state-space variables without actually measuring all of the states of the system. While there was initially some hope that the power spectrum of the data could be used to confirm the presence of strange attractors (Ruelle and Takens, 1971), it was soon seen that information from the power spectrum could not be used to "reconstruct" the strange attractor (Takens, 1981). In order to recreate the state space a different method which utilized a single output of the experimental system was needed.

Another possible method of reconstructing the state space dynamics was then developed by researchers at University of Santa Cruz (Packard, Crutchfield, Farmer and Shaw, 1980). The idea behind this method was to approximate a number of the derivatives of a single state by using a finite difference method. The first derivative is approximated by

$$\frac{dy}{dt}(t) \approx \frac{[y(t) - y(t - \tau_s)]}{\tau_s}. \tag{3.1}$$

Higher order derivatives can be approximated in the same fashion by including more delayed terms of the same output. Each derivative of higher order would include another delayed term. From these approximations of the derivatives of a single state, researchers believed that there might be some connection to the original states defined by the differential equations producing the behavior.

Instead of approximating the actual derivatives, the group at Santa Cruz used the delayed terms themselves to form a new set of coordinates to create a new space where the dynamics take place. This new l-dimensional coordinate system is defined as

$$\psi(t) = [y(t), y(t - \tau_s), \ldots, y(t - (l - 1)\tau_s)]. \tag{3.2}$$

The relationship between these "time-delayed" versions of the single output y and the original state space which defines the dynamics is unknown. However as long as a relationship exists between these two sets of coordinates which is smooth, it was hypothesized that structures which appear in the original state space coordinates would persist in these new coordinates.

Remember that it is possible to predict the future behavior of an autonomous system by making an accurate measurement on the state of the system when the state space dynamics are known. If the relationship between the state-space coordinates and the "time-delay" coordinates is smooth and one-to-one, it should be possible to predict the future behavior of the system from the time-delay coordinates. This is one reason why the existence of a smooth mapping between state-space and time-delay coordinates is so important to researchers.

For linear systems, this relationship between state-space and discrete-time input/output descriptions is well-known. The input/output dynamics of system

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{3.3}$$

where $x \in \Re^n$, $y \in \Re$, and $u \in \Re$ can be represented by

$$y(t) = a_1 y(t-1) + \cdots + a_n y(t-n) + b_1 u(t-1) + \cdots + b_n u(t-n) \tag{3.4}$$

where a_i and b_i are linear coefficients that are computed from A, B, and C and the sampling time of the system. The transformation between the continuous-time and discrete-time dynamics is known as the z-transform in the control literature (Franklin et al., 1986). The z-transformation relies heavily on the linearity of the original system. For systems which exhibit nonlinear dynamics, different tools are needed to prove similar final results. In order to understand these results some basic knowledge from differential geometry is needed.

3.2 Mathematical background

In this section, some mathematical tools from differential geometry are presented. These tools are utilized extensively in proving the embedding theorems which will be presented later. A majority of the background in this section is also given in Guillemin and Pollack (1974).

A fundamental structure of differential geometry is an object called a manifold. Manifolds are simply surfaces which locally look "just like a small piece of Euclidean space" (Guillemin and Pollack, 1974). Common examples of manifolds include the surface of a sphere or cylinder. An example of a surface which is not a manifold is a pyramid since the vertices of the pyramid do not look locally like a plane.

In order to formalize the definition of manifolds mathematically, a mapping called a diffeomorphism is used.

Definition 3.2.1 *A smooth map $f : X \to Y$ of subsets of two Euclidean subspaces is a* ***diffeomorphism*** *if it is one-to-one and onto and if the inverse map $f^{-1} : Y \to X$ is also smooth.*

Using the diffeomorphism, the definition of a manifold can be given.

Definition 3.2.2 *If X is a subset of a Euclidean space $\Re^n$, then X is a k-**dimensional manifold** if it is locally diffeomorphic to $\Re^k$.*

This means that for all points on X, there exists a diffeomorphism which maps the local neighborhood of X to an open set $Y \subset \Re^k$.

For a smooth map $f : X \to Y$, the derivative mapping at a point x where $y = f(x)$ is given by $df_x : T_x(X) \to T_y(Y)$. The derivative mapping is a mapping between two tangent spaces ($T_x(X)$ and $T_y(Y)$). If $\dim X \leq \dim Y$ and the mapping $df_x : T_x(X) \to T_y(Y)$ is injective, f is said to be an **immersion at** x. This means that the matrix df_x is full rank when evaluated at x. When f is an immersion for all points, it is simply called an immersion.

However even for cases where f is an immersion, it may not be a diffeomorphism. Take the mapping which twists a circle into a figure eight given in Figure 3.1. The

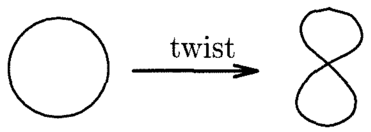


Figure 3.1: An immersion which is not a diffeomorphism

mapping is not a diffeomorphism (and the figure eight is not a manifold) since the mapping is not one-to-one in the neighborhood where the figure eight intersects itself. Another problem can arise even when the mapping is one-to-one. Take the mapping given in Figure 3.2. In this case, the inverse of the mapping is no longer smooth in

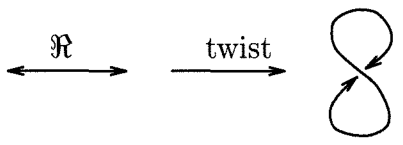


Figure 3.2: An immersion which is one-to-one and not a diffeomorphism

the neighborhood where the mapping nearly intersects itself.

A map $f : X \to Y$ is called proper if the preimage of all compact sets in Y is compact in X. A immersion which is both injective and proper is known as an embedding. When X is a compact manifold, an embedding is simply a one-to-one immersion. Using this definition of an embedding, the following theorem can be stated.

Theorem 3.2.1 (Guillemin and Pollack (1974)) *An embedding $f : X \to Y$ maps X diffeomorphically onto a submanifold of Y.*

This states that a manifold can be represented by a smooth change of coordinates as a submanifold in an equal or higher dimension.

In the case of time-delay embeddings, the mapping is defined by delayed versions of the output. Since it can't easily be determined if this mapping is an embedding, it would be promising if some results that suggested an embedding between the state-space and the time-delay coordinates might exist. In this case, the Whitney embedding theorem is promising since it states that an embedding exists which maps a manifold into a higher dimensional Euclidean space.

Theorem 3.2.2 (Whitney Theorem (Guillemin and Pollack, 1974)) *Every k-dimensional manifold embeds in $\Re^{2k+1}$.*

From this, it is clear that it possible to embed any manifold in a Euclidean space of high enough dimension.

The geometric reasoning behind the Whitney embedding theorem is best explained in a review by Abarbanel (Abarbanel, 1994). In a space of dimension d, a subspace of dimension d_1 and another subspace of dimension d_2 generally intersect in a subspace of dimension $d_{intersect} = d_1 + d_2 - d$. If $d_{intersect} < 0$, then the two subspaces will not intersect in general. Now assume the two subspaces have the same dimension as the manifold to be embedded (k). When the manifold is embedded in a space of dimension d, then the manifold should not intersect itself in general when $d > 2k$. If the manifold happens to intersect itself when $d > 2k$, a small perturbation to the manifold will get rid of the self-intersection.

An example of this self-intersection can be seen by embedding the one-dimensional manifold given in Figure 3.3. According to the Whitney embedding theorem, a space of dimension $d > 2$ is needed to embed the manifold. In $\Re^2$, the manifold intersects

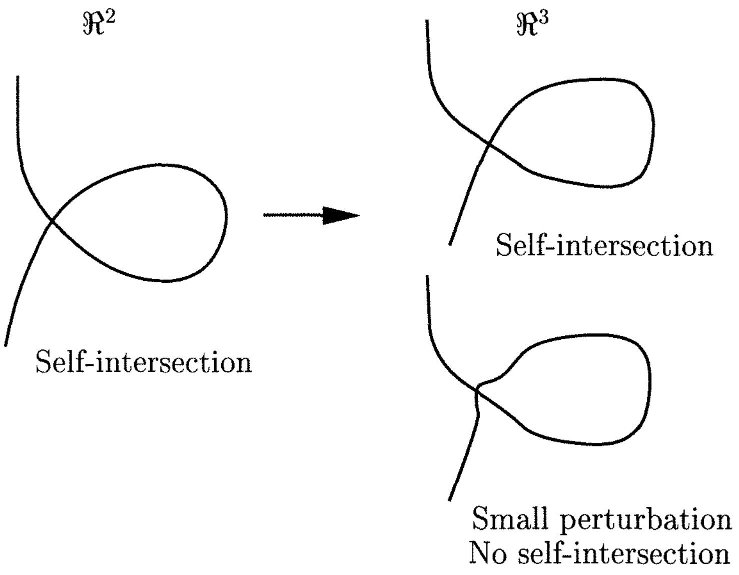


Figure 3.3: Self-intersections for a one-dimensional manifold

itself in a set of dimension $2k - d = 0$. Even when the embedding is perturbed, the self-intersection remains. In $\Re^3$, in general there will be no self-intersections. However, when there is a self-intersection it is removed by a small perturbation.

3.3 The original theorems of Takens

The paper of Takens (1981) is the first work which rigorously justified the use of time-delay coordinates in the analysis of autonomous systems. Specifically, in this work a sufficient condition is given for representing the dynamics of a state-space system in time-delay coordinates. This condition states that, for a state-space system of dimension n, that $l > 2n$ is a sufficient condition for representing the dynamics of the system. In order to see how these results are derived, the work of Takens will be outlined here.

Takens work focuses on dynamical systems with a single observable. The dynamic system is defined as a diffeomorphism ϕ on a manifold M. The diffeomorphism $\phi : M \rightarrow M$, maps M to itself and defines the evolution of the dynamical system. The future position of an initial position $x_0 \in M$ at time t is given by $\phi_t(x_0)$. For discrete-time systems $t \in \mathcal{N}$ and $\phi_i = (\phi)^i$. For continuous systems $t \in \Re$ and $\phi_t(x_0)$

is defined by the integral curve through x_0. If the dynamical system is described by a set of ordinary differential equations, the manifold $M \subset \Re^n$ where n is the number of states in the description.

The single observable is a scalar measurement on the state of the dynamical system. The observable is given by $y : M \rightarrow \Re$. Takens then defines the following problem. If the observable is known as a function of time $(t \rightarrow y(\phi_t(x)))$ for a dynamical system, what can be learned about the original dynamical system? The first clues to answer this question are stated in Theorem 1 of Takens' paper.

Theorem 3.3.1 (Takens (1981) Theorem 1) *Let M be a compact manifold of dimension n. For pairs (ϕ, y), $\phi : M \rightarrow M$ a smooth diffeomorphism and $y : M \rightarrow \Re$ a smooth function, it is a generic property that the map $\Psi_{(\phi,y)}(x) : M \rightarrow \Re^{2n+1}$ defined by*

$$\Psi_{(\phi,y)}(x) = [y(x), y(\phi(x)), \ldots, y(\phi^{2n}(x))] \tag{3.5}$$

is an embedding.

The proof of this theorem relies on the fact the covectors $dy_x, d(y\phi)_x, \ldots, d(y\phi^{2m})_x$ will generically span the tangent bundle $T^*_x(M)$ of the manifold for pairs (ϕ, y) or pairs $(\bar{\phi}, \bar{y})$ which are arbitrarily close to the pair (ϕ, y). If the covectors span $T^*_x(M)$, then the mapping $\Psi_{(\phi,y)}$ is an immersion. This means that the Jacobian of the map $\Psi_{(\phi,y)}(x)$ is full rank for all $x \in M$ and $\Psi_{(\phi,y)}$ is an injective mapping. From this it can be shown, in a method similar to that given by Whitney (1936), that $\Psi_{(\phi,y)}(x) : M \rightarrow \Re^{2n+1}$ (or any generic map from $M \rightarrow \Re^{2n+1}$) is an embedding.

In a similar manner, similar results are found for continuous systems.

Theorem 3.3.2 (Takens (1981) Theorem 2) *Let M be a compact manifold of dimension n. For pairs (X, y), X a smooth vector field and y a smooth function on M, it is a generic property that $\Psi_{(X,y)} : M \rightarrow \Re^{2n+1}$ defined by*

$$\Psi_{(X,y)}(x) = [y(x), y(\phi_1(x)), \ldots, y(\phi_{2n}(x))] \tag{3.6}$$

is an embedding where ϕ_t is the flow of X.

It is also shown in Theorem 3 of Takens (1981) that the map
$\Psi_{(X,y)}(x) = [y(x), \frac{d}{dt}(y(\phi_t(x))), \ldots, \frac{d^{2n}}{dt^{2n}}(y(\phi_t(x)))]$ is an embedding.

Since these mappings are embeddings, this implies that there is a one-to-one relationship which is an immersion between the original state space dynamics and the new coordinates (Guillemin and Pollack, 1974). Since the change of coordinates is smooth and nonlinear, it expected that it should preserve the properties of the original state space dynamics. Since a given initial condition in the state space uniquely determines the state trajectory for the future and the map $\Phi(x)$ defining the time delay coordinates is an embedding, it is expected that one should be able to predict the future outputs of the system in the time-delay coordinates. Mathematically this means that there exists some F such that

$$y(t+1) = F[y(t), y(t-1), \ldots, y(t-2n)]. \tag{3.7}$$

While this result was not presented explicitly, the implication of Takens (1981) was clear to researchers in the area.

3.4 Extending the work of Takens for autonomous systems

The paper of Sauer et al. (1991) extends the results presented by Takens in a number of important ways. The embedding mapping which was shown to be "generic" in the paper of Takens is shown to be found with "probability one", and the manifold M is replaced by what may possibly be a fractal set. Bounds are also put on the dimension of the self-intersecting set when the number of time-delay coordinates is too small. In addition, it is shown that filtered versions of the time-delay coordinates define an embedding of the state space dynamics.

In the proofs of Takens, the generic condition shows that every function which forms the time-delay coordinates is either an embedding or is arbitrarily close to an embedding. It can also be said that the set of functions which are embeddings are "dense" in the function space. While this may seem like a very strong condition, there are examples of sets which which are open and dense but are thin in terms of probability. In Sauer et al. (1991) Arnold tongues are presented as an example of a set which is dense, but one for which the probability of landing on the set is small.

When performing experimental studies, one would like to know that a given measurement function will produce an embedding with probability one. For this reason, a stronger statement of the Whitney embedding theorem is presented.

Theorem 3.4.1 (Whitney embedding prevalence (Sauer et al., 1991)) *Let M be a compact smooth manifold of dimension n contained in $\Re^k$. Almost every smooth map $\Re^k \rightarrow \Re^{2n+1}$ is an embedding of M.*

In addition, the results of the Whitney embedding theorem are extended to sets which are not manifolds. By defining the box-counting dimension of a set $A \in \Re^k$ as

$$\text{boxdim}(A) = \lim_{\epsilon \rightarrow 0} \frac{\log N(\epsilon)}{-\log \epsilon} \tag{3.8}$$

where $N(\epsilon)$ is the number of boxes which intersect the set A for a grid of k-dimensional boxes with size ϵ. Using this definition, a less conservative condition can be found which guarantees that the mapping is an embedding.

Theorem 3.4.2 (Fractal Whitney embedding prevalence (Sauer et al., 1991)) *Let A be a compact subset of $\Re^k$ of box-counting dimension d, and let l be an integer greater than $2d$. For almost every smooth map $F : \Re^k \rightarrow \Re^l$,*

1. *F is one-to-one on A*

2. *F is an immersion on each compact subset C of a smooth manifold contained in A.*

Since any one-to-one immersion is an embedding, this shows that the mapping from A to $\Re^l$ is an embedding for $l > 2d$.

The previous two extensions to Whitney's embedding theorem only consider the case where a number of independent measurements are being made simultaneously on a dynamical system. Of more interest to experimental researcher is the case where time-delay embeddings are utilized, similar to the results of Takens (1981). The specific time-delay embedding considered in this work is

$$\Psi_{(X,y,\tau)}(x) = [y(x), y(\phi_{-\tau}(x)), \ldots, y(\phi_{-(l-1)\tau}(x))] \tag{3.9}$$

where τ is called the "delay." The time-delay embedding theorem is then stated as follows.

Theorem 3.4.3 (Fractal delay embedding prevalence (Sauer et al., 1991)) *Let ϕ be a flow on an open subset U of $\Re^k$, and let A be a compact subset of U of box-counting dimension n. Let $l > 2n$ be an integer, and let $\tau > 0$. Under certain assumptions concerning the equilibria and periodic orbits, then for almost every smooth function y on U the delay coordinate map $\Psi_{(X,y,\tau)} : U \to \Re^n$ is:*

1. *One-to-one on A.*

2. *An immersion on each compact subset C of a smooth manifold contained in A.*

These results extend the original results by embedding compact sets of a known box-counting dimension rather than compact manifolds and by replacing generic with prevalent. A nearly identical theorem is presented for discrete time systems, and as a remark the results are extended to systems with multiple outputs.

Remark 3.4.1 (Sauer et al. (1991)) *The results presented above are easily extended to the more general case where the reconstruction map Ψ consists of a mixture of lagged observations. The more general result states that*

$$\Psi(x) = [y_1(x), \ldots, y_1(\phi^{l_1-1}(x)), \ldots, y_q(x), \ldots, y_q(\phi^{l_q-1}(x))] \tag{3.10}$$

satisfies the conclusion of the above theorem as long as $l_1 + \ldots + l_q > 2n$ and the conditions on periodic points are satisfied.

This result is directly applicable to the inferential prediction problem, in which a primary output variable y^p which cannot be measured is to be predicted as a nonlinear function of a number of secondary output variables y_i^s which can be measured. For a dynamical system defined by

$$\begin{aligned} \dot{x} &= f(x) \\ y^p &= g(x) \\ y_i^s &= h_i(x) \end{aligned} \tag{3.11}$$

$x \in \Re^n$, the mapping

$$\Psi(x) = [h_{i_1}(x), \ldots, h_{i_l}(x)] \tag{3.12}$$

is an embedding of the state for $l > 2n$ for almost every set of smooth f and h_i (the inferential measurement functions). Since the mapping is an embedding, the state is simply a nonlinear transformation of the "secondary coordinates" defined by $[y_{i_1}^s, \ldots, y_{i_l}^s]$. Since the primary variable y^p is a function of the state, then the primary variable can be found as a nonlinear function of the variables in the secondary coordinates or there exists a function G such that

$$y^p(t) = G[y_{i_1}^s(t), \ldots, y_{i_l}^s(t)]. \tag{3.13}$$

Since this result depends only on an embedding between the secondary outputs and the state, $f(x)$ in system (3.11) could be replaced by $f(x, u)$ where u is an external input to the system and the results detailed above will still hold. While these specific results concerning inferential prediction for input/output systems are not found in Sauer et al. (1991), the extension of these results to this specific case is trivial.

In the case $l < 2n$ for the above time-delay mappings, the mapping may not be an embedding. However, most of A will be embedded when A is a smooth manifold and $l > n$. In this case, nearly every mapping Ψ will be an embedding outside of a subset with dimension less than or equal to $2n - l$. A similar result is given for the case where A is a set with a known box-counting dimension. The results follow naturally from the discussion examining how a manifold can have self-intersections globally when the embedding dimension is too small to meet the conditions defined by Whitney's embedding theorem.

It also shown in Sauer et al. (1991) that a map which produces a filtered version of the time-delay coordinates is also an embedding. Using the map $\Psi : \Re^n \rightarrow \Re^w$ defined by

$$\Psi_{(\phi,y)}(x) = [y(x), y(\phi(x)), \ldots, y(\phi^{w-1}(x))], \tag{3.14}$$

the filtered delay coordinate mapping is given by $F(B, \phi, y) : \Re^n \rightarrow \Re^l$ where

$$F_{(B,\phi,y)}(x) = B\Psi_{(\phi,y)}(x) \tag{3.15}$$

and B is an $l \times w$ constant matrix. Note that this constant matrix filter is different than bandpass filters which are also commonly utilized for noise reduction. Under some modest assumptions, $F_{(B,\phi,y)}(x)$ defines an embedding.

Theorem 3.4.4 (Filtered delay embedding prevalence (Sauer et al., 1991)) *Let U be an open subset of $\Re^k$, ϕ be a smooth diffeomorphism on U, and let A be a compact subset of U,* boxdim$(A) = d$. *For a positive integer $l > 2d$, let B be an $l \times w$ matrix of rank l. Assume ϕ has no periodic points of period less than or equal to w. Then for almost every smooth function y, the delay coordinate map $F_{(B,\phi,y)} : U \rightarrow \Re^l$ is:*

1. *One-to-one on A.*

2. *An immersion on each closed subset C of a smooth manifold contained in A.*

Filtering of the time-delay coordinates can help to reduce the effect of noise, so this theorem may be helpful when dealing with data from a physical system.

3.5 Embedding theorems for input/output systems

While the embedding theorems for autonomous systems are interesting for analysis and future prediction of self-driven systems, most engineering applications of interest are systems with inputs as well as outputs. For reasons of control, the availability of an input is crucial for driving the system such that some desired behavior is produced. The systems which will be considered here are of the type

$$\begin{aligned}\dot{x} &= f(x,u) \\ y &= h(x)\end{aligned} \tag{3.16}$$

with $x \in \Re^n$, external input $u \in \Re$, $f : \Re^n \times \Re \rightarrow \Re^n$, and $h : \Re^n \rightarrow \Re$. When modeling the dynamics of this system from discrete measurements on the input and output, it is natural to consider a model of the form

$$y(t) = G[y(t-\tau),\ldots,y(t-l\tau),u(t-\tau),\ldots,u(t-m\tau)] \tag{3.17}$$

where the function G is fitted to the data using standard nonlinear modeling methods as neural networks, radial basis functions, or NARMAX type models. A question similar to that posed by researchers in the field of experimental chaos arises naturally from this analysis. If the equations describing the system dynamics (3.16) are known and the input changes only at the discrete sampling times of the system, how many delayed version of the output l and input m are needed to represent the dynamics of the system?

There have been suggestions for solving this problem for "fading memory" state-space systems (Boyd and Chua, 1985; Sandberg, 1991), however in this case the input/output relationship is only *approximated* as the nonlinear moving average filter

$$\hat{y}(t) = g[u(t-1),u(t-2),\ldots,u(t-m)]. \tag{3.18}$$

By increasing the number of delayed input terms m, the error associated with this approximation can be made arbitrarily small. However, the results of Takens (1981) led researchers to consider a totally different approach to the problem.

3.5.1 Casdagli's initial work

Casdagli (1992) considered this problem following the approach suggested by Takens by representing the input/output dynamics of the system in the form of Equation (3.17). While Casdagli simply considers the existence of an embedding between the state-space of a discrete-time systems and its time-delay coordinates, the extension of these results to the equivalence between the input/output relationship between System (3.16) and (3.17) is trivial for smooth functions f and h. Using this approach, Casdagli claims for $m > 2n$ and $l > 2n$ a globally smooth function G exists which can accurately represent the input/output dynamics for almost all input sequences. Additionally, if $m = l = n + 1$ an almost everywhere smooth function G is able to represent the input/output dynamics.

These claims were only justified using a heuristic argument by Casdagli, no mathematically rigorous treatment was given. Consider the map

$$\begin{aligned}\Psi[x(t-m), u(t-1), \ldots, u(t-m)] &= [\ldots, h(\phi(\phi(x(t-m), u(t-m)), u(t-m+1)), \\ &\qquad h(\phi(x(t-m), u(t-m)), h(x(t-m))] \\ &= [y(t), y(t-1), \ldots, y(t-m)] \qquad (3.19)\end{aligned}$$

where $\phi(x(t), u(t)) = x(t+1)$ is the result of integrating $\dot{x} = f(x, u)$ for one sampling period and u is constant between sampling times. In order to represent the dynamics in the form of Equation (3.17), the map defined by (3.19) will be a diffeomorphism if m is large enough such that $x(t-m)$ can be found from the remaining terms $[u(t-1), \ldots, u(t-m), y(t), \ldots, y(t-m)]$. If this can be found, then there exists a mapping of the form (3.17) since $y(t)$ can be found from $x(t-m)$ and the input terms $[u(t-1), \ldots, u(t-m)]$.

Casdagli (1992) states that to have a unique solution for $x(t-m)$, in general $m > n$. If $m = n$, the solution of the n components of $x(t-m)$ from the terms $[u(t-1), \ldots, u(t-m), y(t), \ldots, y(t-m)]$ is equivalent to solving a set of n simultaneous nonlinear equations. Since a unique n-dimensional solution of n nonlinear equations does not exist in general, $m = n + 1$ is needed to break the degeneracy. Therefore for generic f and h, it is expected that $x(t - m)$ can be found for $m = n + 1$ unless the solution lies on a $n - 1$ "bad" submanifold $\Sigma^{n-1} \subset \Re^n$. On this bad submanifold, the mapping Ψ is not expected to be an embedding due to self-intersection in $\Re^m$. The location of this submanifold of self-intersection will depend on the functions f and h as well as the input sequence u to the system. When the system is on Σ^{n-1}, accurate prediction of $x(t - m)$ is impossible due to the self-intersection. If the dimension of the time-delay vector m is increased by one, the dimension of self-intersection will decrease by one until $m > 2n$ where no self-intersection is expected in general.

The results above are developed in a manner similar to the autonomous case. The only difference is that the self-intersection is dependent on a specific input sequence. One additional problem which arises in the case of input/output dynamics is that even when $m > 2n$ a certain class of inputs could happen to lie on a "bad" subset of the input space which would induce a self-intersection of dimension 0 on mapping (3.19). Casdagli claims that this problem can not be avoided even by increasing the dimension of the embedding m. A self-intersection set of dimension 0 is expected to persist for specific input sequences.

These arguments show that the solution of the equations for $x(t - m)$ is unique. To show that the mapping is also smooth requires the implicit function theorem. If the Jacobian of Ψ is full rank, then the input/output relationship defined by (3.17) should be smooth. The conditions which make $D\Psi$ full rank are equivalent to local observability which will be presented in the next section (Poncet et al., 1995). Since the solutions are unique and the mapping is smooth, Ψ is expected to be an embedding.

3.5.2 Formalizing the input/output embedding theorem

Poncet et al. (1995) set out to formalize the ideas presented in Casdagli (1992) for embeddings of input/output systems. Since the proofs detailed in Poncet et al. (1995) consider discrete-time state-space systems, the proofs will be presented in their original form. However, the extension to continuous-time systems should be relatively straight forward (as it was for autonomous systems).

The starting point is the discrete-time state-space system (f, h)

$$\begin{aligned} x(t) &= f[x(t-1), u(t)] \\ y(t) &= h[x(t)] \end{aligned} \tag{3.20}$$

where once again $x \in \Re^n$, $u \in \Re$, and f and h are polynomial maps. System (3.20) will be referred to as the n-dimensional polynomial system (f, h). The question to be answered is, for an arbitrary polynomial system (f, h) does an input/output relationship

$$y(t) = g[y(t-1), \ldots, y(t-l), u(t), \ldots, u(t-(m-1))] \tag{3.21}$$

exist? The answer according to Poncet et al. (1995) is that there exist $l, m \geq n+1$ and g continuously differentiable on a full measure open subset such that (3.21) holds.

In order to prove this theorem, a pair of preliminary lemmas is needed. However, first some short-hand notation is introduced to simplify the presentation of the proofs.

$$x_0 = x(k) \tag{3.22}$$

$$y_0 = y(k) \quad y_1 = y(k+1) \tag{3.23}$$

$$u_0 = u(k) \quad u_1 = u(k+1) \tag{3.24}$$

where k is an arbitrary time. $f(\cdot, u_i)$ of x is denoted as $f_{u_i}(\cdot)$ and the output functions are denoted as

$$h_i \triangleq \begin{cases} h \circ f_{u_i} \circ \ldots \circ f_{u_1} & i = 1, 2, \ldots \\ h & i = 0 \end{cases} \tag{3.25}$$

For $i = 0, 1, \ldots, i$ the set $\chi_i \subset \Re^d$ is defined as the locus of initial states consistent with the input/output sequence $(y_0, y_1, u_1, \ldots, y_i, u_i)$ associated with a given initial state x_0. Mathematically this is

$$\chi_i(x_0, u_1, \ldots, u_i) \triangleq \{x \in \Re^d \mid h(x) = y_0, h_1(x) = y_1, \ldots, h_i(x) = y_i\}. \tag{3.26}$$

Trivially, x_0 will always be a member of $\chi_i(x_0, u_1, \ldots, u_i)$.

The generalized observability matrix of a given state x_0 and input sequence $(u_1, \ldots, u_i)$ is defined as the $(i+1) \times d$ Jacobian matrix

$$\mathbf{O}_{i+1}(x_0, u_1, \ldots, u_i) \triangleq [\nabla h_0(x_0), \ldots, \nabla h_i(x_0)]^T \tag{3.27}$$

where ∇ is the gradient with respect to x.

The first lemma shows that the initial state can be uniquely determined from an input/output sequence of appropriate length.

Lemma 3.5.1 (Poncet et al. (1995) Lemma 1) *With probability one for any n-dimensional polynomial system* (f, h),

$$\chi_n(x_0, u_1, \ldots, u_n) = \{x_0\} \tag{3.28}$$

holds for an open set of full measure in the joint space of $(x_0, u_1, \ldots, u_n)$.

The proof of this lemma is based on the fact that $\det \mathbf{O}_n$ is a polynomial in the variables $x_0, u_1, \ldots, u_{n-1}$ and since the zeros of a polynomial form a closed set of zero measure, $\mathbf{O}_n$ should be nonsingular with probability one. Since $\mathbf{O}_n$ is nonsingular, according to the Inverse Mapping Theorem there exists a neighborhood of x which contains no other solution. Therefore, only a countable set of isolated points x should satisfy $\chi_{n-1}(x_0, u_1, \ldots, u_{n-1})$. This argument is then extended to show that the additional dimension needed for $\chi_n(x_0, u_1, \ldots, u_n)$ gets rid of the redundant solutions, and the lemma is proved.

A remaining question is what happens for input sequences for which this mapping is expected to have self-intersections? While this case is discussed in Casdagli (1992), it is not mentioned in Poncet et al. (1995). While the definition of "with probability one" may account for the difference in the results, this discrepancy is still a bit puzzling.

The second lemma builds on the first lemma to show that an embedding exists between the state space and time delay coordinates.

Lemma 3.5.2 (Poncet et al. (1995) Lemma 2) *Let (f, h) be a polynomial system. Then with probability one, the map*

$$\Psi : (x_0, u_1, \ldots, u_n) \to (y_0, \ldots, y_n, u_1, \ldots, u_n) \tag{3.29}$$

is an embedding on the full measure open subset in the joint space of $(x_0, u_1, \ldots, u_n)$.

The mapping is shown to be one-to-one using Lemma 3.5.1. Then the mapping Ψ is shown to be an immersion by analyzing the Jacobian of $\Psi_\star(x_0, u_1, \ldots, u_n) \triangleq (y_0, \ldots, y_{n-1}, u_1, \ldots, u_n)$ which is

$$D\Psi_\star = \frac{\partial \Psi}{\partial(x_0, u_1, \ldots, u_n)} = \begin{bmatrix} \mathbf{O}_n & \mathbf{L} \\ \mathbf{Z} & \mathbf{I}_n \end{bmatrix} \tag{3.30}$$

where $\mathbf{O}_n$ is defined in (3.27), $\mathbf{L}$ is a lower triangular matrix, $\mathbf{Z}$ is a matrix of zeros, and $\mathbf{I}_n$ is an identity matrix. It can easily be shown that $\det(D\Psi_\star) = \det(\mathbf{O}_n)$, so if the system is observable then $D\Psi_*$ should be full rank. This means that the Jacobian of $D\Psi$ is also full rank and mapping (3.29) is an immersion. Ψ is then an embedding since it is one-to-one, an immersion, and proper (due to the polynomial form of f, h).

Since $\Psi(x_0, u_1, \ldots, u_n)$ is an embedding, the final result can easily be shown.

Theorem 3.5.1 (Poncet et al. (1995) Proposition 3) *With probability one for any n-dimensional polynomial system (f, h), there exist two positive integers $l, m \geq n + 1$ and a function g continuously differentiable on a full measure open subset such that*

$$y(k) = g[y(k-1), \ldots, y(k-l), u(k), \ldots, u(k-(m-1))] \tag{3.31}$$

with l, m, g independent of k.

In the proof of this theorem, the case $l = m = n + 1$ is considered. Since Lemma 3.5.2 shows that mapping Ψ is an embedding, Ψ^{-1} exists and is smooth. Because of this, the map

$$g : (y_0, \ldots, y_n, u_1, \ldots, u_{n+1}) \rightarrow h \circ f_{u_{n+1}} \circ \ldots \circ f_{u_1} \circ P \circ \Psi^{-1}(y_0, \ldots, y_n, u_1, \ldots, u_{n+1}) \tag{3.32}$$

exists where $\Psi^{-1}(y_0, \ldots, y_n, u_1, \ldots, u_{n+1}) = (x_0, u_1, \ldots, u_{n+1})$ and P is a projection operator such that $P \circ \Psi^{-1} = x_0$. Upon further examination, it is clear that $g(y_0, \ldots, y_n, u_1, \ldots, u_{n+1}) = h_{n+1}(x_0) = y_{n+1}$ which is exactly the desired result.

3.6 Conclusions

A review of the work in the field of "time-delay" embeddings has been presented. It has been shown that a nonlinear time-delay representation exists for generic nonlinear state-space systems. Specifically, a sufficient condition is given for the time-delay representation to exist. The mathematical background for the methods used in these works is introduced and some time-delay embedding theorems are presented for autonomous and input/output systems.

Chapter 4 The false nearest neighbors algorithm and noise corruption

4.1 Motivation

While initial work with the FNN algorithm was very promising, remarkably little work has been completed in two specific areas. First, the choice of the threshold R in the original FNN algorithm was based only on heuristic arguments. Here, theoretical analysis is presented in order to justify the choice of the threshold used by Abarbanel et al. (1993) and to analyze what can happen in special limiting cases.

The second area not completely analyzed by Abarbanel et al. (1993) is the set of problems which can arise when analyzing noise-corrupted data with the FNN algorithm. In the original work, Abarbanel et al. (1993) simply showed that the results of the FNN algorithm degrade gracefully as noise is added to the time-series data under consideration. As will be shown later in this chapter, the FNN algorithm can incorrectly recommend an embedding dimension which is too large when analyzing noise-corrupted data. Even worse, the cause of the error becomes more pronounced when more data are presented to the FNN algorithm for analysis. The source of this problem is easy to isolate using analysis similar to that used to investigate the choice of the threshold R. Extending this analysis to noise-corrupted data leads naturally to a new threshold test which can deal with noise-corrupted time-series in the proper manner. Since data from physical processes are always corrupted by noise, it is important to deal with this problem correctly.

4.2 The false nearest neighbors algorithm (FNN)

Identification consists of two distinct steps: determining the past terms which are needed for predicting future outputs and determining the function relating those past terms to the future terms. While many methods for determining the functional relationship have been published recently, the majority of these methods do not deal with the problem of determining the proper regressor vector. Often it is simply "assumed" that the dynamics can be represented using a particular regression vector and then the functional relationship between the past and future is approximated.

The false nearest neighbors (FNN) algorithm was originally developed for determining the smallest dimension regression vector needed to recreate the dynamics of autonomous chaotic systems (Kennel, Brown and Abarbanel, 1992). The idea behind the FNN algorithm is geometric in nature. If there is enough information in the regression vector to predict the future output, then any two regression vectors which are close in the regressor space will also have future outputs which are close in some sense. If there is not enough information from the past in the regressor vector to recreate the dynamics of the system, then there will be some neighborhoods in the regressor space with very different future outputs. These trajectories which are close in the regression space and have vastly different outputs can be thought of as *false neighbors*, since they are close in the regression space only because of projection onto a space with a dimension too small to represent the dynamics of the system. For noise-free data, there will no longer be any false neighbors when the dimension of the regression vector is large enough to allow accurate prediction of future outputs.

In order to determine whether neighbors are true or false, a test must be defined to determine whether the neighbors have future outputs which are "far apart". In the original FNN algorithm, a ratio test determines whether the distance between future outputs is significantly larger than the distance between time-delay regression vectors which are close in the regressor space. If the distance between future outputs is "large" when divided by the distance between the two regressors which are "nearest neighbors" in the regression space, then the neighbors are considered to be false.

Here is the original FNN algorithm which was defined for autonomous systems.

1. Form the set of regressors

$$\psi_l(t) = [y(t-\tau), \ldots, y(t-l\tau)] \tag{4.1}$$

   and related outputs $y(t)$ from time-series data.

2. Identify the closest regressor (in the Euclidean sense) to a given vector in the regression space. That is, for a given regressor $\psi_l(k)$ find another regressor $\psi_l(j)$ in the data set such that the following distance d is minimized.

$$d = ||\psi_l(k) - \psi_l(j)||_2 \tag{4.2}$$

   It should be noted that times k and j themselves do not need to be close to one another. In fact, if k and j are always close to one another the sampling time τ may be too small and there may be problems in accurately estimating the dimension of the regression vector. (Fredkin and Rice, 1995).

3. Determine if the following expression is true or false

$$\frac{\mid y(k) - y(j) \mid}{||\psi_l(k) - \psi_l(j)||_2} \leq R \tag{4.3}$$

   where R is a previously specified threshold value. If expression 4.3 is true, then the neighbor are recorded as *true* neighbors. If the expression is false, then the neighbors are *false* neighbors.

4. Continue the algorithm for all times k in the data set. Calculate the percentage of points in the data set which have false nearest neighbors.

5. Continue the algorithm for increasing dimension l until the percentage of false nearest neighbors drops to zero (or some acceptably small number). If the percentage of false neighbors is large, the FNN algorithm suggests that the

regressor vector must be extended to include more terms in order to accurately capture the dynamics of the system.

While this single threshold test works quite well in cases where there are "sufficient data" to fill out the embedding space, when the distance between nearest neighbors $||\psi_l(k) - \psi_l(j)||_2$ is large the distance between outputs $| \; y(k) - y(j) \; |$ can be large and still satisfy the above threshold test (Equation 4.3). If the distance between the nearest neighbors embedded in the space of outputs and regressors is roughly the same magnitude as the size of the attractor, then the neighbors should also be considered false neighbors. For this reason, a second threshold test which only becomes important in cases of sparse data is also utilized in the original FNN algorithm (Abarbanel et al., 1993).

The second threshold test is defined as

$$\frac{R_{l+1}}{R_A} < A_{tol} \tag{4.4}$$

where

$$R^2_{l+1} = (y(k) - y(j))^2 + ||\psi_l(k) - \psi_l(j)||^2_2 \tag{4.5}$$

$$R^2_A = \frac{1}{N}\sum_{n=1}^{N}[y(n) - \bar{y}]^2 \tag{4.6}$$

$$\bar{y} = \frac{1}{N}\sum_{n=1}^{N} y(n). \tag{4.7}$$

The recommended threshold of $A_{tol} = 2$ is used in all of the examples given here (Abarbanel et al., 1993). Failing this additional threshold test means that the nearest neighbors are far apart in the extended space of R_{d+1} and that the neighbors should be considered false. Since a failure of the above threshold test implies a failure of the threshold test given in Equation (4.3) when nearest neighbors are close (when $||\psi_l(k) - \psi_l(j)||^2_2$ is small), this test is only important when the nearest neighbors are relatively far apart. While the important portion of this test is the relative size of the distance between the outputs $(y(k) - y(j))^2$ for nearest neighbors and a simpler test could accomplish the same results, the test is left in the form motivated by (Abarbanel et al., 1993).

The FNN algorithm has been used successfully as a tool for modeling many autonomous time-series. Abarbanel (1996) is a good reference to the work of the Institute of Nonlinear Science (INLS) at UC San Diego in the area of "nonlinear signal processing" and modeling of chaotic time-series. The researchers at INLS are working on a number problems associated with chaotic time series including sampling time determination, model order determination, model building, noise reduction, control of chaos, and synchronization of chaotic systems.

Abarbanel (1996) applies the FNN algorithm and other tools to computer generated time series as well as time series recorded from physical systems. Some examples include the Lorenz attractor, the Henon attractor, the dynamics of a metal cutting tool, electrical circuits, the dynamics of a laser, the level of the Great Salt Lake, and fluid turbulence experiments. In these examples the researchers are able to do a relatively good job of future prediction of short-term dynamics (long term prediction is impossible for chaotic systems) using FNN and local modeling methods.

4.3 Theoretical choice of FNN threshold and data requirements

Assume that the minimal representation in the time-delay coordinates is known. Let d be the smallest integer for which there exists a function G uniquely determining the output coordinate $y(t)$ for all time-delay vectors

$$\begin{aligned} y(t) &= G[y(t-\tau), y(t-2\tau), \ldots, y(t-d\tau)] \qquad (4.8) \\ &= G[\psi_d(t)]. \qquad (4.9) \end{aligned}$$

Also assume that the function G is known. How should the threshold R for the ratio test be chosen when the time-series data are noise free? Assuming the data are sufficiently "dense" over the region of interest, the following choice of threshold should be made.

Lemma 4.3.1 *$R = \max_t \|DG(\psi_d(t))\|_2$, where $DG(\mathbf{x})$ is the Jacobian of the function G at the point $\mathbf{x}$, is the smallest choice of the threshold which will give 0 % FNN at the proper dimension d for all data sets.*

Proof: If sufficient data are available, the nearest neighbor to each point will be in a region where a local linear approximation to the function G can be made. Using a linear approximation around the point $\psi_d(k)$, the output of the nearest neighbor $y(j)$ is given by

$$y(k) - y(j) = DG(\psi_d(k))[\psi_d(k) - \psi_d(j)] + \mathcal{O}([\psi_d(k) - \psi_d(j)]^2). \tag{4.10}$$

Ignoring the higher order terms, by Cauchy-Schwarz we know

$$\begin{aligned} \mid y(k) - y(j) \mid &\leq \|DG(\psi_d(k))\|_2 \|\psi_d(k) - \psi_d(j)\|_2 && (4.11)\\ \frac{\mid y(k) - y(j) \mid}{\|\psi_d(k) - \psi_d(j)\|_2} &\leq \|DG(\psi_d(k))\|_2 && (4.12)\\ \frac{\mid y(k) - y(j) \mid}{\|\psi_d(k) - \psi_d(j)\|_2} &\leq \max_t \|DG(\psi_d(t))\|_2 \quad \forall\ k \text{ and nearest neighbor } j. && (4.13) \end{aligned}$$

For any choice of R smaller than $\max_t \|DG(\psi_d(t))\|_2$, the gain of the system may cause the FNN algorithm to record a false nearest neighbor (see Equation (4.3)) at the time delay point $\psi_d(t)$ if the nearest neighbor happens to make the equation $\alpha DG(\psi_d(t)) = [\psi_d(t) - \psi_d(j)]$ true for some $\alpha \in \Re$. In other words, the equality of Equation (4.13) will hold when the nearest neighbor to $\psi_d(t)$ happens to lie in the direction of maximum gain of the Jacobian DG.

Therefore, the lower bound on the choice of the threshold R for the FNN algorithm is dependent on knowledge of the function which needs to be identified. However, when an infinite amount of data is available the threshold R can be chosen arbitrarily large.

Lemma 4.3.2 *For an amount of data approaching infinity, any finite threshold value R will lead to a nonzero percentage of FNN when the embedding dimension is smaller than d.*

Proof: The function G has a unique output for all inputs only for values of embedding dimension $l \geq d$. For $l < d$, the implied function relating the time-delay coordinates to the output may have multiple outputs for a given regressor vector. Take a vector $\psi_l(k)$ which does not have a unique output and a sequence of points $\psi_l(j_i)$ where $lim_{i \to \infty} \psi_l(j_i) = \psi_l(k)$ but $lim_{i \to \infty} y(j_i) - y(k) = \kappa \neq 0$. Now using the FNN threshold test (Equation (4.3)) it is seen that

$$\lim_{i \to \infty} \frac{\mid y(k) - y(j_i) \mid}{\|\psi_l(k) - \psi_l(j_i)\|_2} = \infty \tag{4.14}$$

or that the threshold R must be infinitely large for the threshold inequality (Equation (4.3)) to be true.

When applying the FNN algorithm to finite time-series where the function G is unknown *a priori*, the results of these lemmas do not appear to be helpful. However further analysis of the results of Lemma 4.3.2 leads us to believe that when there are more points in a data set, a larger threshold value can be safely used. On the other hand if the threshold is chosen too large and nearest neighbors are not close in the time-delay space, outputs may be in completely different regions of the attractor and still be considered true neighbors by the ratio test. For this reason the choice of threshold is most important in the analysis of time-series of relatively small size. This problem should not be encountered when large data sets, where the space is "filled out" with data, are analyzed.

Another fact to remember when choosing the threshold R is that points which are false neighbor tend to move very far apart in the output space. This is a consequence of the previously mentioned fact that false neighbor are only close because of projection. It has been our experience (as well as that of Abarbanel et al. (1993)), that the percentage of false nearest neighbors tends to be relatively unchanged for a fairly wide range of R ($10 \leq R \leq 30$) for autonomous time-series. Once R is large enough to account for the local gain of the system, the false neighbors move far enough apart to cause the ratio test to fail for even these large values of R. For input/output time-series, the suggested range of R is $10 \leq R \leq 15$. Larger values are needed for analysis of autonomous, chaotic time-series because of the existence of positive Lyapunov exponents.

The results of Lemma 4.3.1 provide some guidance for the analysis of finite data sets where the function G is unknown. For example, the choice of the threshold may have to be adjusted if the function G is expected to have a large gain. However, for finite data it has been observed that false neighbors tend to have outputs which are "very far" apart so a large threshold should still lead to false neighbors for regressors which don't contain enough delayed terms. While R cannot be arbitrarily large when working with finite amounts of data, in practice fairly large values of R (compared to the gain of G) can be used.

When analyzing a time-series with the FNN algorithm, the amount of data needed to carry out an accurate analysis also should to be considered. For the ratio test in the algorithm to correctly interpret the idea of closeness in the output space, the nearest neighbors must be relatively close in the regression space. If distance between nearest neighbors is large, then the difference between their outputs can be very large and the ratio test could still call the two data points true neighbors when the outputs are totally uncorrelated. For all neighbors to be close, the time-series data should "fill out" the regression space to be analyzed. To fill an ∞-norm unit ball of n-dimension with data such that all individual data points are within an δ-sized ball of one another, approximately δ^{-n} points are needed. As the number of dimensions is increased, the amount of data needed to cover the space increases exponentially. This is a well-known problem in nonlinear identification known as the *curse of dimensionality* (Weigend and Gershenfeld, 1994).

The curse of dimensionality has the following implication for the FNN algorithm. In order to accurately assess whether regression vectors of up to dimension n are able to accurately predict future outputs, a time-series of length approximately 10^n is needed. It should be noted that for certain systems, such as autonomous chaotic attractors, the dynamics of the system (and therefore the regression vectors) can lie on a submanifold of the overall regression space and therefore the space to be filled may be of smaller dimension than the entire regression space (see (Abarbanel et al., 1993) for examples). For some input/output systems examined in this thesis, it is possible that significant correlations may exist among the terms contained in the regression vector. However, locally the data can still lie on a manifold with the same dimension as the dimension of the regression vector. To fill out this manifold, again the amount of data should increase exponentially with the dimension of the regression vector.

4.4 The effect of noise

Now consider the case where the data to be examined are corrupted with noise. Assume that for each time t, what is observed is $y^m(t) = y(t) + \delta$ where δ is magnitude bounded ($\mid \delta \mid \leq \delta_\infty$). The time-delay coordinates will also be corrupted with noise, $\psi_l^m(k) = [y^m(k-\tau), \ldots, y^m(k-l\tau)] = [y(k-\tau)+\delta_1, \ldots, y(k-l\tau)+\delta_l]$

$$\begin{aligned}
\mid y^m(k) - y^m(j) \mid &\leq \mid G(\psi_l(k)) - G(\psi_l(j)) \mid + 2\delta_\infty && (4.15)\\
&\leq \|DG(\psi_l(k))\|_2 \|\psi_l(k) - \psi_l(j)\|_2 + 2\delta_\infty && (4.16)\\
&\leq \|DG(\psi_l(k))\|_2 (\|\psi_l^m(k) - \psi_l^m(j)\|_2 + 2\sqrt{l}\delta_\infty) + 2\delta_\infty && (4.17)\\
&\leq \max_t \|DG(\psi_l(t))\|_2 (\|\psi_l^m(k) - \psi_l^m(j)\|_2 + 2\sqrt{l}\delta_\infty) + \\
&\quad 2\delta_\infty && (4.18)
\end{aligned}$$

Converting the final result into a form similar to the threshold function in (4.3).

$$\frac{\mid y^m(k) - y^m(j) \mid}{\|\psi_l^m(k) - \psi_l^m(j)\|_2} \leq \max_t \|DG(\psi_l(t))\|_2 + \frac{2\delta_\infty(\sqrt{l}\max_t \|DG(\psi_l(t))\|_2 + 1)}{\|\psi_l^m(k) - \psi_l^m(j)\|_2}. \quad (4.19)$$

Note that there are two terms on the right-hand side of the above inequality. The first term accounts for the maximum possible local gain of the system at $\psi_l(k)$, and the second term is due only to the effects of noise. Also note that the second term is inversely proportional to the separation of the nearest neighbors in the time-delay coordinates.

If a ratio test with a threshold independent of the density of the points is used to analyze a time-series which contains noise (such as the original FNN test given in Equation (4.3)), the percentage of false nearest neighbors arising only from noise should increase proportionally with the density of the points in the regression space.

This effect can be seen by using a time-series from a well-studied example. Here the FNN algorithm will be used to examine data from integration of the Lorenz equations.

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= -xz + \rho x - y \\ \dot{z} &= xy - \beta z \end{aligned} \tag{4.20}$$

$$\sigma = 10 \quad \rho = 45.92 \quad \beta = 4.$$

A 50,000 point scalar time-series was found by taking the x output from the integration of the above equations using a sampling time of 0.1. Two noisy data sets were also developed by adding uniformly distributed noise of maximum absolute value 0.5 and 1.0 respectively to the original time-series (the x variable of the Lorenz signal has a standard deviation of 12.36).

The FNN algorithm was utilized on each of the 3 data sets using time series of 3 different lengths (500, 5000, 50000 points). The standard choice of threshold recommended in (Abarbanel et al., 1993) ($R = 17.3$) was used. The results of the algorithm can be seen in Figures 4.1-4.3. When the percentage of FNN drops to 0, the embedding dimension is large enough to represent the dynamics.

For noise-free data, the percentage of false nearest neighbors for a given dimension remains nearly constant as the length of the time-series increases. However for data corrupted with noise, the percentage of false nearest neighbors for a given embedding dimension *increases* as the amount of data is increased. For the FNN algorithm working on noise corrupted data, more data are not necessarily better. This is contrary to the common belief in identification that more data lead to more accurate results.

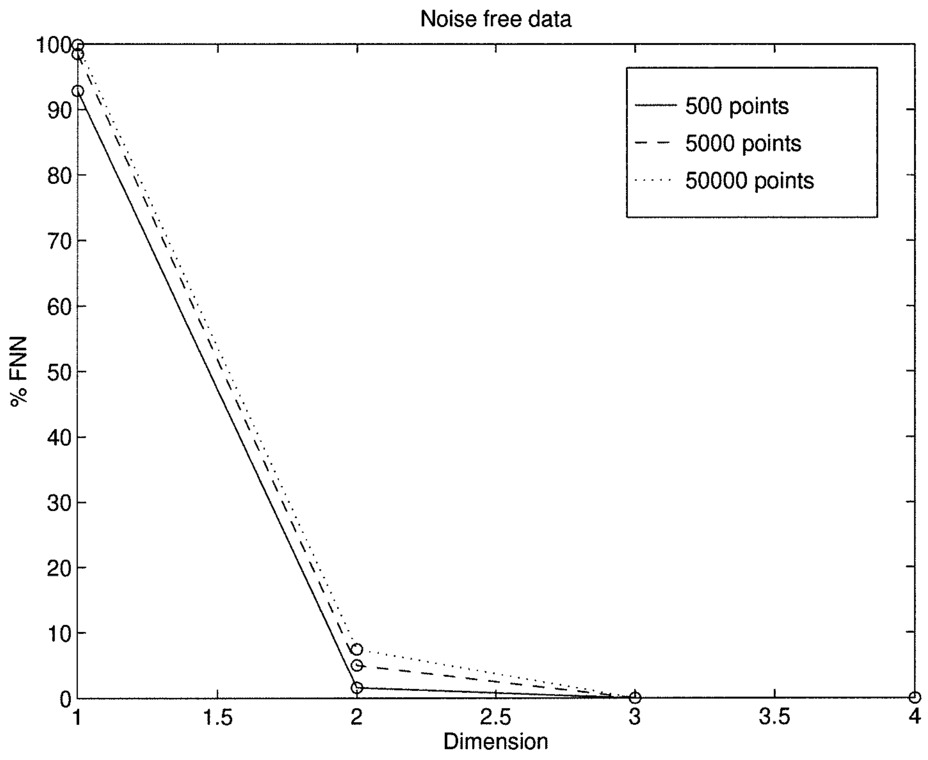


Figure 4.1: FNN plots for noise free Lorenz time-series

Larger time-series lead to "false" false nearest neighbors, neighbors which are a result of noise corruption rather than an incorrect embedding dimension.

4.5 One possible solution

A possible solution to this problem is to account for noise by using a FNN threshold which includes both a constant term (as in the original FNN formulation) and another term to account for noise.

Instead of using Equation (4.3) for the threshold, a logical test for nearest neighbors based on the previous analysis is:

$$\frac{\mid y(k) - y(j) \mid}{\|\psi_l(k) - \psi_l(j)\|_2} \leq R + \frac{2\epsilon R\sqrt{l} + 2\epsilon}{\|\psi_l(k) - \psi_l(j)\|_2}. \tag{4.21}$$

By examining Equation (4.21) further, it can be seen that the threshold test has two distinct limits depending on the size of the term $\|\psi_l(k) - \psi_l(j)\|_2$. When the

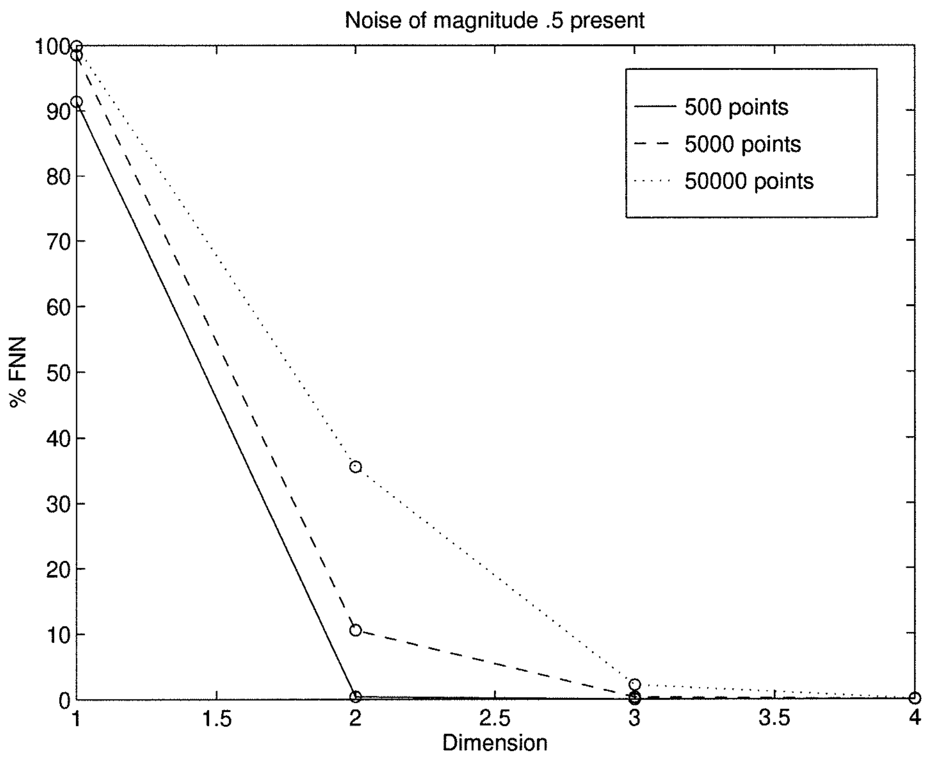


Figure 4.2: FNN plots for Lorenz time-series with magnitude 0.5 noise

distance between the nearest neighbors is relatively large, the first term on the right hand side of the inequality will dominate and this ratio test is equivalent to the original FNN test. When the distance between nearest neighbors approaches zero, the second term on the right hand side dominates. In this limit, the new test is equivalent to asking whether the two observed outputs lie within a certain noise threshold of each other for identical inputs.

The main problem with this choice of threshold is that there are now 2 variables which must be tuned, namely R and ϵ. To be sure that no false nearest neighbors are only the result of noise, one should choose $\epsilon = \delta_\infty$ and R as suggested before. A major problem with this choice of ϵ is that the new threshold test is conservative. In fact, the test may predict a dimension too low with this choice of ϵ. For this reason, a choice of ϵ is around $1/10^{\text{th}}$ the size of the standard deviation of the noise is recommended since this choice provides a good tradeoff between discarding false neighbors which result from noise and not incorrectly counting neighbors in too small of a dimension

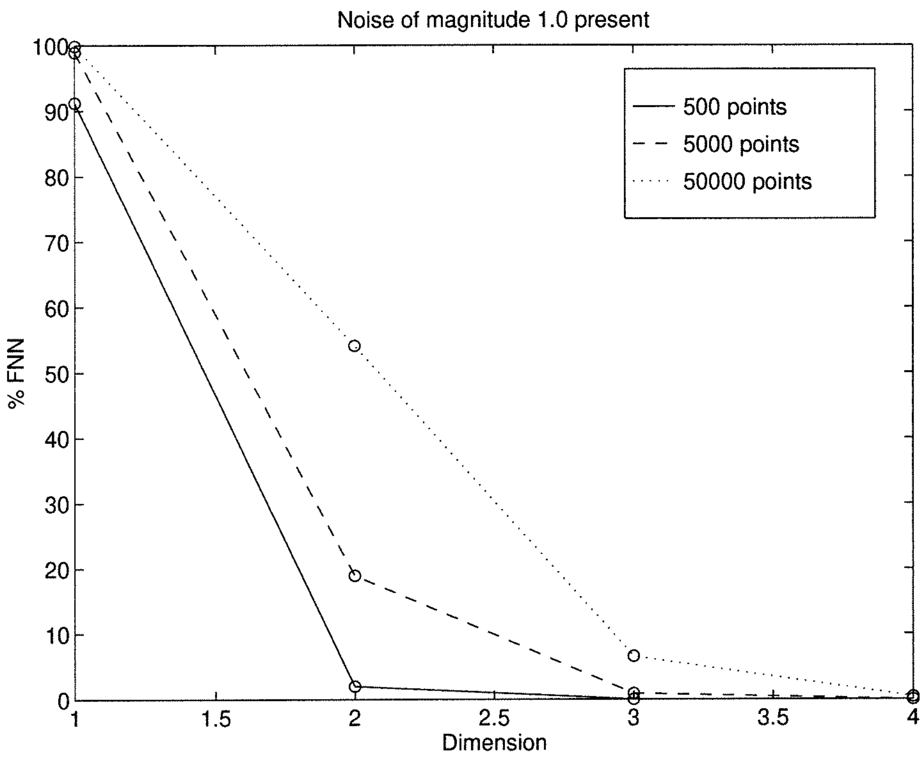


Figure 4.3: FNN plots for Lorenz time-series with magnitude 1.0 noise

as true neighbors. For time-series where δ_∞ is unknown, some physical arguments based on the size of expected measurement noise can be made for determining ϵ. We have observed that the results of the algorithm are qualitatively unchanged by the choice of R and ϵ over a fairly large range. Using this modified test for false nearest neighbors, the problem of an increasing number of false nearest neighbors with increasing amounts of data will not be encountered in noisy data sets.

As can be seen in the following example, utilizing the new ratio test with even a small ϵ term can dramatically affect the results of the FNN algorithm. The same noise corrupted data were examined with the FNN algorithm substituting Equation (4.21) for the standard threshold inequality Equation (4.3). The thresholds were selected such that $R = 17.3$ (as before) and $\epsilon = 0.05$. Figures 4.5 and 4.6 show the results of the modified algorithm on the noisy data. These figures can be compared to Figures 4.3 and 4.4 respectively, which are the results of the original FNN algorithm.

Notice that ϵ is well below the noise magnitude values of δ_∞ (0.5 and 1.0) for

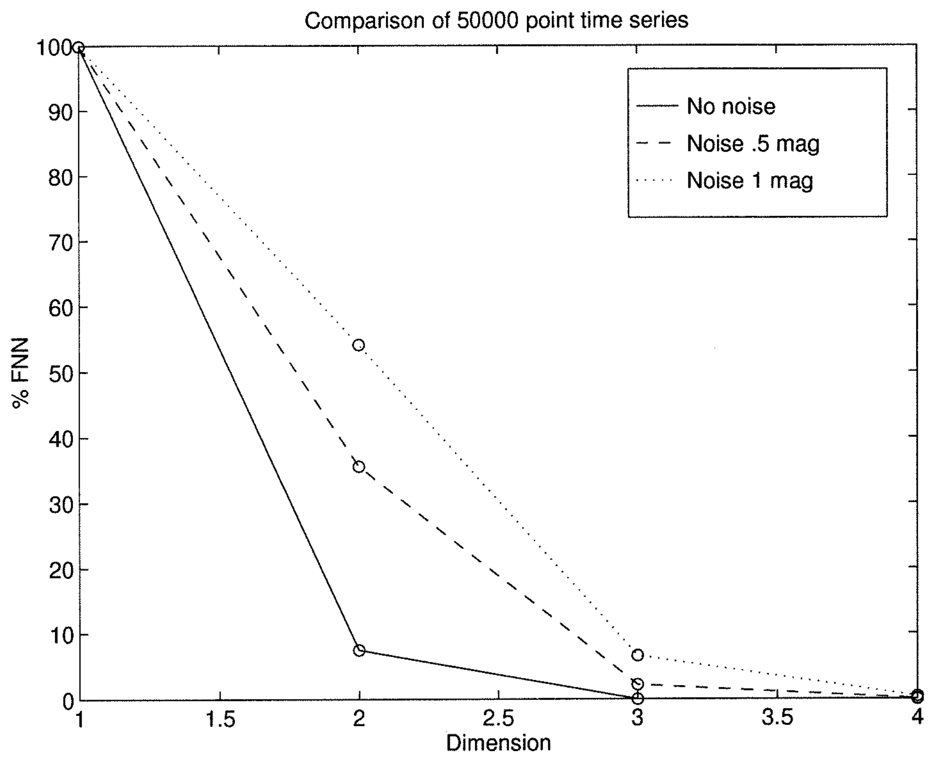


Figure 4.4: FNN plots for the 3 different Lorenz time-series of length 50,000

the two data sets. However, the modified algorithm has the desired results of distinguishing those false nearest neighbors which are the result of noise from those which are the result of an incorrect embedding dimension. The reason that this value of ϵ works well may be both because a conservative choice of R is made and the new FNN threshold is conservative in construction. False nearest neighbors which are the result of noise in the original algorithm are not counted as false nearest neighbors in this modified formulation.

For the data set of size 500, the modified algorithm incorrectly predicts an embedding of dimension 2 for the noisy data set. However, for small data sets problems arising from noise are not encountered when using the original FNN ratio test. For larger data sets, the problem with increasing data causing false nearest neighbors from noise is no longer present when using the new ratio test and correct prediction of the embedding dimension is again possible.

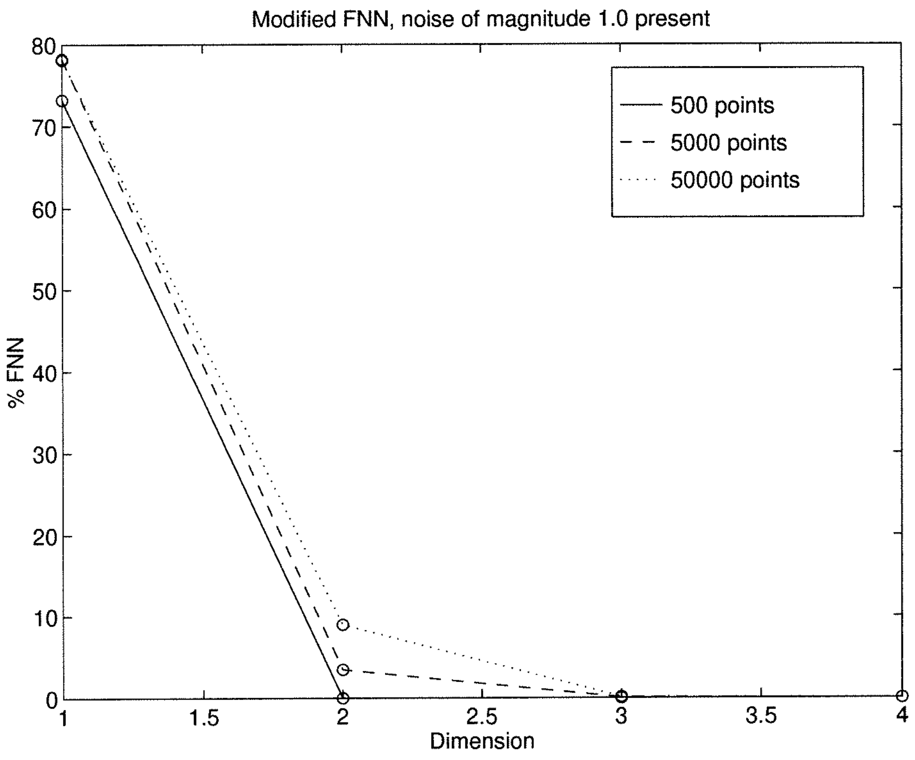


Figure 4.5: FNN plots for Lorenz time-series with magnitude 1.0 noise, modified algorithm

4.6 Additional examples

In this section, two additional examples are presented to illustrate the advantages of the proposed threshold test. The first example consists of data from the Mackey-Glass delay-differential equation, and the second example examines white noise time-series data. The results given by the new threshold test are compared with the results of the standard FNN ratio test.

4.6.1 Mackey-Glass delay-differential equation

The Mackey-Glass equation is the delay-differential equation given below.

$$\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t-\Delta)}{1+x(t-\Delta)^{10}} \tag{4.22}$$

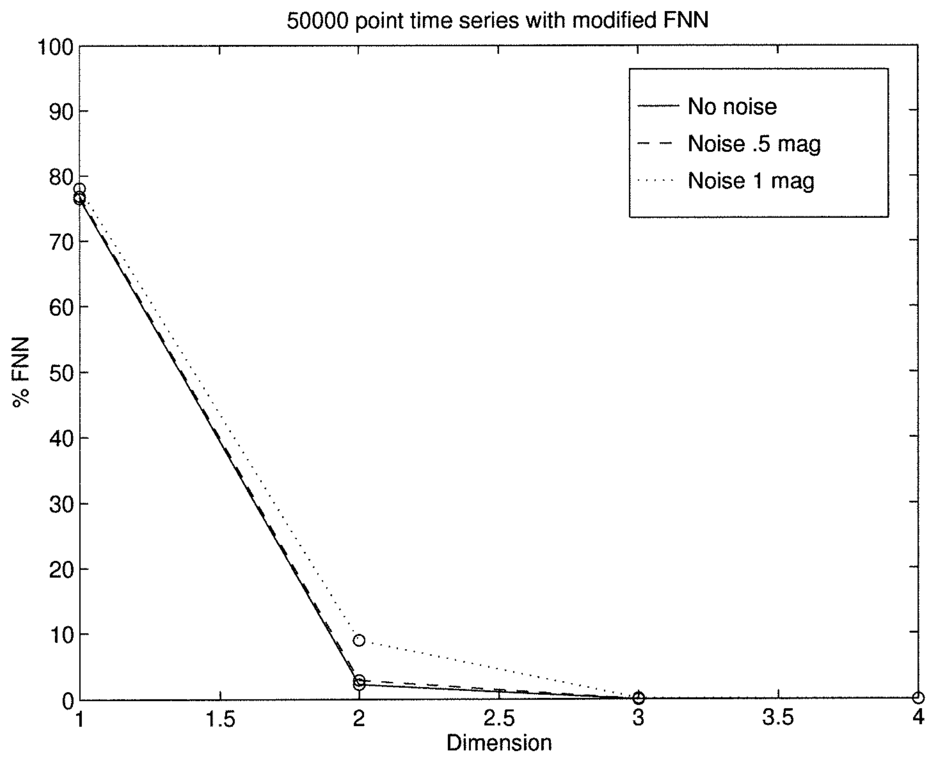


Figure 4.6: FNN plots for modified algorithm (Lorenz time-series, length 50,000)

This delay-differential equation exhibits chaotic behavior over a wide range of delay parameters Δ. For this study, a time-series with 50,000 points was created by integrating Equation (4.22) with $\Delta = 17$. The sampling time used to create the discrete time-series is 6, as is suggested by a previous study of Casdagli (1989).

In addition to the noise free time-series, a noise corrupted time-series is created by adding normally distributed noise with a standard deviation of 0.03 to the original time series. These two time series are then analyzed by the original and modified FNN algorithms, and the results are presented in Figure 4.7 and Table 4.1. For both the original and modified FNN algorithm the threshold $R = 17.3$ is used and in the modified FNN algorithm the threshold $\epsilon = 0.001$ is used. The results of the original FNN algorithm seem to suggest that the proper embedding dimension of the noise free time-series is 4. When the noise corrupted time series is examined by the original FNN algorithm the proper embedding dimension appears to be 5. Again, by applying the modified threshold test, the proper embedding dimension (4) of the noise corrupted time-series can be recovered assuming one considers 0.05 percent false nearest neighbors is close enough to zero for the purpose of determining the embedding dimension.

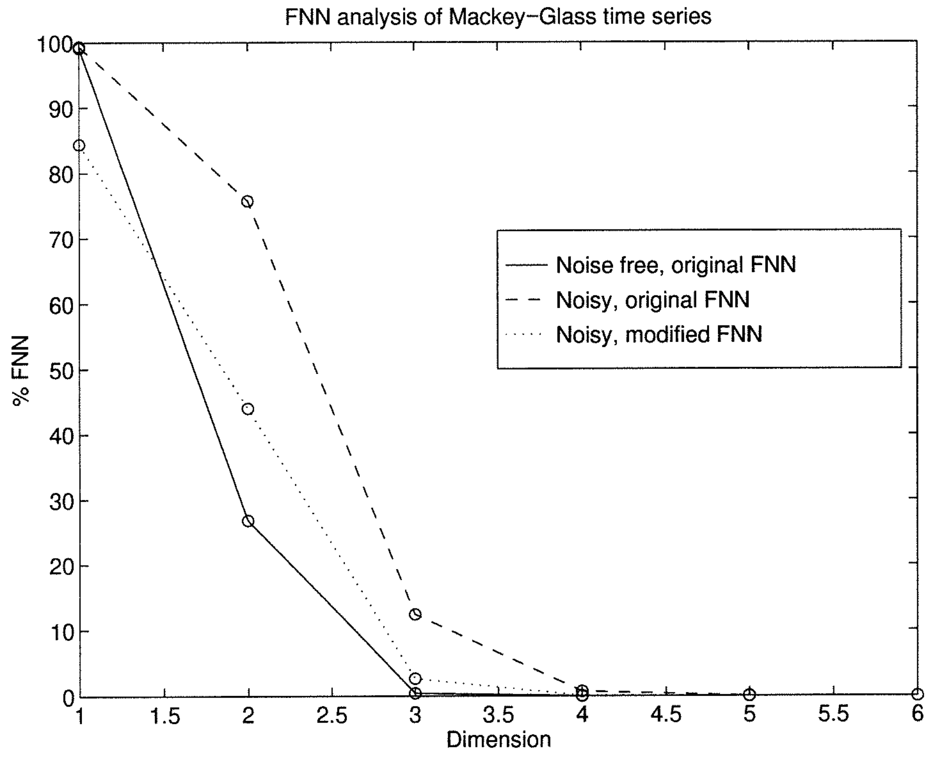


Figure 4.7: FNN plots for Mackey-Glass time-series

% FNN	Dimension					
	1	2	3	4	5	6
Noise-free data, original FNN	99.1178	26.8114	0.42	0	0	0
Noisy data, original FNN	99.37	75.70	12.37	0.71	0.03	0.01
Noisy data, modified FNN	84.36	44.01	2.65	0.05	0	0

Table 4.1: FNN analysis of data from Mackey-Glass equation

What is especially important is that the modified FNN algorithm finds nearly the same percentage of false nearest neighbors for those embedding dimensions near the proper embedding dimension. This is important because accurate prediction of the percentage of false nearest neighbors is crucial when the percentage of false neighbors is small.

4.6.2 White noise

In order to confirm that the new FNN threshold test presented here does not give spurious results, both the original and new FNN threshold tests are applied to a time-series consisting of white noise. A normally distributed white noise time-series of length 50,000 was generated by the MATLAB command `randn` (MathWorks, Inc., 1992). The variance of the time-series is one and the mean is zero.

Again, both the original FNN algorithm and the modified threshold test were applied to the time-series with the threshold $R = 17.3$. For the modified threshold test, thresholds $\epsilon = 0.001$ and $\epsilon = 0.005$ were used. The results of the FNN analysis are given in Figure 4.8. It appears that qualitatively the results are identical for the two different FNN threshold tests. More importantly, none of the tests predict that the time series is deterministic. However, for dimensions larger than 3 a large number of the false nearest neighbors come as a result of the R_{l+1} threshold test (Equation (4.4)). To be sure the results of the original threshold test (Equation (4.3)) are not affected by the threshold modification, a second study was conducted excluding the R_{l+1} threshold test (The results of the previous examples (Lorenz, Mackey-Glass) are not affected by excluding the R_{l+1} threshold test (Equation (4.4)).).

When the R_{l+1} test is excluded (Figure 4.9), the percentage of false nearest neighbors for embedding dimensions 4 and larger are significantly different than those given in Figure 4.8. However, the results of FNN with the original (Equation (4.3)) and modified (Equation (4.21)) threshold tests are nearly identical. The modified FNN threshold test does not significantly change the percentage of false neighbors given by the original FNN algorithm.

4.7 Conclusions

The problem of analyzing noisy time-series with the FNN algorithm has been discussed and illustrated using data from the Lorenz attractor. The problem of false nearest neighbors which arise from noise rather than incorrect embedding dimension

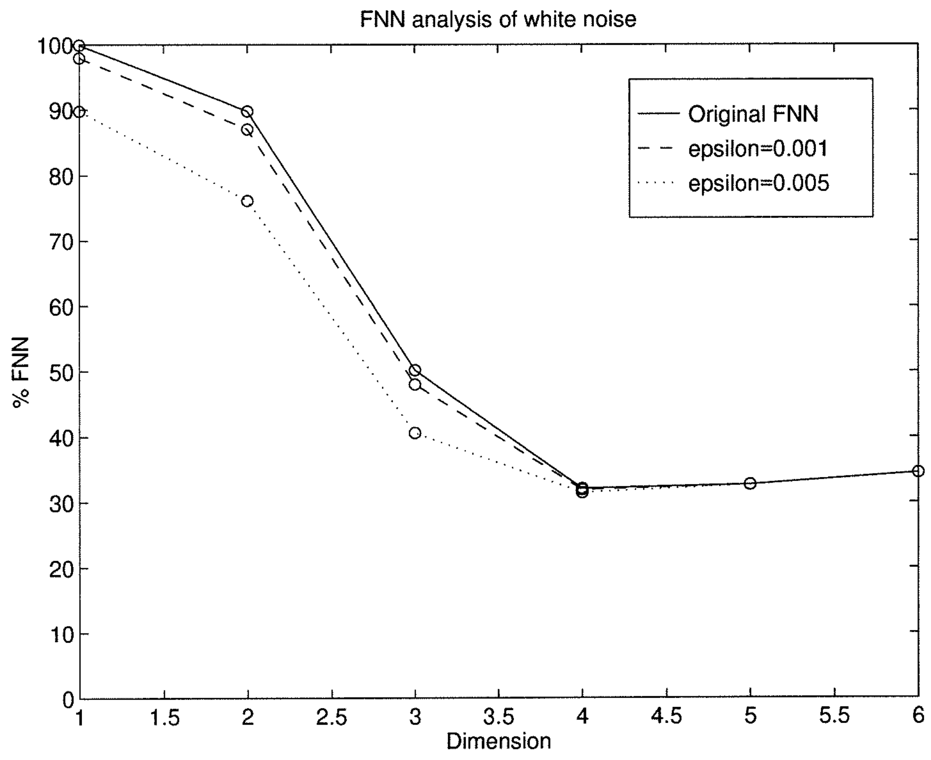


Figure 4.8: FNN plots for white noise with all FNN tests

is one which will be encountered when using FNN to analyze time-series from physical systems. A new ratio test which solves this problem is proposed. However, an easy method of determining the correct choice of thresholds is a problem which remains to be solved (as it does with the original FNN algorithm). Some theoretical results and other guidelines are given to aid the proper choice of the R and ϵ thresholds. The modified threshold test is then applied to time-series data from the Mackey-Glass equation and to a white noise time-series and the results are analyzed.

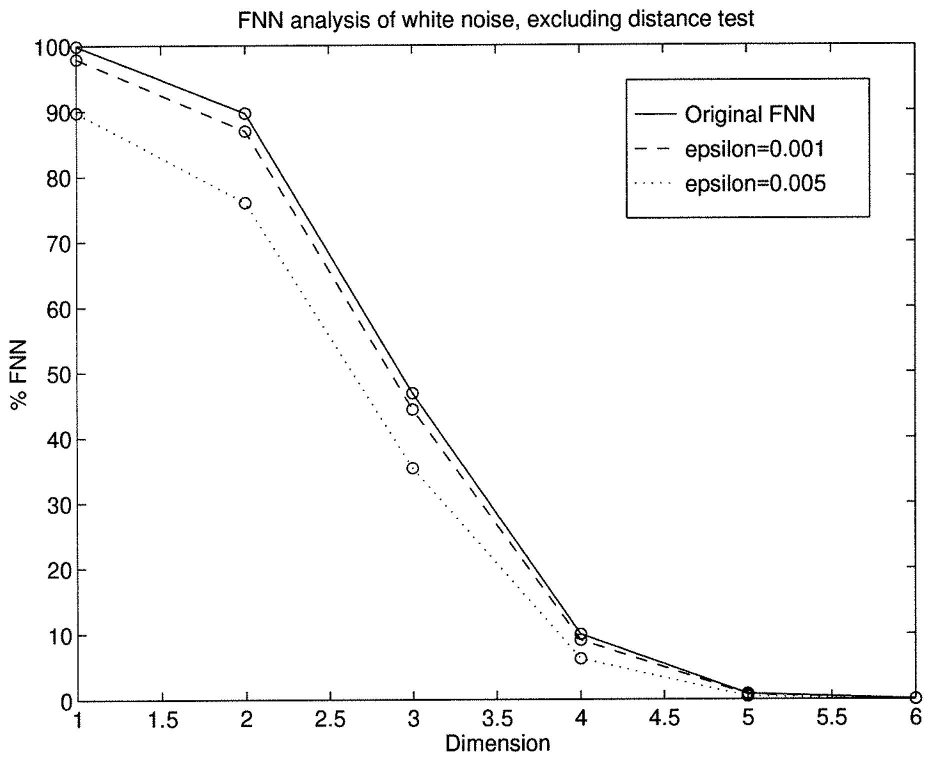


Figure 4.9: FNN plots for white noise, excluding R_{l+1} distance test

Chapter 5 The false nearest neighbors algorithm applied to input/output systems

5.1 Motivation

While the FNN algorithm was originally developed for the analysis of autonomous chaotic time series, for control purposes there is a need to develop models from input/output time series. Once a suitable accurate model is developed, a controller can be designed based on the model. In order to use the FNN algorithm for the analysis of input/output time series, the algorithm needs to be modified to search over both the number of delayed input and output terms.

The FNN algorithm can also be used in the problem of inferential measurement selection. In this problem, a single primary output should be predicted from a number of other secondary outputs. Since a large number of secondary outputs are typically available, the problem is to determine which set of secondary outputs should be used for prediction of the primary output. Once again, the FNN algorithm is presented as a tool for solving this problem.

In this chapter, numerous examples are presented to illustrate how the FNN algorithm can be used successfully as a tool in the nonlinear identification process. The FNN algorithm is applied to input/output time-series from an electrical leg stimulation experiment, a simulation of a continuous polymerization experiment, and a simulated pulp digester. The FNN algorithm for inferential measurement selection is also applied to steady-state data from a binary distillation column simulation. The results of the FNN algorithm are verified by modeling the nonlinear dynamics of these systems using the MARS modeling scheme (Friedman, 1991).

5.2 FNN and input/output dynamics

For determining the proper regressor for input/output dynamics, the only change to the original FNN algorithm involves the regression vector itself. For input/output dynamics, the regression vector must contain delayed versions of both the input and the output.

$$\psi_{l,m}(k) = [y(k-\tau), \ldots, y(k-l\tau), u(k-\tau), \ldots, u(k-m\tau)]. \tag{5.1}$$

In order to determine the proper dimension of the regressor, the percentage of false nearest neighbors must be determined for different numbers of delays in *both* the input and output. Here is an outline of the FNN algorithm for input/output data.

1. Identify the nearest neighbor to a given point in the regressor space. For a given regressor

$$\psi_{l,m}(k) = [y(k-\tau), \ldots, y(k-l\tau), u(k-\tau), \ldots, u(k-m\tau)] \tag{5.2}$$

   find the nearest neighbor $\psi_{l,m}(j)$ such that the distance d is minimized.

$$d = \|\psi_{l,m}(k) - \psi_{l,m}(j)\|_2. \tag{5.3}$$

2. Determine if the following expression is true or false.

$$\frac{\mid y(k) - y(j) \mid}{\|\psi_{l,m}(k) - \psi_{l,m}(j)\|_2} \leq R. \tag{5.4}$$

   If expression (5.4) is true, then the neighbor are *true* neighbors. If the expression is false, then the neighbors are *false* neighbors.

3. Continue the algorithm for all times k in the data set. Calculate the percentage of regressors in the data set which have false nearest neighbors.

4. Continue the algorithm for increasing l and m until the percentage of false

nearest neighbors drops to zero (or some acceptably small number). If the percentage of false neighbors is large, then the regressor vector must be extended to include more delayed input and/or output terms.

In general, for state-space systems of identical size, the regression vector for an input/output system needed to recreate the dynamics will have a dimension two times larger than the regressor needed for an autonomous system. Because there are two parameters present in the regression vector, a two dimensional search is needed to determine the proper number of delayed terms. This makes the FNN algorithm for input/output time-series slightly more difficult to implement than for autonomous systems. In addition, because of the larger regressor size, more data are needed by the FNN algorithm when analyzing problems with input/output dynamics.

5.3 FNN, input/output time-series, and noise corruption

The original FNN algorithm is not robust to the presence of noise in the time-series. The problem can be illustrated in the following way. When noise is present in the time-series, two identical regression vectors which would have identical outputs if no noise were present in the time-series can have outputs which differ by some finite amount. Therefore even when the noise corrupted time-series is embedded in a regression space with the proper dimension, the original FNN ratio test can fail which was shown in the previous chapter. Here the analysis of the previous chapter will be repeated for input/output time series.

Assume (as before), that the proper embedding dimension and the function relating the delayed inputs and outputs to the future output are known. Also assume that the observed inputs ($u^{obs} = u + \delta^u$) and outputs ($y^{obs} = y + \delta^y$) contain magnitude bounded noise ($\mid \delta^u \mid \leq \delta^u_\infty$, $\mid \delta^y \mid \leq \delta^y_\infty$). The observed regression vectors will also contain noise $\psi^{obs}_{l,n}(t) = [y^{obs}(t-\tau), \ldots, u^{obs}(t-\tau), \ldots] = [y(t-\tau)+\delta^y_1, \ldots, u(t-\tau)+\delta^u_1, \ldots]$ As before,

$$
\begin{aligned}
\mid y^{obs}(k) - y^{obs}(j) \mid & \leq \mid G(\psi_{l,m}(k)) - G(\psi_{l,m}(j)) \mid + 2\delta_{\infty}^{y} & (5.5) \\
& \leq \|DG(\psi_{l,m}(k))\|_2 \|\psi_{l,m}(k) - \psi_{l,m}(j)\|_2 + 2\delta_{\infty}^{y} & (5.6) \\
& \leq \|DG(\psi_{l,m}(k))\|_2 (\|\psi_{l,m}^{obs}(k) - \psi_{l,m}^{obs}(j)\|_2 + 2\sqrt{l{\delta_{\infty}^{y}}^2 + m{\delta_{\infty}^{u}}^2}) & (5.7) \\
& \quad + 2\delta_{\infty}^{y} & \\
& \leq \max_t \|DG(\psi_{l,m}(t))\|_2 (\|\psi_{l,m}^{obs}(k) - \psi_{l,m}^{obs}(j)\|_2 + 2\sqrt{l{\delta_{\infty}^{y}}^2 + m{\delta_{\infty}^{u}}^2}) & (5.8) \\
& \quad + 2\delta_{\infty}^{y} &
\end{aligned}
$$

This result can be put in a form similar to the FNN threshold test.

$$
\frac{\mid y^{obs}(k) - y^{obs}(j) \mid}{\|\psi_{l,m}^{obs}(k) - \psi_{l,m}^{obs}(j)\|_2} \leq \max_t \|DG(\psi_{l,m}(t))\|_2 + \frac{2\sqrt{l{\delta_{\infty}^{y}}^2 + m{\delta_{\infty}^{u}}^2}\max_t \|DG(\psi_{l,m}(t))\|_2 + 2\delta_{\infty}^{y}}{\|\psi_{l,m}^{obs}(k) - \psi_{l,m}^{obs}(j)\|_2} \tag{5.9}
$$

Once again, the first term is due to the gain of the noise-free system, and the second term results from noise contamination. In order to solve this problem, a new threshold test can be used for time-series where significant noise corruption is expected. A logical form of this threshold test based on the previous analysis is

$$
\frac{\mid y(k) - y(j) \mid}{\|\psi_{l,m}(k) - \psi_{l,m}(j)\|_2} \leq R + \frac{2R\sqrt{l{\epsilon^{y}}^2 + m{\epsilon^{u}}^2} + 2\epsilon^{y}}{\|\psi_{l,m}(k) - \psi_{l,m}(j)\|_2} \tag{5.10}
$$

where ϵ^u and ϵ^y are related to the expected magnitude of the noise contained in the input and output respectively.

Notice that this threshold test has two distinct limits. When the data are "dense", the second term on the right hand side dominates. In this limit, the new threshold test simply determines whether the two neighbors outputs are within some specified noise-related distance $(2R\sqrt{l{\epsilon^{y}}^2 + m{\epsilon^{u}}^2} + 2\epsilon^{y})$ of one another. In the limit where neighbors are relatively far apart (where problems associated with noise corruption are not expected), this threshold test reverts back to the original FNN threshold test.

While this new threshold test does a better job in dealing with noise corrupted data, there are more parameters to be determined. The main parameter R should be chosen as in the original FNN algorithm, and the terms ϵ^y and ϵ^u should be no larger than the magnitude of the expected noise in the output and input respectively. When the magnitude of the noise is known, it has been observed that choosing a value of ϵ equal to around $1/10^{\text{th}}$ of the standard deviation of the noise leads to good results when using this modified FNN algorithm.

5.4 Case studies

5.4.1 Electrical leg stimulation

The first example is the identification of the dynamics of a human leg stimulated by an electrical signal. Jan Schultheiss was involved in research to help paraplegics walk with the aid of a controlled electrical signal at the Swiss Paraplegic Center at Balgrist Hospital in Zürich Switzerland (del Re, Kraus, Schultheiss and Gerber, 1994). The data come from an experiment which measures the effect of stimulating the quadriceps muscle with an electrical signal. The input is the pulse-width of an electrical stimulation signal to the muscle, and the output is the angle of leg extension measured at the knee. The final goal of this research project is to design a controller for helping paraplegics to walk by utilizing controlled electronic stimulation.

The time-series data consists of 654 input/output samples where the pulse-width of the electronic input which stimulates the leg is scaled to lie between 0 and 1. The leg angle output is scaled such that the full leg extension (or full quadriceps contraction) is 1, the knee being at a resting position is 0, and a leg which swings past its normal resting position on relaxation of the quadriceps causes a negative output. The sampling time for the system is 100 ms. A plot of the entire time-series to be analyzed can be seen in Figure 5.1.

The original FNN algorithm with a threshold $R = 10$ was applied to the entire time-series. Since the time-series is relatively short, problems associated with noise

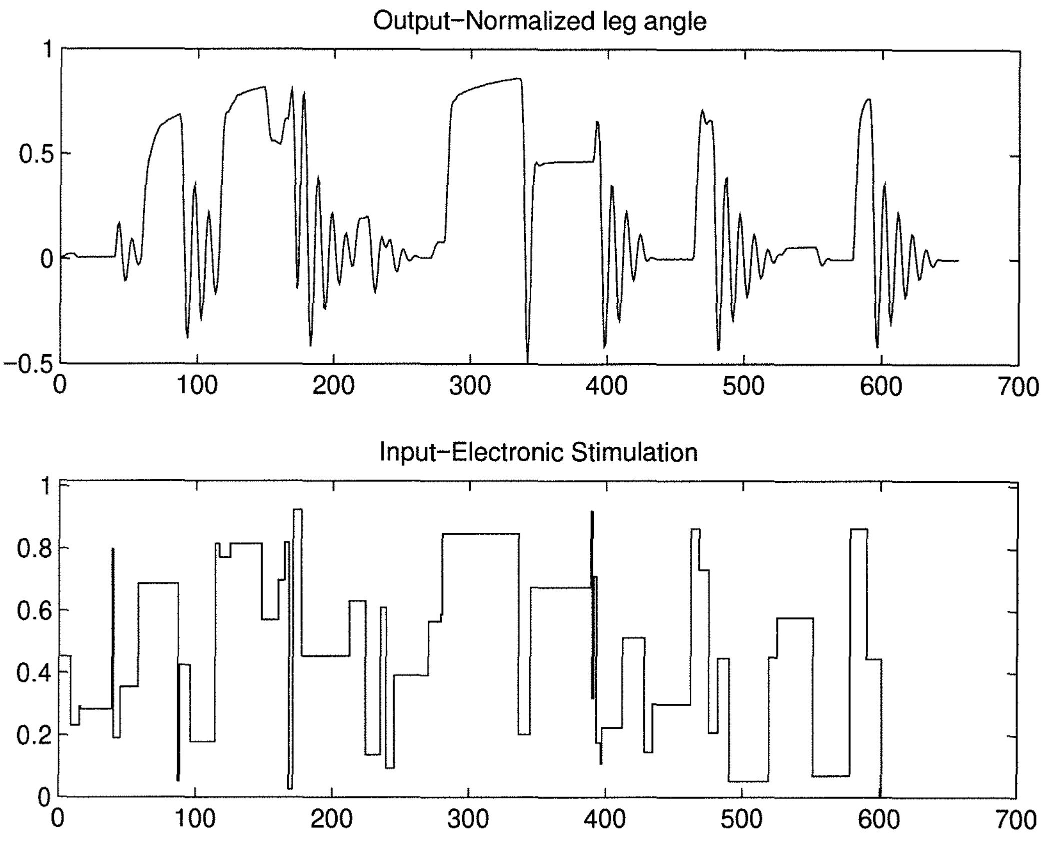


Figure 5.1: Leg stimulation, experimental time-series

corruption are not expected. The results of the FNN algorithm for this data set are given in Table 5.1. By examining the results, it can be seen that the percentage of false nearest neighbors drops to a low percentage of 1.1 when the regression function takes the form

$$y(t) = G[y(t-1), y(t-2), u(t-1), u(t-2)]. \tag{5.11}$$

While it is difficult to determine the definitive "proper" form of the regression vector since there are so little data, it appears that utilizing 2 delayed versions of both the output and input should lead to accurate prediction. Additionally, a regression vector with 2 delayed output terms and a single input delay may work reasonably well since the percentage of FNN is again fairly small. To illustrate the effect of a change in the threshold value R, the results of the FNN algorithm with $R = 15$ are presented in

% FNN	Output delays l			
Input delays m	0	1	2	3
0	100.0	72.2	13.5	4.2
1	87.4	36.5	5.5	2.6
2	82.2	27.8	**1.1**	0.3
3	74.5	23.2	0.5	0.3

Table 5.1: FNN results for leg stimulation data, R=10

Table 5.2. While the percentage of false nearest neighbors is smaller than in Table 5.1

% FNN	Output delays l			
Input delays m	0	1	2	3
0	100.0	68.3	6.8	3.5
1	87.2	23.8	2.3	1.4
2	80.5	18.5	**0.8**	0.3
3	70.0	16.5	0.5	0.3

Table 5.2: FNN results for leg stimulation data, R=15

for each entry, qualitatively the results are similar. In this case, the results of the FNN algorithm are relatively unchanged over a wide range of threshold values R.

In order to verify the results of the FNN algorithm, the nonlinear functional fitting algorithm MARS (Friedman, 1991) was used to determine the function G for regressors with the same number of delayed terms analyzed in Tables 5.1 and 5.2. The first 500 points of the time-series are used to build a model and the MSPE (mean squared prediction error) is determined by comparing the results of the MARS model to the last 154 points of the experimental time-series. After the model was found for a given regression vector, two types of error analysis were performed. The first determines the one-step ahead prediction error, where the actual past outputs and inputs are used to predict the next output of the system. The errors of the one-step ahead analysis are given in Table 5.3. The second analysis involves a simulated model which utilizes the inputs and only the initial conditions of the experimental time-series. Obviously, the second method typically has a larger error and the results

MSPE	Output delays l			
Input delays m	0	1	2	3
0	4.7×10^{-2}	5.3×10^{-3}	4.1×10^{-4}	1.9×10^{-4}
1	7.7×10^{-2}	4.1×10^{-3}	2.7×10^{-4}	1.4×10^{-4}
2	2.2×10^{-2}	4.7×10^{-3}	$\mathbf{1.9 \times 10^{-4}}$	4.8×10^{-5}
3	1.8×10^{-2}	4.0×10^{-3}	1.6×10^{-4}	6.5×10^{-5}

Table 5.3: MSPE error for leg stimulation one-step ahead prediction

of this method are given in Table 5.4.

MSPE	Output delays l			
Input delays m	0	1	2	3
0	4.7×10^{-2}	4.6×10^{-2}	4.5×10^{-2}	4.3×10^{-2}
1	3.9×10^{-2}	1.4×10^{-2}	5.5×10^{-3}	8.1×10^{-3}
2	2.8×10^{-2}	2.1×10^{-2}	$\mathbf{7.5 \times 10^{-3}}$	2.9×10^{-3}
3	2.8×10^{-2}	2.3×10^{-2}	1.9×10^{-3}	2.9×10^{-3}

Table 5.4: MSPE error for leg stimulation simulation

In examining these results, it appears that large values of FNN correspond to larger values of the MSPE (mean square prediction error) for both one-step prediction and simulation. For the one-step ahead prediction, the MSPE should always decrease when more terms are used. However, for the simulation results the error is not guaranteed to decrease since the error resulting from a single step may compound when previous predicted outputs are used to determine future outputs. For these results, it appears that the MSPE does not significantly decrease when including more than two delayed version of both the input and output. To give an idea of the accuracy of the MARS modeling scheme for the regression vector given in Equation (5.11), the results of the one-step ahead prediction and simulation are shown in Figures 5.2 and 5.3 respectively.

Without the FNN method, it would be necessary to build many models with different numbers of delayed terms. After these models were built the results would have to be analyzed and compared to determine a suitable number of delayed terms.

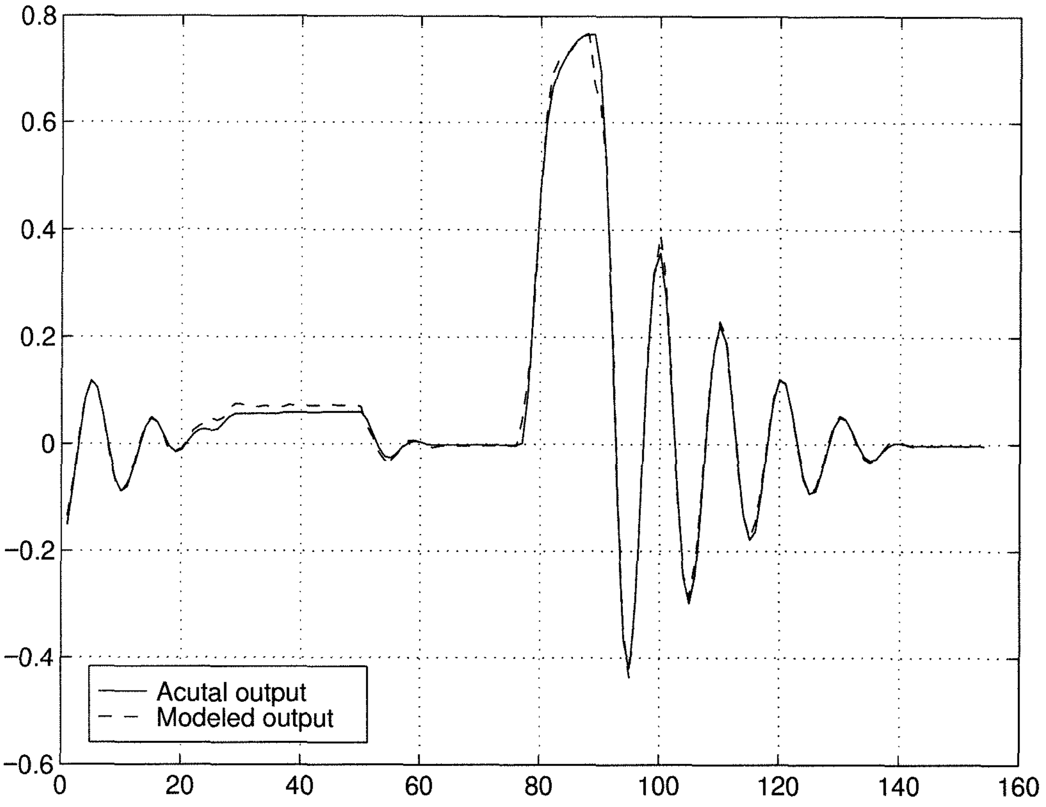


Figure 5.2: MARS modeled one-step ahead prediction for leg stimulation

With the FNN method, a suitable number of delayed terms can be determined in a single step before completing any nonlinear functional modeling. By utilizing the FNN algorithm, time can be saved when performing the nonlinear identification process.

5.4.2 Continuous polymerization reactor

The following example illustrates identification using data from a model of a continuous polymerization reactor. The model describes the free-radical polymerization of methyl methacrylate with azo-bis-isobutyronitrile as an initiator and toluene as a solvent. For further information on the details of this model and how it is derived see Doyle, Ogunnaike and Pearson (1995). The reaction takes place in a jacketed CSTR, and after some simplifying assumptions are made the first-principles model is:

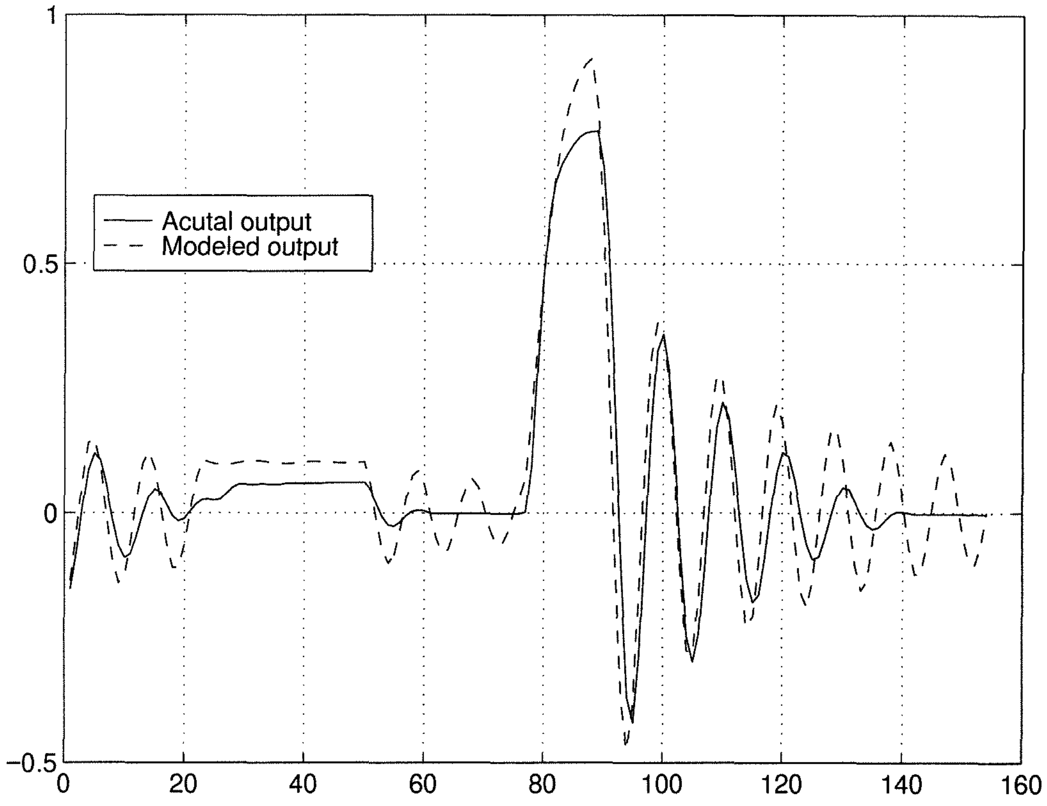


Figure 5.3: MARS modeled simulation results for leg stimulation

$$\begin{aligned}
\dot{x_1} &= 10(6 - x_1) - 2.4568x_1\sqrt{x_2} && (5.12)\\
\dot{x_2} &= 80u - 10.1022x_2 && (5.13)\\
\dot{x_3} &= 0.0024121x_1\sqrt{x_2} + 0.112191x_2 - 10x_3 && (5.14)\\
\dot{x_4} &= 245.978x_1\sqrt{x_2} - 10x_4 && (5.15)\\
y &= \frac{x_4}{x_3}. && (5.16)
\end{aligned}$$

The dimensionless state variable x_1 refers to the monomer concentration, x_2 to the initiator concentration, and x_4/x_3 is the number-average molecular weight (and also the output y). The input u is the dimensionless volumetric flow rate of the initiator.

Since a model of the system is known, large amounts of data can be collected for analysis. For this example, a time-series of length 50,000 is generated by forcing the system with a uniformly distributed random input over the range 0.007 to 0.015 which is passed through a zero order hold with a sampling time of 0.2. By driving the system with this input signal, an output which is roughly in the range of 26,000 to 34,000 is produced which is the desired operating range of the system (Ogunnaike, 1995).

The time-series, with sampling time 0.2, is then normalized so that both the input and output signal have zero mean and a standard deviation of 1. The original FNN algorithm is then applied to the data, and the results are given in Table 5.5. By

% FNN	Output delays l			
Input delays m	0	1	2	3
0	100.0	99.8	94.1	68.1
1	99.7	62.4	0.1	0.0
2	66.6	**0.0**	0.0	0.0
3	0.1	0.0	0.0	0.0

Table 5.5: FNN results for noise free polymerization data

examining the results, it appears that a model of the form

$$y(t) = G[y(t-0.2), u(t-0.2), u(t-0.4)] \tag{5.17}$$

should give an accurate estimate of future outputs.

Using MARS, a model of the dynamics is built using 2,000 training points. The results of one-step ahead modeling and the pure simulation are both nearly identical to the actual output of the system. For prediction of one hundred points not contained in the training set, the MSPE of the scaled data is 7.7×10^{-5} for the simulated output and 3.3×10^{-5} for the one-step ahead predicted output (see Figure 5.4). For comparative purposes, a MARS model excluding the term $y(t-0.2)$ was built and the MSPE error for the one-step ahead prediction is 1.4×10^{-2}. The difference between the actual and predicted outputs is visible, and is illustrated in Figure 5.5.

To simulate a more realistic identification problem, normally distributed noise

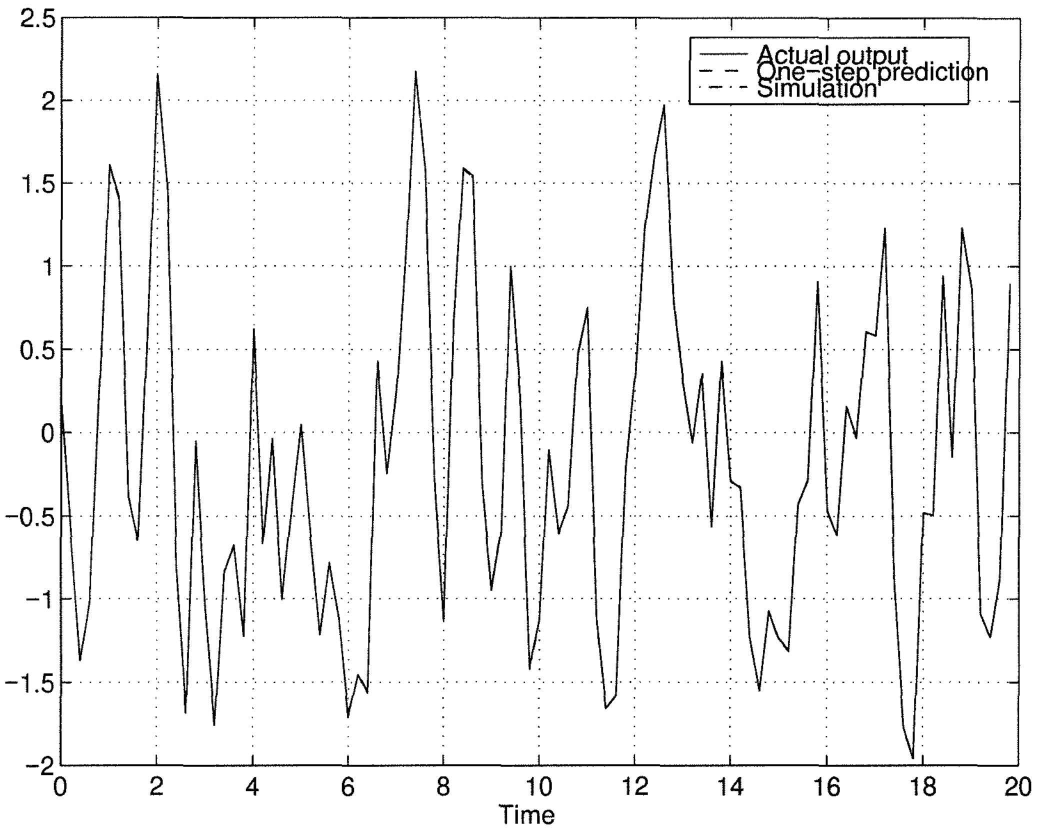


Figure 5.4: Results of MARS modeling for polymerization example ($l = 1$,$m = 2$)

with a standard deviation of 0.1 was added to the output of the time-series. The original FNN algorithm was applied again to this noise corrupted time-series and the results are presented in Table 5.6. Notice that the percentage of FNN is no longer zero for 1 output and 2 input delays. The modified algorithm was then applied to the data with $\epsilon^y = 0.01$ and $\epsilon^u = 0$. The results of the modified algorithm are presented in Table 5.7. While the term ϵ^y is much smaller than the size of the noise, the results of the algorithm still agree qualitatively with the results of the noise-free analysis. To show the sensitivity of the algorithm to the choice of the parameters, the results of the FNN algorithm with $\epsilon^y = 0.02$ are shown in Table 5.8.

One question which arises naturally from this analysis, is why is such a small regression vector is able to represent the dynamics of this system. Since the system has

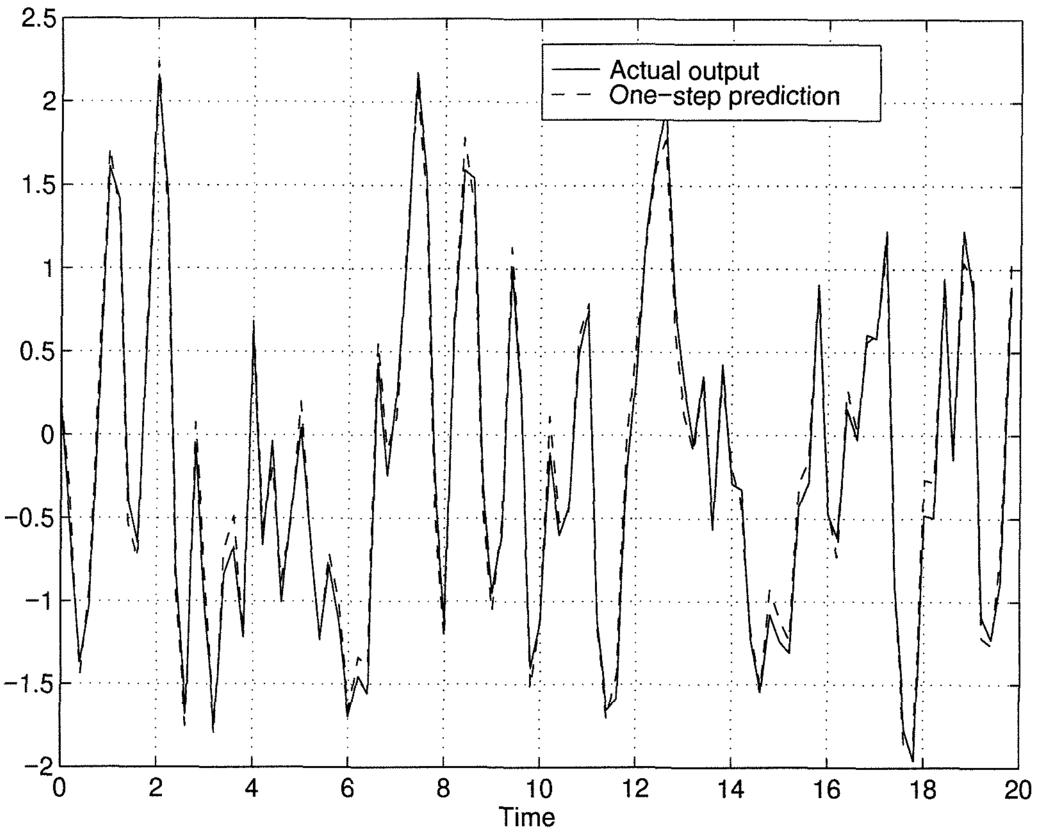


Figure 5.5: Results of MARS modeling for polymerization example ($l = 0$, $m = 2$)

four states, a sufficient (but not necessary) condition for representing the dynamics is a regression vector which includes 4 delayed versions of the input and output. In this case, the reason is that the system has two states which are nearly unobservable which leads naturally to a smaller description when analyzing the input/output dynamics. By linearizing the system about the operating point and performing a balanced realization, it is observed that two of the Hankel singular values are more than 2000 times larger than the remaining singular values. This means that the input/output dynamics of the system can be represented well by a reduced system with only two states (Zhou, Doyle and Glover, 1996). While this method is only valid for analysis of linear systems, it is not unexpected that the reduced characteristics of the system should carry over to the nonlinear system making the smaller dynamical description

% FNN	Output delays l			
Input delays m	0	1	2	3
0	100.0	99.8	94.2	67.5
1	99.7	71.4	3.8	0.1
2	73.4	**4.4**	0.1	0.0
3	2.4	0.1	0.0	0.0

Table 5.6: Original FNN algorithm results for noise corrupted polymerization data

% FNN	Output delays l			
Input delays m	0	1	2	3
0	100.0	85.9	75.8	48.5
1	97.5	21.7	0.0	0.0
2	67.0	**0.1**	0.0	0.0
3	1.9	0.0	0.0	0.0

Table 5.7: Modified FNN algorithm results for noise corrupted polymerization data, $\epsilon^y = 0.1$

predicted by the FNN algorithm feasible.

5.4.3 Simulated pulp digester

The third example consists of simulation data which come from a fundamental model of a pulp digester. The data were provided to us in a dimensionless form by Ferhan Kayihan and Marc Gelormino of Weyerhaeuser Corporate Research and Development in order to test our identification methods. The data consist of 2 inputs and 2 outputs. The time series also includes the effect of random unknown external disturbances to the simulated system. Since the values of these disturbances were not given to us, these disturbances can be thought of as unmeasured inputs to the system. For the generated data, the inputs and disturbances change only at the sampling times of the system. The sampling time was chosen by the researchers at Weyerhaeuser to correspond roughly to the dominant time constant of the system. The time series consists of 417 observed sampling periods.

While the FNN algorithm and analysis presented previously only consider single

% FNN	Output delays l			
Input delays m	0	1	2	3
0	100.0	72.8	58.9	33.1
1	95.4	3.8	0.0	0.0
2	60.5	**0.0**	0.0	0.0
3	1.4	0.0	0.0	0.0

Table 5.8: Modified FNN algorithm results for noise corrupted polymerization data, $\epsilon^y = 0.2$

input/single output systems (SISO), it is straight-forward to extend the FNN algorithm to account for systems with multiple inputs and a single output (MISO). For a system with two inputs the regression vector must be extended, such that both inputs are included. The new prediction equation takes the following form

$$y(t) = G[y(t-\tau), \dots, y(t-l\tau), u_1(t-\tau), \dots, u_1(t-m_1\tau), u_2(t-\tau), \dots, u_2(t-m_2\tau)] \tag{5.18}$$

where l, m_1, and m_2 are the number of delays which need to be determined by the FNN algorithm. By extending the regressor vector ψ of the FNN algorithm to include the additional input term, the FNN algorithm can be used to search for the smallest number of delays in the output and both input dimensions needed to recreate the dynamics of the system. Since the output of the system is not affected by stochastic noise, the original FNN algorithm is applied to these data.

For systems with multiple outputs, each individual output can be analyzed separately with the FNN algorithm. It is possible that different outputs of the same system may need different numbers of delayed terms to represent the dynamics. An example of this is presented in some of the original work on the application of FNN to autonomous chaotic systems (Abarbanel, 1996). In this reference, it is shown that two different outputs of the same underlying chaotic system require a different number of delayed terms to predict the respective outputs.

For the first output, it was determined (using the average mutual information (Abarbanel et al., 1993)) that there is a time delay of approximately three sampling times before either of the two inputs affect the first output. This issue was examined since the researchers at Weyerhaeuser informed us that there may be significant time delays associated with this process. Instead of using the standard regression vector, the following regression vector which accounts for the time delay is used.

$$\psi_{l,m_1,m_2}(t) = [y(t-\tau), \ldots, y(t-l\tau), u_1(t-3\tau), \ldots, u_1(t-(2+m_1)\tau), u_2(t-3\tau), \ldots, u_2(t-(2+m_2)\tau)] \tag{5.19}$$

While the FNN algorithm could be used to determine that this time delay exists if sufficient data were available, since the amount of data is quite limited this type of preliminary analysis is necessary for accurate prediction of the proper regression vector. The results of the FNN analysis for a threshold $R = 10$ are given in Table 5.9.

From the FNN algorithm, it appears that the proper regression vector has $l = 0$, $m_1 = 2$, and $m_2 = 1$ or a representation of the form

$$y(t) = G[u_1(t-3), u_1(t-4), u_2(t-3)] \tag{5.20}$$

should be able to recreate the observed dynamics. To verify the results of the FNN algorithm a MARS model was built using the first 372 points in the time series. The results of the modeling were then verified using the last 43 points of the time series data. The results of the modeling using simulation from initial condition can be found in Figure 5.6. This model appears to be fairly accurate, especially considering that there are unmeasured disturbances to the system.

The results of FNN analysis for output 2 can be found in Table 5.10. For this output, there is no significant time delay between the input and output and it a model of the form

$$y(t) = G[y(t-\tau), u_1(t-\tau), u_1(t-2\tau), u_2(t-\tau)] \tag{5.21}$$

can be used to model the dynamics of the system. The results of MARS modeling for a simulation from initial conditions can be found in Figure 5.7.

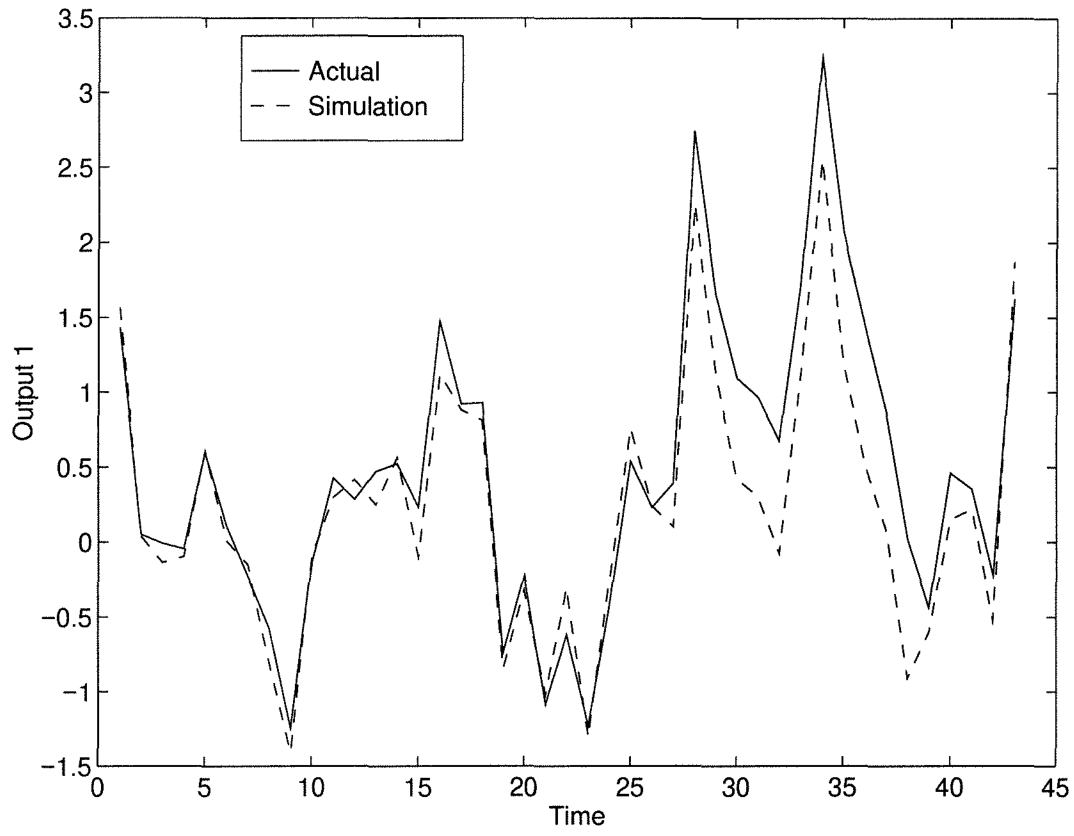


Figure 5.6: Results of MARS simulation for pulp digester output 1

5.5 FNN for inferential measurement selection

In many processes in the chemical industry, it is infeasible (for economic or physical reasons) to physically measure the output to be controlled. However in some of these cases, this output may be able to be determined or "inferred" from other outputs of the system. One example is composition control of a distillation column. The goal of control is often to hold the composition of the product streams constant, but composition analysis introduces a significant measurement delay into the system and the equipment is expensive and difficult to maintain (Mejdell and Skogestad, 1991). Since temperature measurements can be easily made along the column, the output compositions can be estimated using a number of temperature measurements along the column.

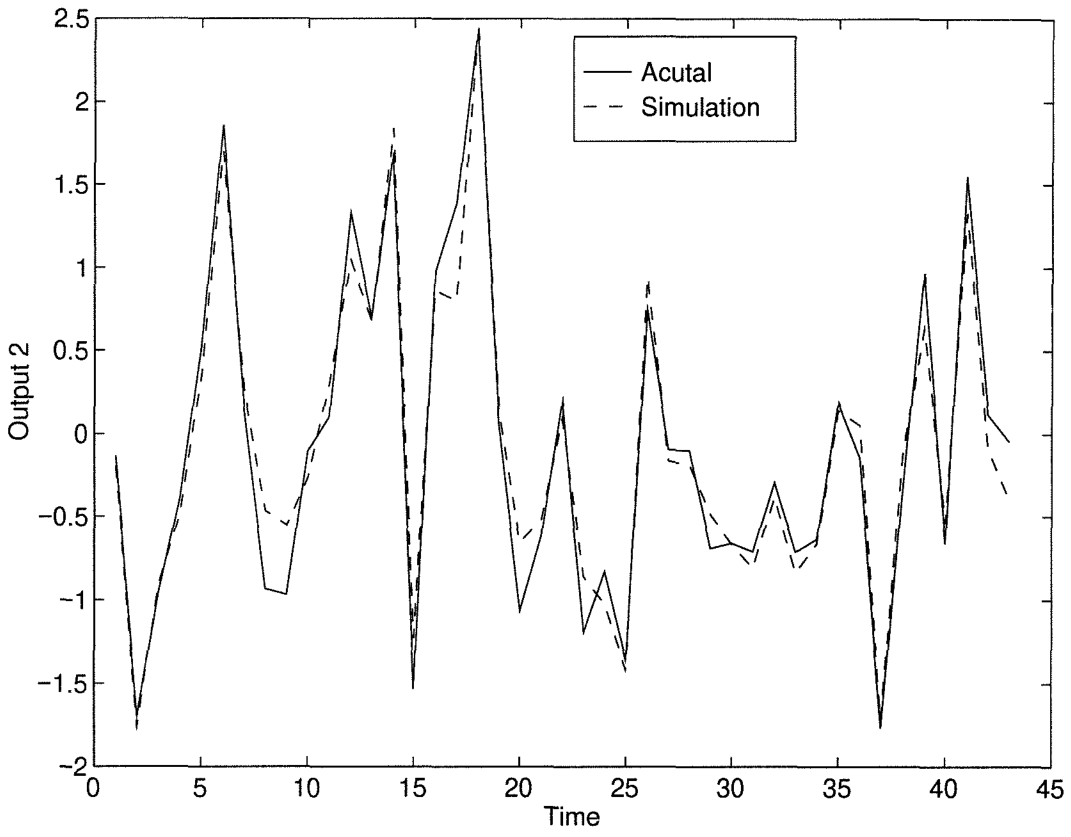


Figure 5.7: Results of MARS simulation for pulp digester output 2

Since the temperature can be measured anywhere along the length of the column, the question of where the temperature should be measured for accurate estimation of the composition remains. For linear systems, this problem has been examined using a number of different methods including PLS (Mejdell and Skogestad, 1991), μ-analysis (Lee and Morari, 1991), singular value analysis (Moore, Hackney and Canter, 1987), Brosilow's selection scheme (Joseph and Brosilow, 1978), and measurement selection based on Kalman filters (Harris, MacGregor and Wright, 1980). While all these methods determine the optimal measurement location in some sense, they are all based on utilization of a linear estimator. To our knowledge, the problem of determining suitable measurement locations for nonlinear inferential prediction has not been studied.

5.5.1 Applying the FNN algorithm

The FNN algorithm for inferential measurement selection is very similar to the original algorithm. However, in this case the regression vector consists of secondary outputs rather than delayed versions of the input and output. In addition, there is no easy way to determine a method for adding terms to the regressor as in the input/output case. In fact, to be sure the optimal measurement set is chosen all combinations of measurements in each dimension must be evaluated.

Here is an outline of the FNN algorithm for inferential measurement selection:

1. Identify the closest point to a given vector in the regression space of secondary variables. For the regressor at time k

   $$\Omega_{i_1,\ldots,i_l}(k) = [y^s_{i_1}(k), \ldots, y^s_{i_l}(k)] \tag{5.22}$$

   find the regressor $\Omega_{i_1,\ldots,i_l}(j)$ in the data set such that the distance

   $$d = ||\Omega_{i_1,\ldots,i_l}(k) - \Omega_{i_1,\ldots,i_l}(j)||_2 \tag{5.23}$$

   is minimized.

2. Determine if the neighbors are true or false using the following expression

   $$\frac{\mid y^p(k) - y^p(j) \mid}{||\Omega_{i_1,\ldots,i_l}(k) - \Omega_{i_1,\ldots,i_l}(j)||_2} \leq R. \tag{5.24}$$

   A modified version of the threshold test which accounts for noise can be used as well.

3. Repeat the algorithm for all times k in the data set and compute the percentage of false nearest neighbors.

4. Compute the percentage of false neighbors for all combinations of secondary measurements in the dimension l.

5. If the percentage of false nearest neighbors is large for all combinations of measurements in the given regression dimension l, increase the number of secondary measurements in the regressor vector.

For systems which require even a moderately sized regression vector, computing the percentage of false nearest neighbors for all combinations of secondary outputs may be too time consuming. If the computational time is too large, the heuristic method of computing the number of false nearest neighbors for a given number of measurements, adding the single measurement which reduces the percentage of false neighbors in that dimension, and using this set of measurements to start the search in the next dimension may have to be used.

5.5.2 Example

For this example, inferential measurement selection will be performed for a binary distillation column. The column example here is Column A taken from another study on distillation dynamics (Skogestad and Morari, 1988). The Antoine equation is used to determine the temperature of the mixture on each tray and uniformly distributed noise with magnitude 0.1 C was added to the temperatures of each tray. Due to the choice of the Antoine parameters, the mixture has constant relative volatility and constant molar flows are also assumed. The specific parameters used to generate data for this example are given in Table 5.11.

The sample data consists of 500 randomly determined steady-states with varying feed composition z_f, distillate composition y_d, and bottoms composition x_b. For each steady-state the output composition and temperature along each tray of the column is recorded. The range of variation in these variables is similar to that used by Mejdell and Skogestad (1991). The feed composition varies from .4 to .6, the distillate composition from .970 to .997, and the bottoms composition from .003 to .03. For all linear analysis, both the temperature and composition were scaled logarithmically as is suggested in (Mejdell and Skogestad, 1991) to improve prediction. For nonlinear analysis, no scaling is performed.

The goal of the analysis, is to determine the "optimal" two tray locations to make temperature measurements for predicting the distillate composition y_d. By examining all combinations of two measurements, the FNN algorithm determines that the smallest number of false neighbors occurs when using trays 10 and 21. For comparison, a linearly based PLS measurement selection method (Mejdell and Skogestad, 1991) suggests using tray 2 and 21.

To compare the measurement selection methods, two different models were built to determine the distillate output from temperature measurements. One model was built using the MARS (nonlinear) modeling scheme, and the second model was built using a linear least-squares estimate. The results of the modeling for the two measurement prediction sets are given in Table 5.12. The FNN based measurement selection scheme seems to reduce the amount of error for a nonlinear estimation scheme while the PLS (linearly) based selection scheme reduces the error of linear estimation. In addition, the modeling error was determined for all combinations of two tray based measurements using the MARS scheme and a minimum error of 1.40×10^{-7} was found for trays 9 and 22. This is quite close to the amount of error found when using the trays suggested by the FNN analysis.

5.6 Conclusions

By determining the smallest regression vector dimension which allows accurate prediction of the output, the FNN algorithm should reduce the overall computational effort needed to perform nonlinear identification. Instead of repeating a large number of identification experiments with different sized regression vectors, FNN can determine the proper embedding dimension using relatively little computation time. After the proper dimension is determined, only a single nonlinear function need to be estimated. For these reasons, an FNN assisted identification scheme should save time when solving difficult nonlinear identification problems since the proper number of delayed terms can be determined without fitting any specific models to the system.

To illustrate this fact, the FNN algorithm was applied to a 2000 point data set from the polymerization example. The FNN algorithm was able to examine 16 different regression vectors (all combinations of 0 to 3 delayed inputs and outputs) in 8 seconds on a Sun SPARCstation 20. For comparison, the same data were modeled using the MARS modeling scheme. With MARS it took 10 seconds to determine the proper nonlinear model for a single regression vector and MARS is one of the faster methods of nonlinear modeling. To analyze a data set consisting of 50000 points, the FNN algorithm took 6.5 minutes to analyze the same set of 16 regression vectors.

The FNN algorithm appears to a useful tool for determining the proper embedding dimension for both input/output dynamics and inferential prediction. A number of examples illustrate the success of the FNN in determining the proper regression vector for accurate prediction. While the FNN algorithm is successful in determining the embedding dimension, it does not give any clues about the proper functional relationship between the regression vector and the output. While this might appear to be a drawback, the main advantage of FNN is that the process of determining the proper regressor is not dependent on a single model structure. This is especially important in nonlinear systems since a single model structure may not do a good job representing all possible relationships between regressors and outputs. Since FNN is not dependent on any single model structure, it can work in conjunction with any nonlinear functional modeling scheme and be a useful tool in the overall nonlinear identification process.

Output delays l	Input delays m_1	Input delays m_2	% FNN
0	0	0	100.00
0	0	1	94.63
0	0	2	47.32
0	1	0	96.34
0	1	1	35.85
0	1	2	3.66
0	2	0	35.37
0	2	1	**0**
0	2	2	0
1	0	0	93.66
1	0	1	39.02
1	0	2	3.66
1	1	0	36.10
1	1	1	0.98
1	1	2	0
1	2	0	4.39
1	2	1	0
1	2	2	0
2	0	0	50.00
2	0	1	4.39
2	0	2	0
2	1	0	4.39
2	1	1	0
2	1	2	0
2	2	0	0
2	2	1	0
2	2	2	0

Table 5.9: FNN algorithm results for pulp digester, output 1

Output delays l	Input delays m_1	Input delays m_2	% FNN
0	0	0	100.00
0	0	1	95.12
0	0	2	30.73
0	1	0	97.07
0	1	1	32.44
0	1	2	1.46
0	2	0	44.88
0	2	1	1.71
0	2	2	0
1	0	0	95.85
1	0	1	37.07
1	0	2	3.17
1	1	0	48.29
1	1	1	2.44
1	1	2	0
1	2	0	6.34
1	2	1	**0**
1	2	2	0
2	0	0	48.78
2	0	1	3.17
2	0	2	0
2	1	0	8.29
2	1	1	0
2	1	2	0
2	2	0	0.73
2	2	1	0
2	2	2	0

Table 5.10: FNN algorithm results for pulp digester, output 2

z_f	α	N_{trays}	N_f	D/F	L/F
0.5	1.5	40	21	.5	2.706

		Antoine Parameters		
Comp #	T_b(K)	A	B	C
1	341.9	15.83660	2697.55	-48.78
2	355.4	15.43113	2697.55	-48.78

Table 5.11: Distillation column example

Tray #	Prediction Scheme	
	Linear	MARS
10,21 (FNN scheme)	9.26×10^{-6}	1.57×10^{-7}
2,21 (PLS scheme)	5.09×10^{-6}	2.79×10^{-6}

Table 5.12: MSPE for different prediction schemes, distillation column

Part III

Nonlinear model reduction

Chapter 6 Model reduction of nonlinear systems

6.1 Introduction

While significant theoretical advances have been made recently in the control of nonlinear systems, for these methods to be successfully applied to physical systems an accurate low-dimensional dynamical description is needed. When the system models are of high dimension, controller synthesis using methods based on differential geometry becomes unwieldy due to algebraic manipulations needed for controller design. If model predictive control methods are being used, computational requirements may be large for system descriptions of even moderate dimension. For reasons which will be illustrated in the next paragraph, these concerns are especially important for chemical processes due to the way models are developed for these systems.

Dynamical descriptions of many chemical processes are derived from first-principle models of the physics and chemistry of the system. These dynamic models are developed by writing detailed descriptions of the reaction kinetics, thermodynamics, heat transfer, material and mass balances. Models developed in this way are surprisingly accurate but are typically of fairly high order and complexity. At the same time, these models generally exhibit behavior on widely differing time scales. If one is uninterested with behavior occurring on fast time scales, it is natural to consider a model reduction scheme which discards this portion of the dynamics of the model. Heuristically, models can be simplified by making pseudo steady-state or equilibrium assumptions. However, these assumptions are not mathematically rigorous. For this reason, assumption made in this way may not always be accurate.

Ideally, low-dimensional models should be developed from the complete first-principle models in some sort of optimal fashion. For linear systems, a number of techniques for model reduction exist. One commonly used method is balanced truncation, which was introduced in Moore (1981). In balanced truncation, first a balanced realization of the system is developed and then the smaller singular values of the balanced realization are discarded. In this way, an upper bound can be defined on the norm of the difference between the actual system and the reduced model

$$\|G(s) - G_r(s)\|_\infty \tag{6.1}$$

where $G(s)$ is the original model and $G_r(s)$ is the reduced model. A similar method can be used for frequency weighted balanced model reduction (Enns, 1984). The method of $\mathcal{H}_\infty$ balancing has also been extended to model reduction of nonlinear systems (Scherpen, 1996). The second common method of model reduction for linear systems involves minimizing the Hankel norm between the original system and the reduced model (Zhou et al., 1996).

In this work, systems which exhibit large time scale separations will be examined. For systems with stable dynamics and large time scale separations, the dynamics will converge very quickly to a reduced-order invariant manifold in the state-space. Once on this manifold the system will remain there and only the slower dynamics of the system take place. Since the majority of the system dynamics take place on this manifold, it seems natural that any model reduction scheme should find the location of this reduced manifold. On this manifold there are no fast dynamics, and this manifold will be called the *slow* manifold.

For linear systems, the process of discarding the fast modes of the system is done by performing a modal decomposition and truncating the fast modes of the system (Brogan, 1991). For the system

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{6.2}$$

where the eigenvalues of A are negative real and not repeated, this is done by finding the transformation which puts the A matrix into the Jordan form $T^{-1}AT = \Lambda$. Because of our assumptions on A, the matrix Λ will be diagonal and it is assumed that the eigenvalues in Λ are ordered with increasing magnitude. Transforming system (6.2) into a system with the new state-space coordinates $\bar{x} = T^{-1}x$, the result is

$$\begin{aligned} \dot{\bar{x}} &= \Lambda\bar{x} + B_T u \\ y &= C_T\bar{x} + Du \end{aligned} \tag{6.3}$$

where $B_T = T^{-1}B$ and $C_T = CT$. In this new set of coordinates, the "modes" of the system are decoupled because the matrix Λ is diagonal.

Now suppose that some of the eigenvalues have very large magnitude. Then the matrix Λ could be decoupled such that

$$\Lambda = \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix} \tag{6.4}$$

where Λ_1 contains eigenvalues which have small magnitude and Λ_2 contains eigenvalues which have large magnitude. The full system can then be written as

$$\begin{aligned} \begin{bmatrix} \dot{\bar{x}}_1 \\ \dot{\bar{x}}_2 \end{bmatrix} &= \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} + B_T u \\ y &= C_T \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \end{bmatrix} + Du. \end{aligned} \tag{6.5}$$

Since the states contained in $\bar{x}_2$ decay very quickly due to the fact that the eigenvalues in Λ_2 have a large magnitude when compared to Λ_1, a good approximation of this system is given by

$$\begin{aligned} \dot{\bar{x}}_1 &= \Lambda_1\bar{x}_1 + B_T^\star u \\ y &= C_T^\star\bar{x}_1 + Du \end{aligned} \tag{6.6}$$

where $B_T^\star$ and $C_T^\star$ are truncated so that they are of the appropriate dimension.

For a nonlinear system this type of modal decomposition will not work globally. This is because the transformation which "decouples" the modes is nonlinear. Determining this decomposition from the original equations is not an easy problem in general. For this reason, a computational method for determining this transformation will be utilized here. The first problem which will be tackled is how the "space" where the slow dynamics take place can be identified.

The algorithm of Maas and Pope (Maas and Pope, 1992) will be presented as a method for identification of invariant reduced-order manifolds for stable systems which exhibit the time-scale separation property. While this method has been published previously in the literature, theoretical justification for the algorithm was not presented in the original work. Here, it will be shown rigorously that the method correctly identifies the slow manifold. Before the theoretical results are presented, a brief background on the behavior of singularly perturbed systems will be presented. This method will be applied to two different examples, a distillation column and a two-phase chemical reactor. For each of these examples, the resulting reduced-order description will be compared to other standard methods of producing reduced order models. In addition, some ideas on how this method can be used to produce reduced-order models will be discussed.

6.1.1 Singular perturbation theory

Systems of ordinary differential equations (ODEs) which exhibit dynamic behavior evolving over vastly different time scales can be represented by singularly perturbed systems. The standard form of the singular perturbation model consists of a state-space model in which the derivatives of a number of the states are multiplied by a small parameter ϵ.

$$\begin{aligned} x' &= f(x, y, \epsilon) \\ \epsilon y' &= g(x, y, \epsilon) \end{aligned} \qquad (6.7)$$

where $' = \frac{d}{dt}$, $x \in \Re^l$, $y \in \Re^m$, and $\epsilon \in \Re$. When ϵ is small, this system of ODEs will exhibit a certain characteristic two-time-scale behavior. When $\epsilon \to 0$, System (6.7) becomes

$$\begin{aligned} x' &= f(x, y, 0) \\ 0 &= g(x, y, 0). \end{aligned} \tag{6.8}$$

In this limit, $g(x, y, 0) = 0$ defines an invariant region upon which the dynamics $x' = f(x, y, 0)$ take place. Outside of this region, the dynamics are not defined since the condition $g(x, y, 0) = 0$ does not hold.

By scaling time such that $\tau\epsilon = t$, system (6.7) becomes

$$\begin{aligned} \dot{x} &= \epsilon f(x, y, \epsilon) \\ \dot{y} &= g(x, y, \epsilon) \end{aligned} \tag{6.9}$$

where $^{\cdot} = \frac{d}{d\tau}$. For $\epsilon \neq 0$, system (6.9) is equivalent to system (6.7). However, the behavior is quite different when $\epsilon \to 0$. In this limit, system (6.9) becomes

$$\begin{aligned} \dot{x} &= 0 \\ \dot{y} &= g(x, y, 0). \end{aligned} \tag{6.10}$$

Notice that for system (6.10) x is constant and only the value of y changes with time.

Because of the scaling of time with ϵ, system (6.7) is called the *slow* system and system (6.9) is called the *fast* system. When $\epsilon = 0$ the set in the state space for which slow dynamics are defined, $g(x, y, 0) = 0$, is the set of critical points for the fast system. In the two limiting cases, analysis of the dynamics is simplified by the fact that the state-space dynamics are limited to a manifold which is smaller than the original state-space. For process control, the more compelling limiting case is the slow system. Since the system trajectories quickly converge to the reduced order manifold defined by $g(x, y, 0) = 0$ when the fast dynamics are stable, control may not be needed for the fast dynamics. In order to improve performance in the region of slow dynamics, only a description of these slow dynamics may be needed.

Unfortunately, many physical systems which exhibit behavior consistent with singularly perturbed systems do not take the standard form of System (6.7) after first-principles modeling. In order to convert these systems into the standard form, transformations based on the physical insight into the specific system can be used (Khalil, 1996). Often these transformations are not obvious, as choosing a suitable "small" parameter can be difficult. Converting a system into the "optimal" form such that the slow and fast dynamics are completely decoupled may not even be possible using physical insight. Fortunately, the optimal slow manifold can be identified from the complete description of the system dynamics using a computational algorithm. The algorithm detailed in the next section will illustrate a method of identifying this manifold of slow dynamics.

6.2 The Maas and Pope algorithm

The algorithm described in the paper of Maas and Pope (1992) was developed to examine problems associated with combustion. The algorithm computes discrete points located on the slow manifold from a dynamical description of a system which consists of ODEs which exhibit time scale separation. The motivation for this algorithm is the fact that simulations of complete models of detailed combustion kinetics can take hundreds of hours of supercomputer time. By identifying the manifold of slow dynamics, it is hoped that methods can be developed for producing a reduced model of the system in order to decrease the computational time needed for accurate simulations.

For the purpose of describing the Maas and Pope algorithm assume the following system, which exhibits time scale separation, is given

$$\dot{z} = F(z) \tag{6.11}$$

with $z \in \Re^n$. From this description, an $n_r < n$ dimensional manifold where the slow dynamics exist is to be found. The dynamics of the overall system from arbitrary initial conditions should decay very quickly onto this n_r-dimensional invariant slow manifold. While the original algorithm does not develop a reduced model of the dynamics on the slow manifold, it does identify individual points on the slow manifold.

The idea behind the algorithm presented in the paper of Maas and Pope is based mainly on local arguments, and a rigorous justification of the algorithm is not presented in the original paper. In the paper it is stated that "While the development is mathematical, no attempt at rigor is made" and the analysis of the algorithm is given only for linearized dynamics. The algorithm makes use of the eigenvalue decomposition of the Jacobian of the system dynamics. Take the Jacobian of F evaluated at z_0 defined as

$$J = \frac{\partial F}{\partial z}(z_0). \tag{6.12}$$

It is stated that the eigenvalues of J identify the time scales associated with movement in the state-space. In addition the eigenvectors associated with these eigenvalues describe the "characteristic directions" associated with these time scales.

The goal of the Maas and Pope (1992) algorithm is to identify the set in the state-space where the projection of the dynamics onto the directions consisting of only fast linearized dynamics is zero. On this surface, the dynamics of the fastest time scales should be at equilibrium. The manifold is found in the following manner. Suppose the basis of eigenvectors of the Jacobian at a point z is given by

$$V(z) = \begin{pmatrix} | & | & & | \\ v_1 & v_2 & \cdots & v_n \\ | & | & & | \end{pmatrix}, \tag{6.13}$$

where the eigenvectors are sorted such that the absolute value of the real part of the corresponding eigenvalues is decreasing. The inverse of this matrix is

$$V^{-1}(z) = \begin{pmatrix} - & \tilde{v}_1 & - \\ - & \tilde{v}_2 & - \\ & \vdots & \\ - & \tilde{v}_n & - \end{pmatrix}. \tag{6.14}$$

The inverse also defines the coordinate transformation from the original coordinates to a basis consisting of the eigenvectors. On the low dimensional manifold of slow dynamics the following relationship holds

$$W(z)F(z) = 0 \tag{6.15}$$

where

$$W(z) = \begin{pmatrix} - & \tilde{v}_{n_r+1} & - \\ - & \tilde{v}_{n_r+2} & - \\ & \vdots & \\ - & \tilde{v}_n & - \end{pmatrix}. \tag{6.16}$$

Note that W is a function of z, and the Jacobian of F (specifically its eigenvalue-eigenvector decomposition) must be evaluated at each point of interest. The matrix W projects the dynamics onto the eigenvectors corresponding to the $n - n_r$ smallest eigenvalues.

For a linear system, the matrix V^{-1} would be constant. The matrix W would also be constant and would be the matrix which transforms the original coordinates to $\bar{x}_2$, the states which decay quickly. The quantity $WF = 0$ is then equivalent to $\Lambda_2 \bar{x}_2 = 0$.

Since the eigenvectors are not orthogonal, the matrix $W(z)$ may be ill-conditioned and the inversion may be problematic numerically. The solution to the above problem is better behaved computationally if the Schur basis is used to define the spaces referring to the slow and fast eigenvalues. The Schur basis is orthogonal, and the computations associated with it are better behaved numerically. The Schur decomposition is defined as follows (Horn and Johnson, 1985)

$$J = QNQ^T, \tag{6.17}$$

where Q is an orthonormal matrix and N is a triangular matrix which is sorted such that the eigenvalues appearing along the diagonal are sorted in order of descending magnitude of the real part. If Q is

$$Q = \begin{pmatrix} | & | & & | \\ q_1 & q_2 & \cdots & q_n \\ | & | & & | \end{pmatrix} \tag{6.18}$$

the slow manifold is defined by points z such that

$$Q_L^T(z)F(z) = \begin{pmatrix} - & q_{n_r+1}^T & - \\ - & q_{n_r+2}^T & - \\ & \vdots & \\ - & q_n^T & - \end{pmatrix} F(z) = 0. \tag{6.19}$$

In the original work, both Newton's method and continuation methods are suggested for finding the manifold where $Q_L^T(z)F(z) = 0$. Since any fixed point z_f will be on the manifold, as $F(z_f) = 0$ for a fixed point by definition, a fixed point can be identified and used as a starting point for the algorithm. In order to make the equations consistent, n_r additional parametric equations must be defined. An example would be fixing n_r of the variables to some given values by solving the extended set of equations

$$\begin{pmatrix} Q_L^T(z)F(z) \\ P(z,\tau) \end{pmatrix} = 0 \tag{6.20}$$

where

$$P(z,\tau) = z_{n_i} - \tau_i \qquad i = 1, \ldots, n_r \tag{6.21}$$

where τ is a vector of fixed parameters. The solution of Equation (6.20) for z give a single point on the slow manifold. By solving this equation repeatedly with different values of the parameter vector τ, numerous points on the manifold can be computed which are arbitrarily close together by the choice of parameters τ.

For reasons of computational speed, in this work a nonlinear optimization is used to identify the slow manifold. An interface has been written to the program NPSOL (Gill, Murray, Saunders and Wright, 1994) such that the following problem is solved

$$\min_{z \in \Re^n} \|Q_L^T(z)F(z)\|_2 \tag{6.22}$$

with n_r of the states of the system fixed

$$z_{n_i} = \tau_i \quad i = 1, \ldots, n_r. \tag{6.23}$$

Note that τ_i behave like constraints on some of the state variables to make the minimization solution unique. The choice of n_i determines how the slow manifold should be parameterized. If the objective function is equal to zero after the minimization at a single point z, the condition for identifying the slow manifold has been met. By repeating the algorithm for different values of the vector τ, a number of different points on the manifold can be calculated be repeating the process.

6.3 Validating the algorithm

In this section, a justification for the slow manifold identification algorithm of Maas and Pope will be given for a system of ODEs in the standard singular perturbation form. For the limiting case where the time separation between the slow and fast dynamics is infinite ($\epsilon \to 0$) the proof is fairly straight-forward. In the case where ϵ is small but finite, the problem is more difficult and results from singular perturbation theory known as Fenichel fibering are used to derive the Fenichel normal form. Once this normal form is derived, it is can be shown that the Maas and Pope algorithm identifies the proper slow manifold. While only the standard form of singularly perturbed systems is considered here, it is expected that the results should extend to any singularly perturbed system since a coordinate transformation exists for transforming any singularly perturbed system into the standard form (Fenichel, 1983). It should be noted that much of the background information in this section (especially for the finite time scale separation results) comes from the papers of Jones (1994) and Fenichel (1979). For the results here, it will be assumed that the dynamics appear in the standard form for singularly perturbed systems. When the dynamics are not in this form originally, there always exists a nonlinear change of coordinates to put them in this form (Fenichel, 1979). For this reason, working with the special form should not be problematic.

When $\epsilon \to 0$ there is an infinite time scale separation between the fast and slow dynamics. In this case the set of critical points of (6.10) is defined by $g(x, y, 0) = 0$. This is the set where the slow dynamics defined by (6.8) exist. The manifold where $g(x, y, 0) = 0$ is defined by a set of m equations in $\Re^{l+m}$. This manifold is expected to be l dimensional (at least locally) by the preimage theorem of differential geometry and it is possible that the manifold has a boundary. In this paper, this manifold will be called $\mathcal{M}_0 = \{(x, y) : g(x, y, 0) = 0\}$ where the subscript refers to the fact that $\epsilon = 0$.

Before presenting the formal validation of the algorithm, a list of assumptions will be presented. The first assumption is on the smoothness of the functions f and g. This assumption ensures that the transformation to the Fenichel Normal Form is smooth.

Assumption 6.3.1 *The functions f and g are assumed to be C^∞ on the set $U \times I$, $U \in \Re^{l+m}$ and the interval I contains* 0.

The second assumption concerns the dynamics on the manifold $\mathcal{M}_0$. Since the manifold is a set of critical points, directions normal to $\mathcal{M}_0$ should correspond to eigenvalues which are nonzero and eigenvalues tangent to $\mathcal{M}_0$ should be identically zero.

Definition 6.3.1 *The manifold $\mathcal{M}_0$ is normally hyperbolic if the linearization of (6.10) has exactly l eigenvalues on the imaginary axis* $\mathrm{Re}(\lambda) = 0$ *for all* $(x, y) \in \mathcal{M}_0$.

Assumption 6.3.2 *The manifold $\mathcal{M}_0$ is normally hyperbolic.*

Since the linearization of Equation (6.9) for $\epsilon = 0$ is

$$\begin{aligned} \begin{pmatrix} \dot{\delta x} \\ \dot{\delta y} \end{pmatrix} &= \begin{pmatrix} \epsilon\frac{\partial f}{\partial x}(x,y,\epsilon) & \epsilon\frac{\partial f}{\partial y}(x,y,\epsilon) \\ \frac{\partial g}{\partial x}(x,y,\epsilon) & \frac{\partial g}{\partial y}(x,y,\epsilon) \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 \\ \frac{\partial g}{\partial x}(x,y,0) & \frac{\partial g}{\partial y}(x,y,0) \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}, \end{aligned} \tag{6.24}$$

$\mathcal{M}_0$ is normally hyperbolic whenever $\frac{\partial g}{\partial y}(x,y,0)$ is nonsingular for $(x,y) \in \mathcal{M}_0$.

Since $\mathcal{M}_0$ may have boundary, the manifold cannot be invariant in general since trajectories can escape the manifold at the boundary. However, it can be shown that $\mathcal{M}_0$ will be *locally invariant.*

Definition 6.3.2 *A set $\mathcal{M}$ is locally invariant under the set of differential equations (6.9) if it has a neighborhood $\mathcal{N}$ such that no trajectory can leave $\mathcal{M}$ without leaving $\mathcal{N}$.*

Note that the only difference between local invariance and invariance is at the boundaries of the set $\mathcal{M}$.

In order to simplify the notation, it will be assumed that the manifold $\mathcal{M}_0$ can be represented as the graph of a function h. Since the matrix $\frac{\partial g}{\partial y}(x,y,0)$ is nonsingular on the manifold due to the assumption of normal hyperbolicity, it is locally invertible for any $(x,y) \in \mathcal{M}_0$. By the Implicit Function Theorem, y can always found as a function of x locally. Assuming that a solution can be made globally, then $\mathcal{M}_0$ can be represented by a graph of a function h^0.

Assumption 6.3.3 *$\mathcal{M}_0$ can be represented as a graph $\mathcal{M}_0 = \{(x,y) : y = h^0(x)\}$ for some compact domain $x \in K$ where $K \subset \Re^l$.*

As y can be found from x locally, the main assumption needed here is that the function $h^0(x)$ can be pieced together globally from the local descriptions.

6.3.1 Infinite time scale separation

For the limiting case $\epsilon \to 0$, the representation of the dynamics with the fast time scaling (6.9) will be utilized. For this set of equations, the dynamics can be linearized about a point $(x, h^0(x)) \in \mathcal{M}_0$. The linearization takes the form

$$\begin{aligned}
\begin{pmatrix} \dot{\delta x} \\ \dot{\delta y} \end{pmatrix} &= \begin{pmatrix} \epsilon\frac{\partial f}{\partial x}(x,h^0(x),0) & \epsilon\frac{\partial f}{\partial y}(x,h^0(x),0) \\ \frac{\partial g}{\partial x}(x,h^0(x),0) & \frac{\partial g}{\partial y}(x,h^0(x),0) \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} \\
&= \begin{pmatrix} 0 & 0 \\ 0 & \frac{\partial g}{\partial y}(x,h^0(x),0) \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}
\end{aligned} \tag{6.25}$$

as $\frac{\partial g}{\partial x}(x, h^0(x), 0) = 0$ on $\mathcal{M}_0$ by the definition of $h^0(x)$ and $\epsilon = 0$.

The eigenvalues of the Jacobian given in (6.25) can be decomposed into two distinct groups. The first group of eigenvalues is identically zero and the set of eigenvectors associated with these eigenvalues can be defined as $\left\{ \begin{pmatrix} e_1 \\ 0 \end{pmatrix}, \begin{pmatrix} e_2 \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} e_l \\ 0 \end{pmatrix} \right\}$ where the orthonormal vectors e_i form a basis of the space $\Re^l$. Since the matrix $\frac{\partial g}{\partial y}(x, h^0(x), 0)$ is nonsingular, the eigenvalues of this matrix are bounded away from zero. The invariant eigenvector space associated with these nonzero eigenvalues is spanned by the orthonormal basis vectors $\left\{ \begin{pmatrix} 0 \\ \hat{e}_1 \end{pmatrix}, \begin{pmatrix} 0 \\ \hat{e}_2 \end{pmatrix}, \ldots, \begin{pmatrix} 0 \\ \hat{e}_m \end{pmatrix} \right\}$ which span the invariant space which can be defined as $\mathcal{E}$. These basis vectors should span the same invariant space as the rows of Q_L^T.

On the invariant manifold $\mathcal{M}_0$ it can be shown that the projection of the original dynamics onto the space $\mathcal{E}$ is identically zero. In mathematical terms, this projection onto the invariant space associated with the nonzero eigenvalues is given by

$$\begin{pmatrix} 0 & \hat{e}_1^T \\ 0 & \hat{e}_2^T \\ 0 & \vdots \\ 0 & \hat{e}_m^T \end{pmatrix} \begin{pmatrix} \epsilon f(x,0,0) \\ g(x,h^0(x),0) \end{pmatrix} = 0 \tag{6.26}$$

as $g(x, h^0(x), 0) = 0$ by the definition of $h^0(x)$. Therefore, on $\mathcal{M}_0$, $Q_L^T F(x_0, y_0)$ from Equation (6.19) is zero and the conditions specified in the Maas and Pope (1992) algorithm are satisfied.

For points (x_0, y_0) not on $\mathcal{M}_0$, it should be shown that the quantity $Q_L^T F(x_0, y_0)$ is not zero. If it is possible for this quantity to be zero for $(x_0, y_0) \notin \mathcal{M}_0$, the manifold could be identified incorrectly. The linearization of (6.9) about $(x_0, y_0) \notin \mathcal{M}_0$ takes the following form when $\epsilon = 0$

$$\begin{aligned}
\begin{pmatrix} \dot{\delta x} \\ \dot{\delta y} \end{pmatrix} &= \begin{pmatrix} \epsilon \frac{\partial f}{\partial x}(x, y, \epsilon) & \epsilon \frac{\partial f}{\partial y}(x, y, \epsilon) \\ \frac{\partial g}{\partial x}(x_0, y_0, \epsilon) & \frac{\partial g}{\partial y}(x_0, y_0, \epsilon) \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} \\
&= \begin{pmatrix} 0 & 0 \\ \frac{\partial g}{\partial x}(x_0, y_0, 0) & \frac{\partial g}{\partial y}(x_0, y_0, 0) \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix}
\end{aligned} \tag{6.27}$$

As long as $\frac{\partial g}{\partial y}(x_0, y_0, 0)$ is nonsingular, this matrix has exactly l eigenvalues which are zero and the invariant space associated with the these eigenvalues is spanned by the basis vectors $\left\{ \begin{pmatrix} e_1 \\ 0 \end{pmatrix}, \begin{pmatrix} e_2 \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} e_l \\ 0 \end{pmatrix} \right\}$. The orthogonal complement to this space (that space which refers to nonzero eigenvalues) will be spanned by the basis vectors $\left\{ \begin{pmatrix} 0 \\ \hat{e}_1 \end{pmatrix}, \begin{pmatrix} 0 \\ \hat{e}_2 \end{pmatrix}, \ldots, \begin{pmatrix} 0 \\ \hat{e}_m \end{pmatrix} \right\}$. Off the manifold, it can be shown that the expression $Q_L^T F(x_0, y_0)$ from Equation (6.19) is non-zero. This is because for $(x_0, y_0) \notin \mathcal{M}_0$, $g(x_0, y_0, 0) \neq 0$ by definition. Since this is true

$$\begin{pmatrix} 0 & \hat{e}_1^T \\ 0 & \hat{e}_2^T \\ 0 & \vdots \\ 0 & \hat{e}_m^T \end{pmatrix} \begin{pmatrix} \epsilon f(x_0, y_0, 0) \\ g(x_0, y_0, 0) \end{pmatrix} \neq 0 \tag{6.28}$$

6.3.2 Finite time-scale separation

In the case where the time-scale separation between the variables is finite, the parameter ϵ takes a small, non-zero value. In this case, the slow manifold will no longer be $\mathcal{M}_0$. In addition, since ϵ is finite the Jacobian no longer has the special form of Equation (6.25) which makes the analysis easy. However when ϵ is sufficiently small, a manifold of slow dynamics which is close to $\mathcal{M}_0$ still exists. It is shown by Fenichel (1979) that there exists a manifold which is called $\mathcal{M}_\epsilon$.

Theorem 6.3.1 (Fenichel's Invariant Manifold Theorem) *If $\epsilon > 0$ and sufficiently small, there exists a manifold $\mathcal{M}_\epsilon$ which lies within $\mathcal{O}(\epsilon)$ of $\mathcal{M}_0$ and is diffeomorphic to $\mathcal{M}_0$. In addition, $\mathcal{M}_\epsilon$ is invariant to the flow of (6.9).*

Now in the fast scaling, the manifold $\mathcal{M}_\epsilon$ will not need to be a manifold of stationary points as was the case for $\mathcal{M}_0$. Figure 6.1 illustrates the behavior of trajectories in the neighborhood of the slow manifold in these two different cases.

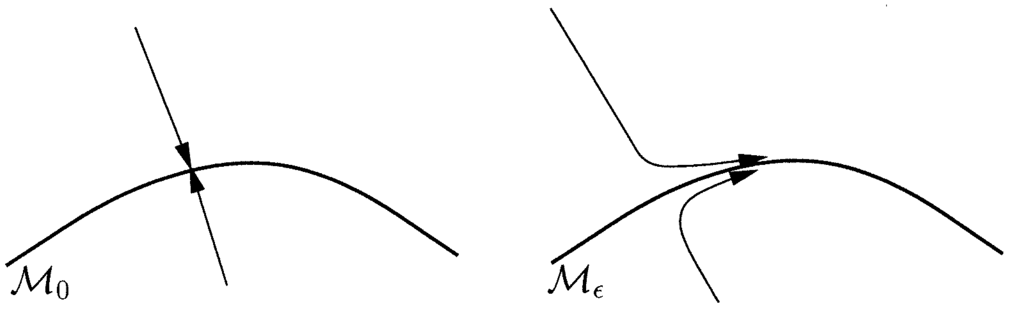


Figure 6.1: Behavior of trajectories in the neighborhood near the slow manifold for $\mathcal{M}_0$ and $\mathcal{M}_\epsilon$.

Since the slow manifold is $\mathcal{M}_\epsilon$, it should be shown that the algorithm proposed by Mass and Pope identifies $\mathcal{M}_\epsilon$ from the equations given in (6.9). As the manifold $\mathcal{M}_\epsilon$ is diffeomorphic to $\mathcal{M}_0$, it is expected that properties which hold for $\mathcal{M}_0$ should also hold for $\mathcal{M}_\epsilon$. In order to show this rigorously, the Fenichel Normal Form will be derived (Jones, 1994) which converts equations from the general form of (6.9) to one in which the variables are "decoupled" in some sense. Working with the coordinates in the Fenichel Normal Form, it can be shown that the algorithm of Maas and Pope properly identifies the slow manifold.

First, it will be assumed without loss of generality that $h^0(x) = 0$ for all points $x \in K$. The set K is simply the compact domain where $(x, h^0(x)) \in \mathcal{M}_0$. If $h^0(x) \neq 0$ in the original coordinates, the change of coordinates from y to $\tilde{y} = y - h^0(x)$ gives the desired property. Using this assumption, Equation (6.9) can be converted to the following form on $\mathcal{M}_0$

$$\begin{aligned} \dot{x} &= \epsilon f(x,y,\epsilon) \\ \dot{y} &= L(x)y + G(x,y,\epsilon) \end{aligned} \tag{6.29}$$

where $L(x) = \frac{\partial g}{\partial y}(x, h^0(x), \epsilon)$ and $|G(x,y,\epsilon)| \leq \gamma(|y| + |\epsilon|)$ since G is higher order in y and ϵ.

Using linear algebra, a transformation can be found such that y is transformed into coordinates (a, b)

$$\begin{aligned} \dot{x} &= \epsilon f(x,y,\epsilon) \\ \dot{a} &= A(x)a + G_1(x,a,b,\epsilon) \\ \dot{b} &= B(x)b + G_2(x,a,b,\epsilon) \end{aligned} \tag{6.30}$$

where the eigenvalues of $A(x)$ are strictly positive and the eigenvalues of $B(x)$ are strictly negative. This is true since the eigenvalues of $L(x)$ are bounded away from zero by the hyperbolic manifold assumption. It will also be assumed that $a \in \Re^{m_a}$ and $b \in \Re^{m_b}$ where $m_a + m_b = m$.

When $\epsilon = 0$, each point $(x, h^0(x)) \in \mathcal{M}_0$ has a stable and unstable manifold associated with it. The m_b-dimensional stable manifold $W^s(\mathcal{M}_0)$ is defined by the condition $a = 0$, while the m_a-dimensional unstable manifold $W^u(\mathcal{M}_0)$ is defined by $b = 0$. When $\epsilon > 0$, it turns out that these stable and unstable manifolds persist and are simply perturbations of the manifolds $W^s(\mathcal{M}_0)$ and $W^u(\mathcal{M}_0)$.

Theorem 6.3.2 (Fenichel Invariant Manifold Theorem 2) *If $\epsilon > 0$ and sufficiently small, there exist manifolds $W^s(\mathcal{M}_\epsilon)$ and $W^u(\mathcal{M}_\epsilon)$ which are $\mathcal{O}(\epsilon)$ perturbations and are diffeomorphic to $W^s(\mathcal{M}_0)$ and $W^u(\mathcal{M}_0)$.*

In order to prove this theorem, graphs which define $W^s(\mathcal{M}_\epsilon)$ and $W^u(\mathcal{M}_\epsilon)$ will be shown to exist.

Theorem 6.3.3 (Jones Invariant Manifold Graph Theorem) *If $\epsilon > 0$ is sufficiently small, for some $\Delta > 0$*

1. *there is a function $a = h_s(x, b, \epsilon)$ defined for $x \in K$ and $|b| \leq \Delta$ such that the graph*
$$W^s(\mathcal{M}_\epsilon) = \{(x, a, b) : a = h_s(x, b, \epsilon)\} \tag{6.31}$$
is locally invariant under the dynamics defined by (6.30).

2. *there is a function $b = h_u(x, a, \epsilon)$ defined for $x \in K$ and $|a| \leq \Delta$ such that the graph*
$$W^u(\mathcal{M}_\epsilon) = \{(x, a, b) : b = h_u(x, a, \epsilon)\} \tag{6.32}$$
is locally invariant under the dynamics defined by (6.30).

A proof of this theorem is found in (Jones, 1994).

Since a graph exists which defines the manifolds $W^s(\mathcal{M}_\epsilon)$ and $W^u(\mathcal{M}_\epsilon)$, the coordinates can be transformed to simplify the dynamics. In the first step, the coordinates are transformed to

$$\begin{aligned} x_1 &= x \\ a_1 &= a - h_s(x, b, \epsilon) \\ b_1 &= b \end{aligned} \tag{6.33}$$

which causes the manifold $W^s(\mathcal{M}_\epsilon)$ to be defined by $a_1 = 0$. The coordinates are then transformed again to

$$\begin{aligned} x_2 &= x_1 \\ a_2 &= a_1 \\ b_2 &= b_1 - h_u(x_1, a_1 + h_s(x_1, b_1, \epsilon), \epsilon) \end{aligned} \tag{6.34}$$

and now the surface $b_2 = 0$ defines the manifold $W^u(\mathcal{M}_\epsilon)$.

Since the stable and unstable manifolds are invariant under the dynamics defined by (6.30), the surfaces $a_2 = 0$ and $b_2 = 0$ must be invariant. This means that $a_2 = 0$ implies $\dot{a}_2 = 0$ and $b_2 = 0$ implies $\dot{b}_2 = 0$ which means that the equations describing the dynamics in these coordinates take the form

$$\begin{aligned} \dot{x}_2 &= \epsilon F_2(x_2, a_2, b_2, \epsilon) \\ \dot{a}_2 &= \Lambda(x_2, a_2, b_2, \epsilon) a_2 \\ \dot{b}_2 &= \Gamma(x_2, a_2, b_2, \epsilon) b_2 \end{aligned} \tag{6.35}$$

locally, where $F_2(x_2, a_2, b_2, \epsilon) = f(x, a - h_s(x, b, \epsilon), b - h_u(x, a, \epsilon))$ and $\Lambda(x_2, a_2, b_2, \epsilon)$ and $\Gamma(x_2, a_2, b_2, \epsilon)$ are matrices. Additionally, System (6.35) should be equivalent to (6.30) when $\epsilon = 0$. For this reason $\Lambda(x_2, 0, 0, 0) = A(x)$ and $\Gamma(x_2, 0, 0, 0) = B(x)$.

With this transformation, the coordinates defining the stable and unstable manifold of a given point $v_0 \in \mathcal{M}_0$ have been "straightened." In order to decouple the slow directions, theory which is known as Fenichel Fibering is needed. Since it has been shown that the manifolds $W^s(\mathcal{M}_0)$ and $W^s(\mathcal{M}_0)$ are persistent when the parameter ϵ is finite, the next question is whether the manifolds associated with a single point $v_0 \in \mathcal{M}_0$ also perturb in a similar fashion when ϵ is small. Since individual points $v_\epsilon \in \mathcal{M}_\epsilon$ are no longer fixed points, it is not clear that these structures should remain after the perturbation.

It turns out that these structures will still remain, although the dynamic nature of the system in this region is quite different than in the case $\epsilon = 0$.

Theorem 6.3.4 (Fenichel Invariant Manifold Theorem 3) *For all $v_\epsilon \in \mathcal{M}_\epsilon$ there exist manifolds*

$$W^s(v_\epsilon) \subset W^s(\mathcal{M}_\epsilon) \tag{6.36}$$

and

$$W^u(v_\epsilon) \subset W^u(\mathcal{M}_\epsilon) \tag{6.37}$$

of appropriate dimension which are $\mathcal{O}(\epsilon)$ perturbations and are diffeomorphic to $W^s(v_0)$ and $W^u(v_0)$ respectively. In addition, these manifolds are invariant as long as the tra-

jectory does not escape the local neighborhood of interest.

A sketch of the stable manifolds $W^s(v_0)$ and $W^s(v_\epsilon)$ is given in Figure 6.2.

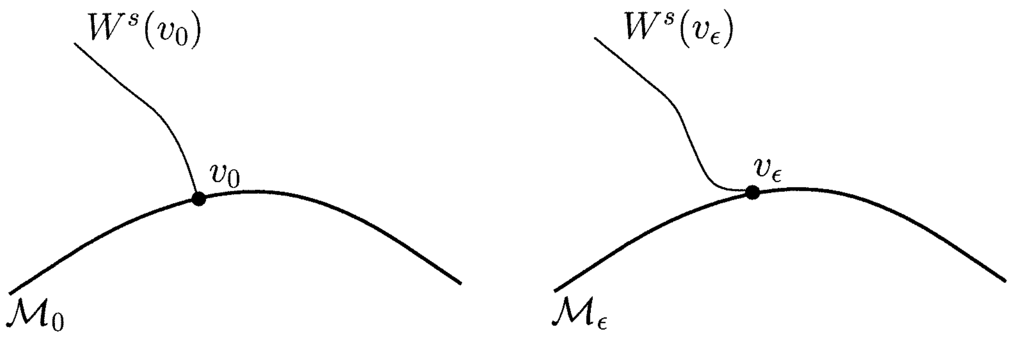


Figure 6.2: A sketch of $W^s(v_0)$ and $W^s(v_\epsilon)$.

Once again this theorem is shown by determining a graph function which describes the manifold associated with a single given point on $\mathcal{M}_\epsilon$.

Theorem 6.3.5 (Jones (1994)) *If $\epsilon > 0$ is sufficiently small, then*

1. *in $a_2 = 0$ (which is $\subset W^s(\mathcal{M}_\epsilon)$) there exists, for each $v_\epsilon = (\hat{x}, \epsilon) \in \mathcal{M}_\epsilon$, a function $x = h_s^v(b)$ defined for $|b| \leq \Delta$ such that the graphs*

$$W^s(v_\epsilon) = \{(x, 0, b, \epsilon) : x = h_s^v(b)\} \tag{6.38}$$

form a locally invariant family.

2. *in $b_2 = 0$ (which is $\subset W^u(\mathcal{M}_\epsilon)$) there exists, for each $v_\epsilon = (\hat{x}, \epsilon) \in \mathcal{M}_\epsilon$, a function $x = h_u^v(a)$ defined for $|a| \leq \Delta$ such that the graphs*

$$W^u(v_\epsilon) = \{(x, a, 0, \epsilon) : x = h_u^v(a)\} \tag{6.39}$$

form a locally invariant family.

These graphs, $h_s^v(b)$ and $h_u^v(a)$, define the stable and unstable manifolds associated with a given point $v_\epsilon \in \mathcal{M}_\epsilon$. In fact, the graphs define mappings from (v_ϵ, b) to $(h_s^v(b), b)$ or from (v_ϵ, a) to $(h_s^v(a), a)$ respectively. By taking the inverse of these mapping the base point of the fibers which are on $\mathcal{M}_\epsilon$ can be found. This inverse mapping takes a fiber in $W^s(\mathcal{M}_\epsilon)$ or $W^u(\mathcal{M}_\epsilon)$ to its base on $\mathcal{M}_\epsilon$.

These inverses can be defined as $\pi^- : (x, b, \epsilon) \to \hat{x} \in \mathcal{M}_\epsilon$ and $\pi^+ : (x, a, \epsilon) \to \hat{x} \in \mathcal{M}_\epsilon$. By using these inverses as a coordinate transformation, the fibers inside $W^s(\mathcal{M}_\epsilon)$ and $W^u(\mathcal{M}_\epsilon)$ can also be straightened. This is done first by straightening the stable manifold

$$\begin{aligned} x_3 &= \pi^-(x_2, b_2, \epsilon) \\ a_3 &= a_2 \\ b_3 &= b_2 \end{aligned} \tag{6.40}$$

and then by straightening the fibers of the unstable manifold

$$\begin{aligned} x_4 &= \pi^+(x_3, a_3, \epsilon) \\ a_4 &= a_3 \\ b_4 &= b_3. \end{aligned} \tag{6.41}$$

By making these transformation, the slow flow becomes decoupled from the flow on the manifolds $W^s(\mathcal{M}_\epsilon)$ and $W^u(\mathcal{M}_\epsilon)$. If either $a = 0$ or $b = 0$ then what was $f(x, a, b, \epsilon)$ is only a function of x and ϵ in these new coordinates. For this reason, the new form of the equations in these transformed coordinates is given by

$$\begin{aligned} \dot{x}_4 &= \epsilon\{h(x_4, \epsilon) + H(x_4, a_4, b_4, \epsilon)\} \\ \dot{a}_4 &= \Lambda(x_4, a_4, b_4, \epsilon) a_4 \\ \dot{b}_4 &= \Gamma(x_4, a_4, b_4, \epsilon) b_4 \end{aligned} \tag{6.42}$$

where $H(x_4, a_4, b_4, \epsilon)$ is a bilinear function of a_4 and b_4. This change of coordinates will be valid only in a local neighborhood of the manifold, which can be defined as the set $D = \{(x_4, a_4, b_4, \epsilon) : |a_4| \leq \Delta, |b_4| \leq \Delta, x_4 \in K, \epsilon \in [0, \epsilon_0]\}$.

Since this change of coordinates is smooth, the algorithm of Maas and Pope can be validated in these coordinates. By linearizing the dynamics about a point $(x_{\mathcal{M}}, 0, 0)$, $x_{\mathcal{M}} \in K$, the above equations take the form

$$\begin{aligned} \dot{\delta x} &= \epsilon \frac{\partial h}{\partial x}(x_{\mathcal{M}}, \epsilon) \\ \dot{\delta a} &= \Lambda(x_{\mathcal{M}}, 0, 0, \epsilon)\delta a \\ \dot{\delta b} &= \Gamma(x_{\mathcal{M}}, 0, 0, \epsilon)\delta b. \end{aligned} \tag{6.43}$$

The Jacobian of this linearization is

$$J_\epsilon = \begin{pmatrix} \epsilon \frac{\partial h}{\partial x}(x_{\mathcal{M}}, \epsilon) & 0 & 0 \\ 0 & \Lambda(x_{\mathcal{M}}, 0, 0, \epsilon) & 0 \\ 0 & 0 & \Gamma(x_{\mathcal{M}}, 0, 0, \epsilon) \end{pmatrix} \tag{6.44}$$

and the eigenvalues of J_ϵ lie in two distinct groups. The first group of eigenvalues consists of m eigenvalues which have a real part with absolute value of magnitude $\mathcal{O}(\epsilon)$. The eigenvectors associated with these eigenvalues are contained in the space spanned by the vectors $\left\{ \begin{pmatrix} e_1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} e_2 \\ 0 \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} e_l \\ 0 \\ 0 \end{pmatrix} \right\}$ where the vectors e_i form a basis of the space $\Re^l$. Those eigenvalues which have real part which is bounded away from the origin have eigenvectors contained in the space spanned by the vectors $\left\{ \begin{pmatrix} 0 \\ \hat{e}_1 \\ 0 \end{pmatrix}, \ldots \begin{pmatrix} 0 \\ \hat{e}_{m_a} \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \check{e}_1 \end{pmatrix} \ldots, \begin{pmatrix} 0 \\ 0 \\ \check{e}_{m_b} \end{pmatrix} \right\}$.

The dynamics of system (6.42) evaluated for $(x_{\mathcal{M}}, 0, 0) \in \mathcal{M}_\epsilon$ are described by

$$\begin{aligned} \dot{x} &= \epsilon h(x_{\mathcal{M}}, \epsilon) \\ \dot{a} &= 0 \\ \dot{b} &= 0. \end{aligned} \tag{6.45}$$

Once again it is easily seen that the projection of the dynamics onto the space of the fast eigenvalues is identically zero on the manifold.

$$\begin{pmatrix} 0 & \hat{e}_1^T & 0 \\ & \vdots & \\ 0 & \hat{e}_{m_a}^T & 0 \\ 0 & 0 & \check{e}_1^T \\ & \vdots & \\ 0 & 0 & \check{e}_{m_b}^T \end{pmatrix} \begin{pmatrix} \epsilon h(x_{\mathcal{M}}, \epsilon) \\ 0 \\ 0 \end{pmatrix} = 0. \tag{6.46}$$

Since the projection of the dynamics onto the eigenvectors associated with the fast direction is identically zero, the Maas and Pope algorithm should again give the proper result when on the manifold $\mathcal{M}_\epsilon$.

To show that the projection of the dynamics onto the eigenspace associated with the fast eigenvalues is non-zero for a point $w \notin \mathcal{M}_\epsilon$, a special case will be considered. Assume that the fast dynamics of the system are stable. In this case the dynamics in the set D near $\mathcal{M}_\epsilon$ are

$$\begin{aligned} \dot{x}_4 &= \epsilon h(x_4, \epsilon) \\ \dot{b}_4 &= \Gamma(x_4, b_4, \epsilon) b_4 \end{aligned} \tag{6.47}$$

and the Jacobian of the dynamics linearized about the point $w = (x, b)$ is

$$J = \begin{pmatrix} \epsilon \frac{\partial h}{\partial x_4}(x, \epsilon) & 0 \\ \frac{\partial \Gamma}{\partial x_4}(x, b, \epsilon) & \frac{\partial \Gamma}{\partial b_4}(x, b, \epsilon) b + \Gamma(x, b, \epsilon) \end{pmatrix} \tag{6.48}$$

The eigenspace associated with eigenvalues which have $\mathcal{O}(\epsilon)$ real part is spanned by the vectors $\left\{ \begin{pmatrix} e_1 \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} e_l \\ 0 \end{pmatrix} \right\}$ where the vectors e_i form a basis of the space $\Re^l$ and the eigenspace associated with eigenvalues with are $\mathcal{O}(1)$ is spanned by the orthogonal vectors $\left\{ \begin{pmatrix} 0 \\ \hat{e}_1 \end{pmatrix}, \ldots, \begin{pmatrix} 0 \\ \hat{e}_{m_b} \end{pmatrix} \right\}$ where the vectors $\hat{e}_i$ form a basis of $\Re^{m_b}$.

The projection of the dynamics onto the space associated with fast eigenvalues is given by

$$\begin{pmatrix} 0 & \hat{e}_1^T \\ \vdots & \\ 0 & \hat{e}_{m_a}^T \end{pmatrix} \begin{pmatrix} \epsilon h(x,\epsilon) \\ \Gamma(x,b,\epsilon)b \end{pmatrix} \neq 0. \tag{6.49}$$

This projection is non-zero because $b \neq 0$ (by definition of the $(x,b) \notin \mathcal{M}_\epsilon$) and by the fact that Γ is nonsingular (normally hyperbolic manifold assumption). It can also easily be shown that the when the fast dynamics are strictly unstable that the projection of the dynamics onto the space associated with the fast eigenvalues is nonzero using the same method.

In the case where there are both stable and unstable fast dynamics, it is expected that the results are similar. Showing this rigorously is a bit messy, since the additional terms associated with the bilinear term H make the Jacobian non-diagonal for points not on $\mathcal{M}_\epsilon$.

6.4 Example: Binary distillation

Distillation is used widely throughout the chemical and petroleum industries for separation. There has been a great deal of work directed at understanding the dynamics and improving control of distillation processes because of its importance in these industries. Modeling of the process in a tray-by-tray fashion is straight-forward by considering component balances. In the simplest case of a binary mixture with 100% tray efficiency, constant molar flow, and constant molar holdups tray i is modeled as

$$M_i \dot{x}_i = Lx_{i-1} + Vy_{i+1} - Lx_i - Vy_i \tag{6.50}$$

where x_i is the liquid composition on tray i, y_i is the vapor composition on tray i, L is the liquid molar flow rate, V is the vapor flow rate, and M_i is the molar holdup of tray i. By assuming the vapor liquid equilibrium is governed by constant relative volatility (α), y_i is given by

$$y_i = k(x_i) = \frac{\alpha x_i}{(1 + (\alpha - 1)x_i)}. \tag{6.51}$$

These set of differential algebraic equations can be easily integrated, and early computational studies of the transient behavior of distillation columns are presented by Rosenbrock (1957).

While these detailed models have been available for a long time, there have been numerous attempts to quantify the dynamics behavior of these differential algebraic equations and to develop a reduced-order model which accurately describes the dynamics. It has been well-known for a long time that the dynamics of a distillation column are dominated by a single large time constant which can be estimated by considering the column as a giant "mixing tank" (Davidson, 1956; Moczek, Otto and Williams, 1965; Wahl and Harriot, 1970; Skogestad and Morari, 1987). Later, the MIMO dynamics of the linearized input/output system were studied and a second smaller time constant was discovered which has important implications for control purposes (Skogestad and Morari, 1988). Input directionality is important when determining this second time constant, and these ideas are extended in the work of Sågfors and Waller (1995). Other recent approaches to this problem utilize nonlinear methods. Nonlinear wave theory is utilized in the work of Hwang (1991) and Hwang (1995). A singular perturbation theory approach, which will be described later in this section, is applied by Lévine and Rouchon (1991).

The example which will be studied is a binary distillation column with 40 ideal trays, a reboiler, and a total condenser. Constant molar overflow and constant relative volatility ($\alpha = 1.5$) are assumed. The feed is liquid with a light component composition $z_f = 0.5$. The liquid molar holdup of each tray is assumed to be equal and is defined as $M_i = M = 0.5$. The distillate flow rate is $D/F = 0.5$ and the liquid reflux flow rate is $L/F = 2.702$ at the operating point of interest. The composition of the distillate output for these operating point is $x_d = x_1 = 0.99$. This column is identical to Column A from Skogestad and Morari (1988).

In order to study the transient dynamics around the operating point, the system is initialized at another steady-state where $L/F = 9.0$ and $D/F = 0.2$. Since the reflux is higher at this operating point, the distillate has a higher purity ($x_d = 0.999$). At time $t = 0$, the flows of the distillate and liquid reflux are changed to those of the operating point of interest ($L/F = 2.702$, $D/F = 0.5$). From the simulation results of the dynamic transition to the steady-state, the behavior seems to be show evidence of a time-scale separation. The response of the distillate, after the first 50 minute transient period, appears to be an output from a first order system (Figure 6.3). To

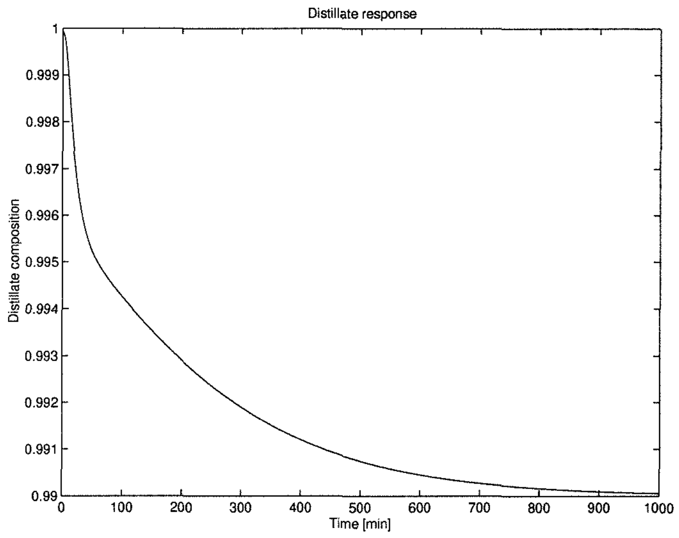


Figure 6.3: Time response of the distillate composition for $L/F : 9.0 \rightarrow 2.702$, $D/F : 0.2 \rightarrow 0.5$ at $t = 0$.

further illustrate the dynamics of the simulated system, a snapshot of the composition profile along the length of the column is given in Figure 6.4. The composition profile is given at times $[0, 50, 100, 200, 300, 400, 500, 600]$. As time progresses, the composition moves from the initial high purity profile to lower purity.

Since the output seems to suggest that the system exhibits behavior consistent with a one-dimensional slow manifold, three methods will be presented for identifying the one-dimensional "slow" manifold and the results will be compared withe the simulation results. In the first method, modal decomposition will be applied to a linearized model of the column dynamics. A linear model of the system can be built

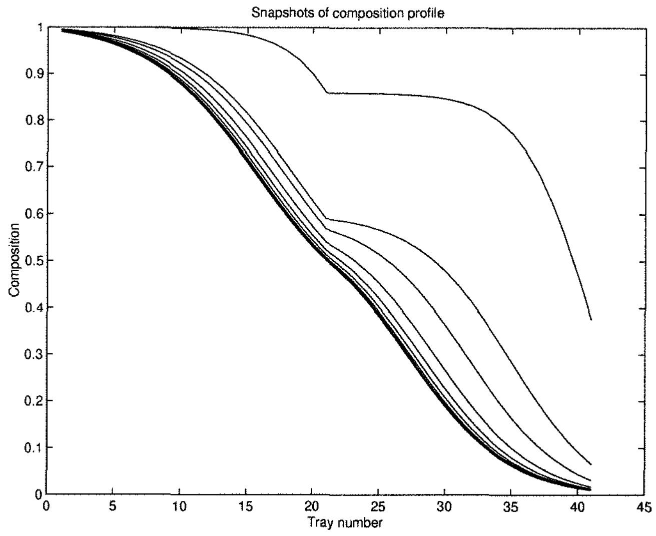


Figure 6.4: Snapshots of the column composition profile at times $[0, 50, 100, 200, 300, 400, 500, 600]$.

at the operating point of Column A ($L/F = 2.702$, $D/F = 0.5$)

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= [1, 0, \ldots, 0]x \end{aligned} \tag{6.52}$$

where the C matrix takes this special form since the output is the distillate, x_1. The variable x, u, and y of this system are deviation variables about the steady-state. By performing a modal decomposition and reducing the system such that only a single mode is kept, the following model is found:

$$\begin{aligned} \dot{\bar{x}}_1 &= \lambda_1 \bar{x}_1 + B_T^\star u \\ y &= [1, 0, \ldots, 0] t_1 \bar{x}_1 \end{aligned} \tag{6.53}$$

where λ_1 is the eigenvalue of smallest magnitude and $T = [t_1, t_2, \cdots, t_{41}]$ is a matrix consisting of the eigenvectors t_i of A sorted such that t_1 is the eigenvector associated with the eigenvalue λ_1 (for ideal binary distillation, all the eigenvalues of A are negative real and distinct). In the neighborhood of the operating point, System 6.53 should be a good approximation to the dynamics of the full system since λ_1 is more than an order of magnitude smaller than the other eigenvalues.

From the above reduced model,

$$y = t_{11}\bar{x}_1 \tag{6.54}$$

where t_{11} is the first component of the eigenvector associated with the dominant eigenvalue. By assuming the remaining modes are at equilibrium ($\bar{x}_2 = \cdots = \bar{x}_{41} = 0$), the original state x can be found by the transformation

$$t_1\bar{x}_1 = x. \tag{6.55}$$

Using Equations (6.54) and (6.55), the state profile from the one-dimensional reduced model can be found from the output y.

$$x = \frac{y}{t_{11}}t_1. \tag{6.56}$$

This method is used to determine the linear approximation of the one-dimensional slow manifold of the distillation system. As can be seen in Figure 6.5, this approx-

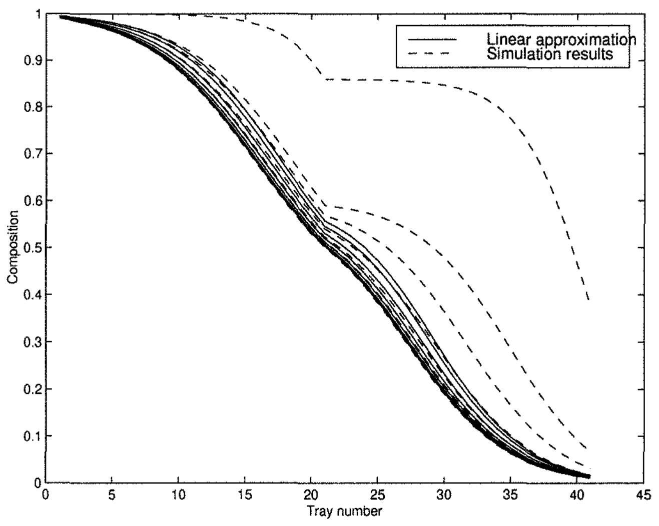


Figure 6.5: The linear approximation to the one-dimensional slow manifold

imation does a good job of finding the low-dimensional manifold when the column profile is close to the steady-state. This is expected, since a linear approximation is valid in a small neighborhood of the steady-state. However, for short times after the step change this method is unable to capture the features of the composition profile. This is due to the fact that the modal decomposition method only utilizes a single linear model.

In an attempt to find a better approximation of the slow manifold, nonlinear methods will be used. The second method which will be described here involve transforming the original equations describing the distillation column into the standard form for singularly perturbed systems (System 6.7). This physically motivated transformation is given in Lévine and Rouchon (1991) and will be used here to compute the slow manifold associated with this assumption.

The dynamic description of a section of N trays of a distillation column with no feed is

$$\begin{aligned}
M\dot{x}_1 &= Lx_0 + Vy_2 - Lx_1 - Vy_1 \\
&\vdots \\
M\dot{x}_{j-1} &= Lx_{j-2} + Vy_j - Lx_{j-1} - Vy_{j-1} \\
M\dot{x}_j &= Lx_{j-1} + Vy_{j+1} - Lx_j - Vy_j \\
M\dot{x}_{j+1} &= Lx_j + Vy_{j+2} - Lx_{j+1} - Vy_{j+1} \\
&\vdots \\
M\dot{x}_N &= Lx_{N-1} + Vy_{N+1} - Lx_N - Vy_N.
\end{aligned}$$

To put the original system equations in the standard form of singularly perturbed systems, the following change of coordinates is made (Lévine and Rouchon, 1991)

$$\begin{pmatrix} x_1 \\ \vdots \\ x_{j-1} \\ x_j \\ x_{j+1} \\ \vdots \\ x_N \end{pmatrix} \rightarrow \begin{pmatrix} x_1^f = x_1 \\ \vdots \\ x_{j-1}^f = x_{j-1} \\ x^s = \sum_{i=1}^N x_i/N \\ x_{j+1}^f = x_{j+1} \\ \vdots \\ x_N^f = x_N \end{pmatrix} \tag{6.57}$$

where the section has a total of N trays.

By defining the holdup of the entire section as $\bar{M} = NM$, this transformation results in the following description

$$\begin{aligned} \frac{1}{N}\bar{M}\dot{x}_1^f &= Lx_0 + Vy_2^f - Lx_1^f - Vy_1^f \\ &\vdots \\ \frac{1}{N}\bar{M}\dot{x}_{j-1}^f &= Lx_{j-2}^f + Vk(x^s - \frac{1}{N}\sum_{i\neq j} x_i^f) - Lx_{j-1}^f - Vy_{j-1}^f \\ \bar{M}\dot{x}^s &= Lx_0 + Vy_{N+1} - Lx_N^f - Vy_1^f \\ \frac{1}{N}\bar{M}\dot{x}_{j+1}^f &= L(x^s - \frac{1}{N}\sum_{i\neq j} x_i^f) + Vy_{j+2}^f - Lx_{j+1}^f - Vy_{j+1}^f \\ &\vdots \\ \frac{1}{N}\bar{M}\dot{x}_N^f &= Lx_{N-1}^f + Vy_{N+1} - Lx_N^f - Vy_N^f \end{aligned}$$

Note that these equations are in the standard form for a singularly perturbed system if the factor $\frac{1}{N}$ is considered as the small parameter (ϵ in the previous notation).

If it is assumed that the holdup of the entire column is large compared to the holdup on a single tray ($\frac{1}{N} \rightarrow 0$), the description of the slow system is given by

$$\begin{aligned} 0 &= Lx_0 + Vy_2^f - Lx_1^f - Vy_1^f \\ &\vdots \end{aligned}$$

$$\begin{aligned}
0 &= Lx^f_{j-2} + Vy^s - Lx^f_{j-1} - Vy^f_{j-1} \\
\bar{M}\dot{x}^s &= Lx_0 + Vy_{N+1} - Lx^f_N - Vy^f_1 \\
0 &= Lx^s + Vy^f_{j+2} - Lx^f_{j+1} - Vy^f_{j+1} \\
&\vdots \\
0 &= Lx^f_{N-1} + Vy_{N+1} - Lx^f_N - Vy^f_N
\end{aligned}$$

The "reduced" description of the dynamics of the system involve a single ODE with $N-1$ algebraic constraints.

In order to compute the slow manifold for Column A, a term to account for the external feed Fx_F needs to be added to the right hand side of the equation describing the feed tray. By defining the transformation such that x_j is the feed tray, only the dynamic equation for x^s will include terms because of the feed. There are also some slight modifications to the above equations because the first equation now describes the condenser and the last equation describes the reboiler. Applying the transformation to the equations describing Column A, the set of algebraic constraints describing the slow manifold are:

$$\begin{aligned}
0 &= Vy^f_2 - Vx^f_1 \\
0 &= Lx^f_1 + Vy^f_3 - Lx^f_2 - Vy^f_2 \\
&\vdots \\
0 &= Lx^f_{j-2} + Vy^s - Lx^f_{j-1} - Vy^f_{j-1} \\
0 &= (L+F)x^s + Vy^f_{j+2} - (L+F)x^f_{j+1} - Vy^f_{j+1} \\
&\vdots \\
0 &= (L+F)x^f_{N-2} + Vy^f_N - (L+F)x^f_{N-1} - Vy^f_{N-1} \\
0 &= (L+F)x^f_{N-1} - (L+F-V)x^f_N - Vy^f_N
\end{aligned}$$

The $N-1$ algebraic constraints defined above involve N variables, so it is expected that the surface in the state-space which satisfies these algebraic conditions is a one-manifold.

By specifying the distillate composition x_1, the remaining tray compositions can be found by solving the above equations. Using the distillate composition from the simulation at the times corresponding to the snapshots, the slow manifold at this time can be found. The results of computing the slow manifold using this method are given in Figure 6.6. It appears that this assumption captures the main characteristics of the profile.

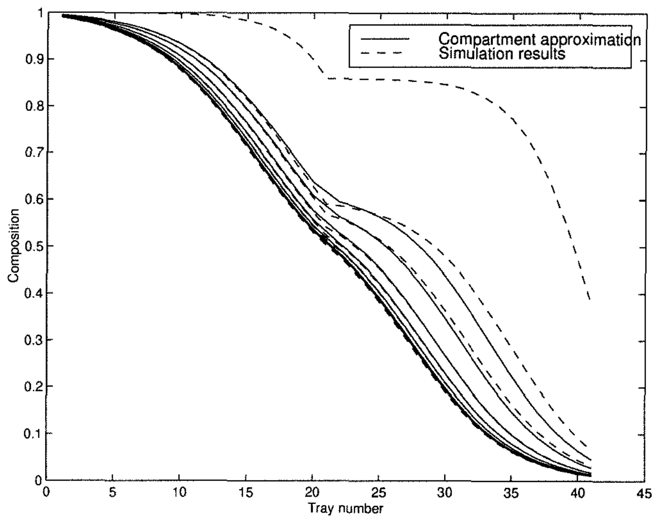


Figure 6.6: The compartment approximation to the one-dimensional slow manifold

While the approximation of Lévine and Rouchon (1991) appears to capture the main characteristics of the slow manifold, it is likely that this transformation is not the optimal one for determining the slow manifold. The computational algorithm of Maas and Pope (1992) is the third method which is applied to this example. The algorithm described in Section 6.2 is used to determine the slow manifold where x_1, the distillate composition, is used as a fixed parameter. Computation of a single column profile on the slow manifold with a specified distillate composition takes approximately 10 minutes on a Sun SPARCstation20. The results are given in Figure 6.7. It appears that this computational algorithm of Maas and Pope (1992) does the best job of capturing the slow manifold of the three methods presented.

In order to compare these methods, the norm of the difference between the simulation results and the profile given by the different methods of computing the slow

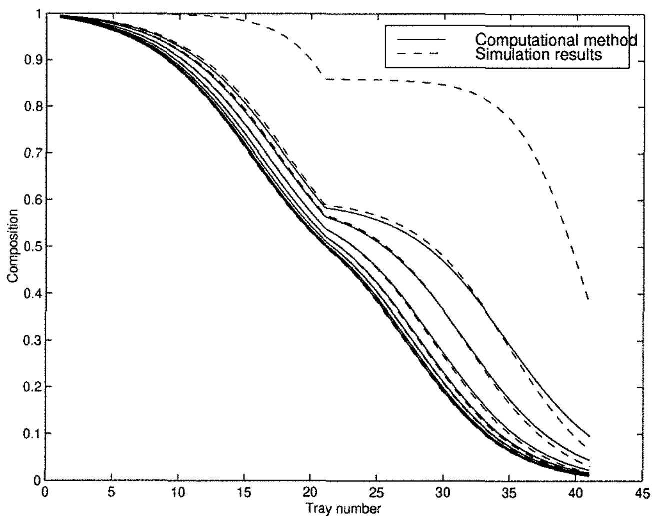


Figure 6.7: The results of the computational methods for identifying the one-dimensional slow manifold

manifold was found. These results are presented in Figure 6.8. From this comparison, it appears that the computational algorithm does the best job of determining the slow manifold.

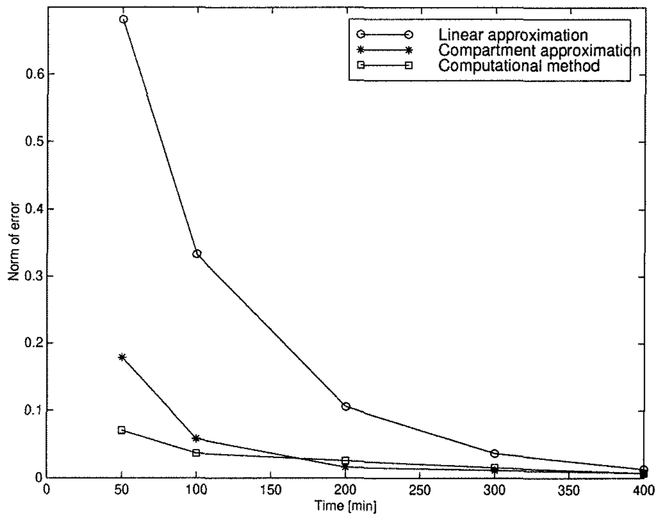


Figure 6.8: The norm of the error associated with each approximation method as a function of time

6.5 Forming reduced-order models

The computational algorithm outlined previously is only able to identify individual points on the manifold of slow dynamics. While this description of the manifold may provide some insight into the dynamics of the system, it would be preferable to develop a reduced model of the slow dynamics. With this reduced dynamical description, it may be possible to design an effective controller for the process using the reduced description.

Roussel and Fraser (1991) approached the problem of model reduction for systems exhibiting time-scale separations in a different manner than the method considered here. Their algorithm for model reduction computes a reduced model of the dynamics by performing certain algebraic manipulations on the full set of ODEs. Assume that the following ODEs, which are identical to Equation (6.11), describe the dynamics of the system.

$$\begin{aligned} \dot{z} &= F(z) \\ \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_n \end{pmatrix} &= \begin{pmatrix} F_1(z_1, z_2, \ldots, z_n) \\ F_2(z_1, z_2, \ldots, z_n) \\ \vdots \\ F_n(z_1, z_2, \ldots, z_n) \end{pmatrix}. \end{aligned} \tag{6.58}$$

It is also assumed by Roussel and Fraser (1991)that the system exhibits relaxation behavior that cascades through a hierarchy of smooth manifolds Σ such that $\Re^n \equiv \Sigma_n \supset \Sigma_{n-1} \supset \ldots \supset \Sigma_1$, where Σ_i is an i-dimensional invariant manifold.

Since Σ_{n-1} is an $n-1$ dimensional manifold in the state-space $\Re^n$, it is expected that a single nonlinear equation is able to describe this manifold in the state-space. In general, the constraint describing this manifold can be written by representing a single variable, z_n, as a nonlinear function of the remaining variables

$$z_n = z_n(z_1, z_2, \ldots, z_{n-1}). \tag{6.59}$$

By differentiating (6.59) with respect to time

$$\begin{aligned} \dot{z}_n &= \frac{\partial z_n}{\partial z_1}\dot{z}_1 + \ldots + \frac{\partial z_n}{\partial z_{n-1}}\dot{z}_{n-1} \\ F_n(z_1, z_2, \ldots, z_n) &= \frac{\partial z_n}{\partial z_1}F_1(z_1, \ldots, z_n) + \ldots + \frac{\partial z_n}{\partial z_{n-1}}F_{n-1}(z_1, \ldots, z_n) \quad (6.60) \end{aligned}$$

is found.

Equation (6.60) can be rearranged such that z_n appears on the left hand side and partial derivatives of z_n appear on the right hand side. After the rearrangement, the following iterative scheme is defined

$$z_n^{i+1}(z_1, \ldots, z_{n-1}) = h(z_1, \ldots, z_{n-1}, \frac{\partial z_n^i}{\partial z_1}, \ldots, \frac{\partial z_n^i}{\partial z_{n-1}}) \quad (6.61)$$

Roussel and Fraser (1991) claim that it is expected that Equation (6.61) has a stable fixed point $z_n^\star(z_1, \ldots, z_{n-1})$ that attracts all smooth starting functions which are sufficiently close to Σ_{n-1} for stable dynamical systems (6.58).

The solution, $z_n^\star(z_1, \ldots, z_{n-1})$, describing the surface Σ_{n-1} can then be substituted into the original equations to arrive at a reduced description of the dynamics on the invariant manifold Σ_{n-1}.

$$\begin{aligned} \begin{pmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \vdots \\ \dot{z}_{n-1} \end{pmatrix} &= \begin{pmatrix} F_1(z_1, z_2, \ldots, z_n^\star(z_1, \ldots, z_{n-1})) \\ F_2(z_1, z_2, \ldots, z_n^\star(z_1, \ldots, z_{n-1})) \\ \vdots \\ F_{n-1}(z_1, z_2, \ldots, z_n^\star(z_1, \ldots, z_{n-1})) \end{pmatrix} \quad (6.62) \\ &= \begin{pmatrix} F_1^r(z_1, z_2, \ldots, z_{n-1}) \\ F_2^r(z_1, z_2, \ldots, z_{n-1}) \\ \vdots \\ F_{n-1}^r(z_1, z_2, \ldots, z_{n-1}) \end{pmatrix}. \quad (6.63) \end{aligned}$$

By repeating the process on the reduced system, the dimension of the description can be reduced by one with each iteration. While the algorithm of Roussel and Fraser (1991) is appealing because the final result is a reduced model of the system, the symbolic manipulations make this algorithm computationally intensive for systems of even moderate complexity and dimension n.

The Maas and Pope (1992) algorithm identifies the reduced manifold, so it should be possible to fit the equations that describe the manifold from the results of the algorithm. The output of the algorithm is a number of points on the n_r-dimensional slow manifold

$$\begin{array}{c} (z_1, z_2, \ldots, z_n)_1 \\ (z_1, z_2, \ldots, z_n)_2 \\ \vdots \\ (z_1, z_2, \ldots, z_n)_M. \end{array} \tag{6.64}$$

The points on the slow manifold should satisfy $n - n_r$ nonlinear constraints.

These nonlinear constraints can be estimated from the data by performing $n - n_r$ black-box identifications. The results of the black-box regressions can be defined as

$$\begin{aligned} z_{n_r+1} &= G_1(z_1, \ldots, z_{n_r}) \\ z_{n_r+2} &= G_2(z_1, \ldots, z_{n_r}) \\ &\vdots \\ z_n &= G_{n-n_r}(z_1, \ldots, z_{n_r}). \end{aligned} \tag{6.65}$$

Once again these equations defining the constraints can be substituted into the original equations

$$\begin{aligned} \dot{z}_1 &= F_1(z_1, \ldots, z_{n_r}, G_1(\mathbf{z}), \ldots, G_{n-n_r}(\mathbf{z})) \\ \dot{z}_2 &= F_2(z_1, \ldots, z_{n_r}, G_1(\mathbf{z}), \ldots, G_{n-n_r}(\mathbf{z})) \\ &\vdots \\ \dot{z}_{n_r} &= F_{n_r}(z_1, \ldots, z_{n_r}, G_1(\mathbf{z}), \ldots, G_{n-n_r}(\mathbf{z})) \end{aligned} \tag{6.66}$$

where $\mathbf{z} = [z_1, \ldots, z_{n_r}]$. By making this substitution, a reduced model of the dynamics is formed.

While this reduced model will not have any fast dynamics if the functions G_i are able to represent the slow manifold exactly, this black-box approximation may not exactly describe the manifold. If there is any mismatch between the manifold and the functions G_i, there may be mismatch between the slow dynamics of the original model and this reduced model and there may also be a mismatch at steady-state.

Possibly the best approach to determining a reduced model of the slow dynamics would be to use the Maas and Pope (1992) algorithm to determine an initial approximation of the slow manifold. This initial approximation could then be used in the algorithm of Roussel and Fraser (1991), and the slow manifold would converge after iterations to the actual manifold. To reduce the computational requirements of the algorithm of Roussel and Fraser (1991), a numerical approximation of these functions could be used. There is some initial work in this direction, but the results are not yet published (Davis, 1997).

6.6 Example: Two-phase CSTR

In this example, the dynamics of a two-phase chemical reactor will be studied. The reactor has a pure gas feed, A, and a pure liquid feed, B, which react to form C ($A+B \rightarrow C$). An illustration of the system is given in Figure 6.9. While components

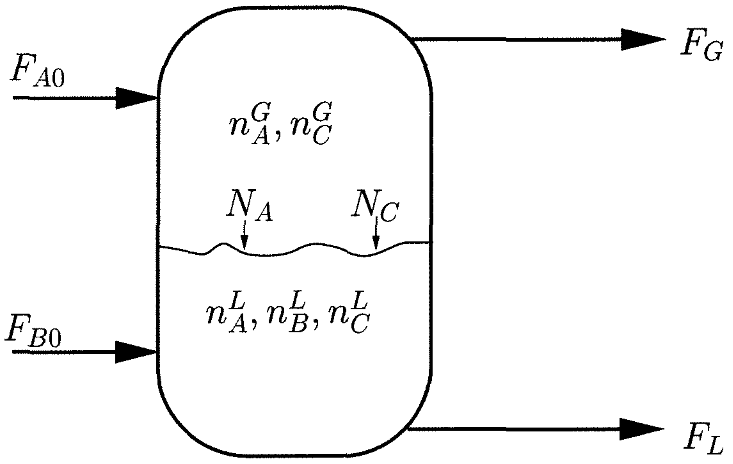


Figure 6.9: Gas-liquid phase chemical reactor

A and C exist in both the liquid and gas phase, component B remains purely in the liquid phase. The effect of mass transfer between the gas and liquid phase is also considered in the model. The equations describing the reactor are as follows:

$$\begin{aligned}
\frac{dn_A^G}{dt} &= F_{A0} - N_A - F_G y_A \\
\frac{dn_C^G}{dt} &= -N_C - F_G y_C \\
\frac{dn_A^L}{dt} &= N_A - kC_AC_BV_L - F_L x_A \\
\frac{dn_B^L}{dt} &= F_{B0} - kC_AC_BV_L - F_L x_B \\
\frac{dn_C^L}{dt} &= N_C + kC_AC_BV_L - F_L x_C
\end{aligned} \tag{6.67}$$

where

n_i Number of moles of component i in liquid (superscript L) or gas (superscript G) phase

x_i Mole fraction of liquid component i

y_i Mole fraction of gas component i

$V_L = (n_A^L + n_B^L + n_C^L)/\rho$ Liquid volume

$C_i = n_i^L/V_L$ Concentration in the liquid phase

$N_i = k_L(x_A^\star - x_A)$ Mass transfer from the gas to liquid phase of component i

$x_i^\star = Py_i/P_i^{\text{sat}}$

$P = (n_A^G + n_C^G)RT/(V - V_L)$ Reactor pressure

$k = k_0 e^{-E_a/RT}$ Reaction rate constant

with the parameters and variables at steady-state given in Tables 6.1 and 6.2.

Figures 6.10 and 6.11 give simulation results of the full dynamical system from arbitrary initial conditions ($[n_A^G, n_C^G, n_A^L, n_B^L, n_C^L] = [3000, 1000, 3300, 10500, 1500]$). Two time-scales appear to be present in the simulation results. An initial fast transient, and a slower time constant associated with the long time dynamics.

Variable	Description	Steady-state value
n_A^G	Molar gas holdup of A (mol)	3552
n_C^G	Molar gas holdup of C (mol)	1238
n_A^L	Molar liquid holdup of A (mol)	3061
n_B^L	Molar liquid holdup of B (mol)	10630
n_C^L	Molar liquid holdup of C (mol)	1310

Table 6.1: Reactor variables at steady-state

Parameter	Description	Value
F_{A0}	Inlet vapor flowrate (mol/s)	175
F_{B0}	Inlet liquid flowrate (mol/s)	250
F_G	Outlet gas flowrate (mol/s)	92.2
F_L	Outlet liquid flowrate (mol/s)	284.2
R	Gas constant $(J/mol\ K)$	8.314
E_a	Activation Energy (J/mol)	110000
k_0	Preexponential reaction factor $(m^3/mol\ s)$	10^{11}
ρ	Molar liquid phase density (mol/m^3)	15000
V	Reactor volume (m^3)	1.8
T	Reactor temperature (K)	341.5
k_L	Mass transfer coefficient (mol/s)	2500
P_A^{sat}	Saturation vapor pressure for A at T (Pa)	51.11×10^6
P_C^{sat}	Saturation vapor pressure for C at T (Pa)	56.49×10^6

Table 6.2: Reactor parameters

In Figure 6.11 the first 200 seconds of the simulation are plotted, and the effect of the time-scale separation can be clearly seen. Some very fast dynamics occur in the first 20 seconds. The behavior appears to be consistent with the mass-transfer of the system, since the liquid and gas holdups of components A and C adjust very quickly. After this initial transient, the behavior is consistent with a time-scale of approximately 100 seconds. Most likely, this behavior is associated with the chemical reaction.

One possible way to reduce this model is to use a physical understanding of the process as motivation. By assuming that the mass transfer dynamics are much faster than the other dynamics of the system. This means that the mass transfer term reaches steady-state much faster than the rest of the system. Therefore, in order to reduce the equations, it is assumed that $\dot{N}_A = \dot{N}_C = 0$. This is equivalent to

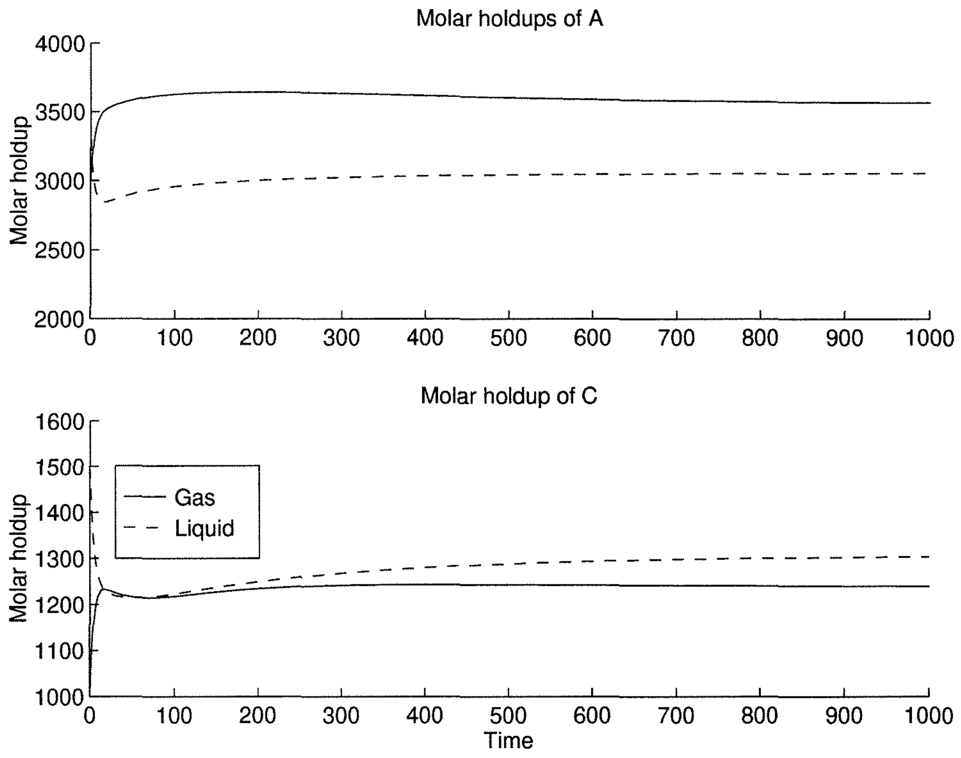


Figure 6.10: Simulation of the full set of equations for the CSTR (Top graph: n_A^G - solid line, n_A^L - dashed line. Bottom graph: n_C^G - solid line, n_C^L - dashed line.).

making the mass transfer dynamics reach equilibrium infinitely fast and provides two algebraic constraints that describe the slow manifold.

By transforming to a new set of variables which describe the overall holdup of each of the individual components in the reactor ($[n_A = n_A^G + n_A^L, n_B^L, n_C = n_C^G + n_C^L]$), the fast mass transfer dynamics (N_A and N_C) no longer appear. The new set of differential algebraic equations describing the reduced dynamics for the physically based model reduction scheme become

$$\begin{aligned}
\frac{dn_A}{dt} &= F_{A0} - F_G y_A - kC_A C_B V_L - F_L x_A \\
\frac{dn_B^L}{dt} &= F_{B0} - kC_A C_B V_L - F_L x_B \\
\frac{dn_C}{dt} &= kC_A C_B V_L - F_L x_C - F_G y_C \qquad (6.68) \\
(\dot{x}_A^\star - \dot{x}_A) &= 0 \\
(\dot{x}_C^\star - \dot{x}_C) &= 0.
\end{aligned}$$

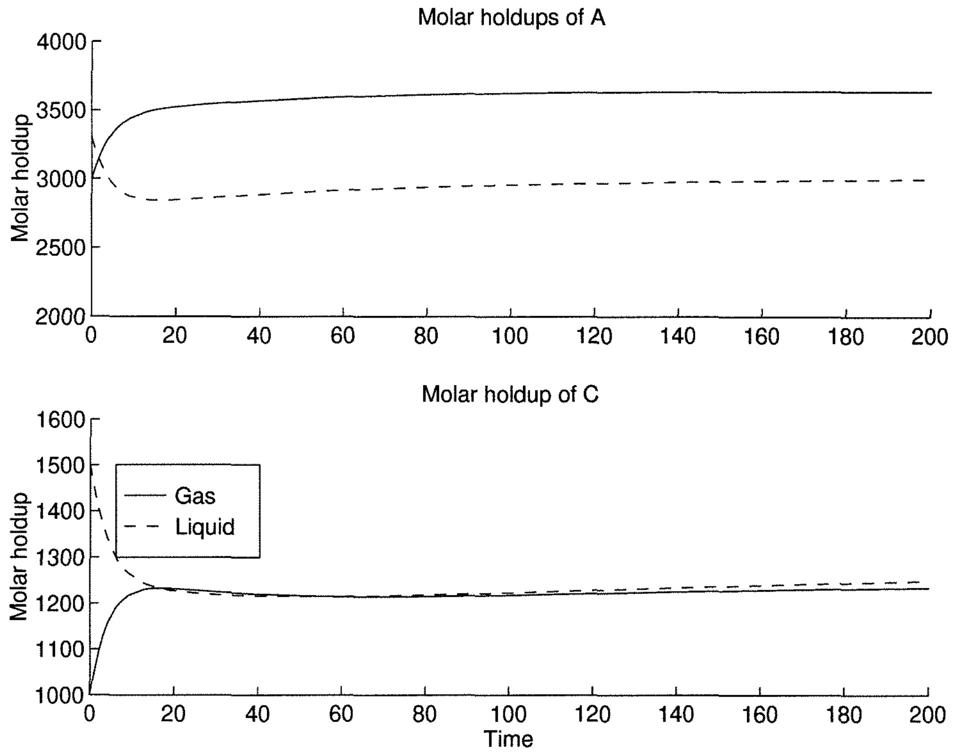


Figure 6.11: Simulation of the full set of equations for the CSTR for the first 200 seconds.

It is possible to convert back to the original set of coordinates by utilizing the algebraic constraints for calculation.

Figures 6.12 and 6.13 give the results of simulation based on System (6.68). The initial conditions for this simulation are such that $[n_A, n_B^L, n_B]$ match the initial conditions of the full system. The concentration differences $n_A^G - n_A^L$ and $n_C^G - n_C^L$ are then adjusted until $(\dot{x}_A^\star - \dot{x}_A) = 0$ and $(\dot{x}_C^\star - \dot{x}_C) = 0$. This is the projection of the dynamics onto the slow manifold that results from the physically motivated model reduction scheme.

In Figure 6.13 the fast dynamics do not appear. The reason is that the mass transfer terms N_A and N_C do not appear in the differential equations. Notice that the long term dynamics are reproduced quite well using this approximation. However, the simulated solution of this set of equations requires a combined differential algebraic equation solver. In addition, this model can only be used with control design schemes which are specifically developed for differential algebraic systems.

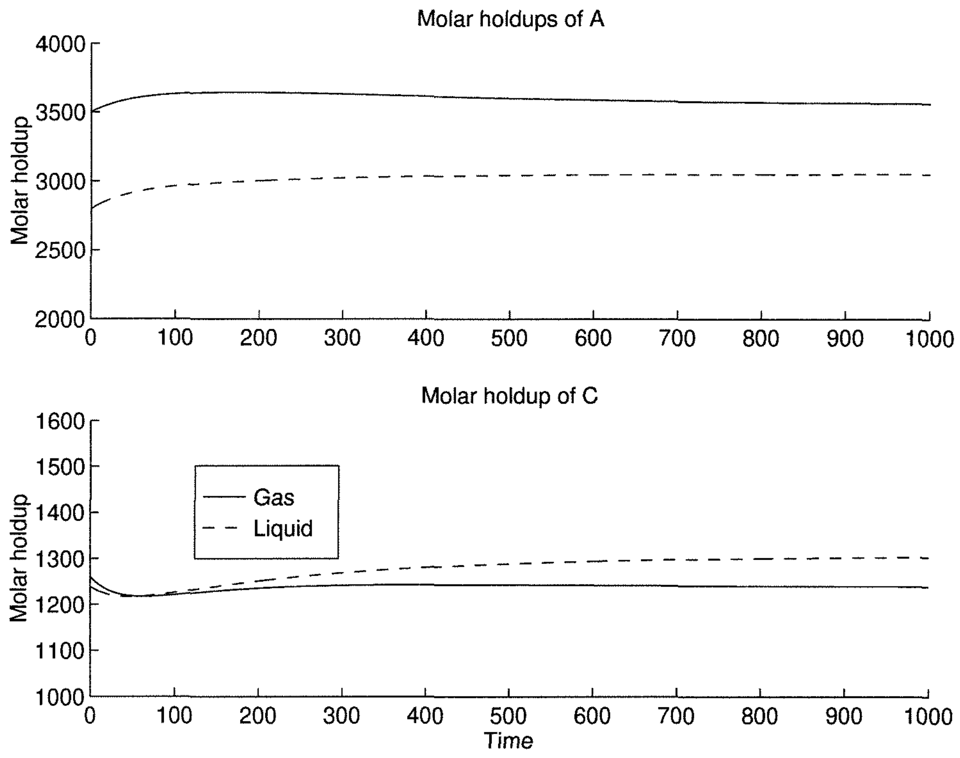


Figure 6.12: Simulation results for the physically motivated model reduction.

The computational algorithm of Maas and Pope was then used to identify the 3-dimensional manifold of slow dynamics. The variables $[n_C^G, n_A^L]$ are solved for as a function of the remaining variables $[n_A^G, n_B^L, n_C^L]$ over an equally spaced grid of 8,000 points for molar holdups less than 300 *mol* from the steady-state using the Maas and Pope algorithm. Computationally this takes approximately 15 minutes on a Sun SPARCstation 20. After this is completed, the variables $[n_C^G, n_A^L]$ are estimated as a function of the other variables using a quadratic polynomial. The fitted functions are

$$\begin{aligned} n_C^G &= H_1(n_A^G, n_B^L, n_C^L) \\ n_A^L &= H_2(n_A^G, n_B^L, n_C^L). \end{aligned} \tag{6.69}$$

Substituting these fitted functions into the original equations, reduced equations are as follows:

$$\frac{dn_A^G}{dt} = F_{A0} - N_A - F_G y_A$$

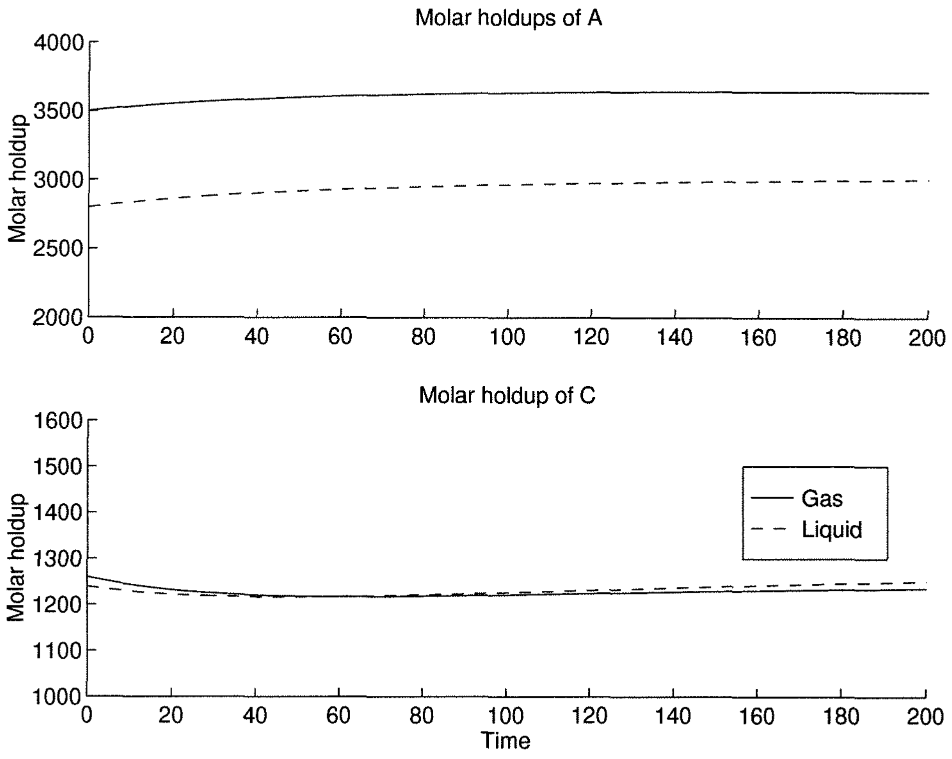


Figure 6.13: Simulation results for the physically motivated model reduction in the first 200 seconds.

$$
\begin{aligned}
\frac{dn_B^L}{dt} &= F_{B0} - kC_AC_BV_L - F_Lx_B \\
\frac{dn_C^L}{dt} &= N_C + kC_AC_BV_L - F_Lx_C \\
n_C^G &= H_1(n_A^G, n_B^L, n_C^L) \\
n_A^L &= H_2(n_A^G, n_B^L, n_C^L).
\end{aligned} \qquad (6.70)
$$

Note that for this set of equations, the reduced form is much more natural. By substituting the expressions $H_1(n_A^G, n_B^L, n_C^L)$ and $H_2(n_A^G, n_B^L, n_C^L)$ for n_C^G and n_A^L, the system can easily be simulated as a set of only 3 differential equations. This is much different than the system which results from the physically motivated model reduction where a differential algebraic solver is needed for simulation.

The results of the simulation of System (6.70) are given in Figures 6.14 and 6.15. In this simulations, the initial values of $[n_A^G, n_B^L, n_C^L]$ are specified to be identical to those values used in the simulation of the full system. The initial values of $[n_C^G, n_A^L]$

are determined from the constraints $H_1(n_A^G, n_B^L, n_C^L)$ and $H_2(n_A^G, n_B^L, n_C^L)$. Note that this choice of initial conditions for the reduced system is not necessarily the optimal choice. It is simply an arbitrary projection of the initial conditions onto the slow manifold.

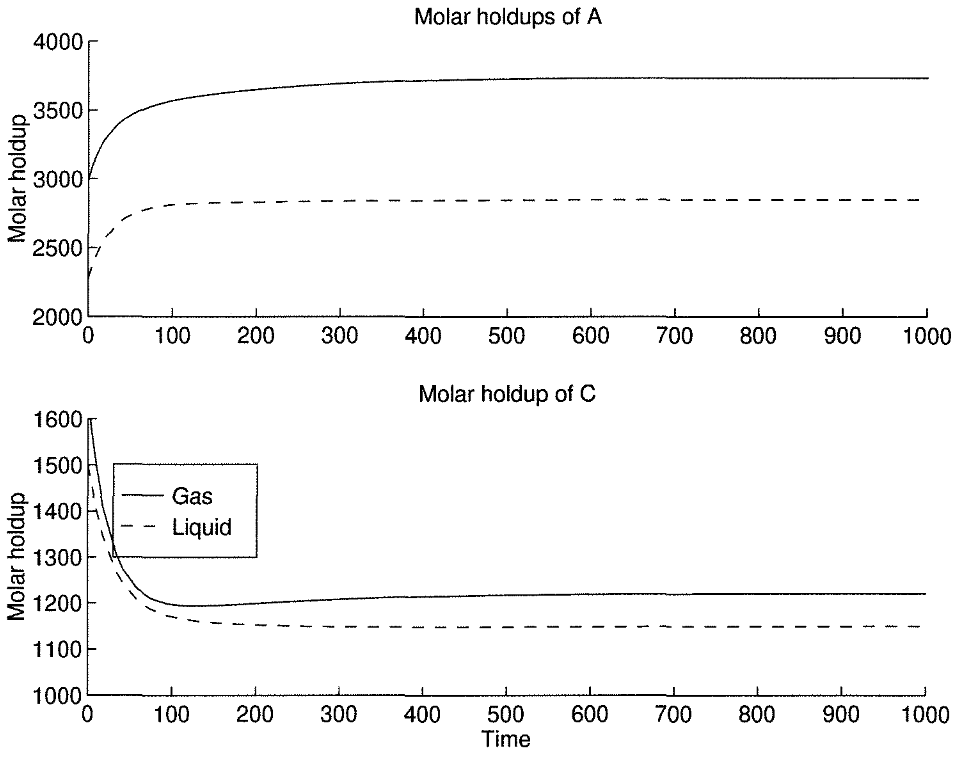


Figure 6.14: Simulation results for the reduced model generated by black-box fitting of the results of the Maas and Pope algorithm

The simulations using this model no longer exhibit the fast dynamics of the full set of equation. In addition, it appears that the computed initial conditions for $[n_C^G, n_A^L]$ are similar to the full simulation result after the mass transfer reaches steady-state (around 20 seconds). Dynamically, the results appear quite similar to the full simulation results if the initial transient is ignored. However, since the fit to the slow manifold is only approximate there is some error associated with the final steady-state of the reduced model. The error has a major impact on the steady-state behavior since the term N_A is not exactly cancelled out by the function H_1 and H_2. However, the results of this model reduction are a set of 3 differential equations which can be utilized easily by a nonlinear control scheme. Finally, Figure 6.16 compares the

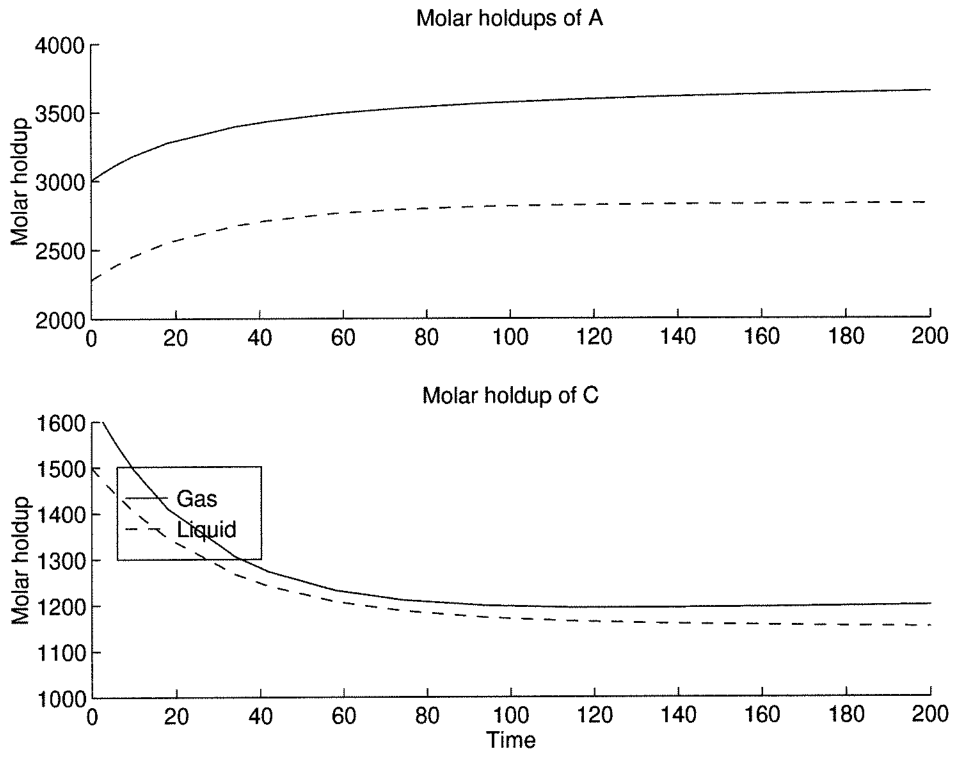


Figure 6.15: Simulation results for the reduced model generated by black-box fitting of the results of the Maas and Pope algorithm for the first 200 seconds.

results of the three different models for the molar holdup of component A.

This example was selected because the results of the physically motivated based model reduction work quite well. The physically reduced model gives a benchmark that the results of the Maas and Pope based model reduction scheme can be tested against. However, for many systems which exhibit time-scale separations these types of accurate assumptions will be difficult or impossible to identify. The computational algorithm provides a way to accurately identify the manifold of slow dynamics in a systematic way.

6.7 Conclusions

The algorithm of Maas and Pope (1992) has been examined as a tool for nonlinear model reduction of systems which exhibit behavior with time-scale separations. First, it was shown that the algorithm identified the proper reduced manifold of slow dy-

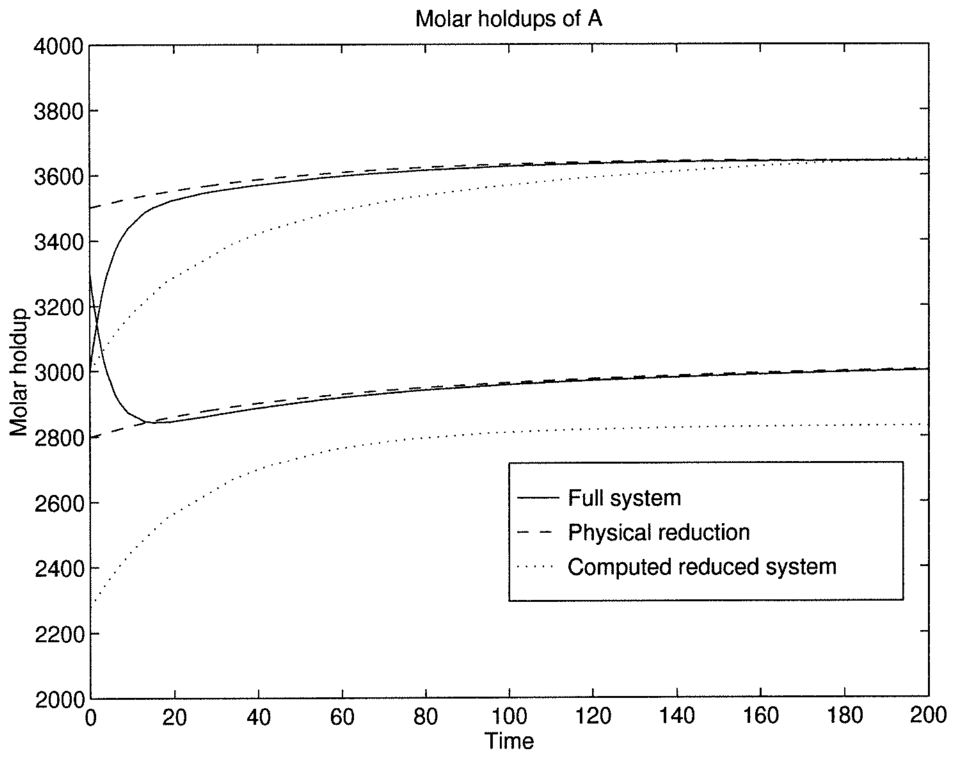


Figure 6.16: Comparison of the three different models for the first 200 seconds.

namics for systems with infinite and finite time-scale separations. The slow manifold of a simulated binary distillation column was found using the algorithm and the results were compared with previously utilized methods of linear model reduction and physically based nonlinear model reduction. The Maas and Pope algorithm appears to do the best job representing the slow manifold of the system.

Some thoughts on how the results of the algorithm could be used for model reduction were then presented and this method of model reduction was used on a model of a two-phase chemical reactor. The results were compared with the results of a standard physically based model reduction and the results of the full simulations. In this example, the slow dynamics of the system were reproduced quite well by the Maas and Pope algorithm and black-box modeling. However, the mismatch between the final steady-state of the reduced and full model illustrates the need for more work in this area. Possible subjects include determining the optimal projection for initial conditions in the state-space onto the slow manifold and determining more computationally efficient ways of developing reduced models.

Part IV

Data-based control methods

Chapter 7 An introduction to data-based control methods

7.1 Motivation

Historically, new developments in control applications and theory have been closely tied to advances in computational processing power. Before the development of processors of reasonable size and price, the state of the art in process control involved proportional, integral, and derivative control of single-input/single-output loops. Pneumatics were used for the implementation of these control laws. In order to determine appropriate tuning parameters for these controllers, online "trial and error" tuning methods were used extensively. The Ziegler-Nichols tuning method (Ziegler and Nichols, 1942) provides a starting point for this type of controller tuning. In order to use the Ziegler-Nichols method, a few simple closed-loop experiments are performed to determine the steady-state gain and critical frequency of the system to be controlled. From these two quantities, suggested values of the proportional, integral, and derivative gains are given by simple formulas. More complicated pencil and paper methods were also available for determining controllers for processes which could be represented by simple descriptions. However, a simple model of the process first had to be developed and few rigorous methods were available to ensure robustness of the controller.

In the late 60's and early 70's, the first small and relatively affordable processors became available. Computational power suddenly became available for solving reasonably sized numerical problems offline. Along with this advance in computational power came major advances in the control of state-space systems. Since controller design involving these models require fairly complicated matrix computations, advances in offline computational power made these methods feasible. The state-space methodology also made it possible to deal with multiple-input/multiple-output systems in a systematic manner. Simple online processors were available for the actual controller implementation and theory of the control of sampled-data systems was developed in order to account for problems which may arise due to digital implementation of the controller. Soon after, the theory of optimal control and robust control advanced significantly as a result of the increasing amounts of offline computational power available to control engineers.

In the 80's, processors had became so fast and inexpensive that fairly powerful personal computers became widely available to the general public. With the speed of these new processors, suddenly it became possible to perform significant computations online while the process was running. For the first time, control engineers could take advantage of a control method which could harness this processing power for computing the control law online. Before this time, controllers relied on fixed mathematical formulas for determining the controlled input. By performing an optimization problem at each sampling time of the system rather than using a fixed control law, it was hoped that better performance could be achieved. One specific implementation of this methodology is currently used widely throughout industry and is known as Dynamic Matrix Control (DMC) (Cutler and Ramaker, 1980). Academic research in this area has focused on the problem of Model Predictive Control (MPC) (Morari and Lee, 1997). Because of the feasibility of online computations, the theory of adaptive control also experienced interest. A class of these adaptive controllers perform online identification and adapt their control law to changing system conditions.

There have also been significant advances in the storage of computer data. Now it is relatively affordable to store massive amounts of data in an easily accessible format. For this reason, it is possible to collect massive amounts of data for relatively little cost. In order to get a competitive edge, companies have started to gather and store many different types of information. However, once the data are collected the companies have one major question. "How can the data be used?". In the area of marketing, companies are using new techniques known as "data mining" to search large data sets and identify trends among customers, products, and markets

(Strassel, 1997). With these techniques, useful bits of information that could easily be overlooked are identified from large amounts of data. In fact, it is even reported that data mining tools helped to make key adjustments in Orlando's game plan in the NBA playoffs after losing two straight games to the Miami Heat (Koprowski, 1997). Orlando went on to win the series.

In the field of process engineering and control, industry has developed large, archival data bases from operating processes. However, there is little understanding of how this data can be used for modeling, control, and process monitoring. At a number of recent conferences industry has called on academia to develop tools which would allow them to effectively utilize this data in process control and engineering (Trainham, 1996; Harg, 1997). It appears that gathering and collecting large amounts of data are no longer problematic, but gaining useful information from this data remains elusive. However, data-based control methods would allow these massive amounts of data to be used directly in control.

7.2 Local modeling

As discussed in Section 2.1, in order to form a black box models of process dynamics proper regression vector and functional relationship between regression vectors and future outputs needs to be determined. For a set of observed regressors $\psi(t)$ and observed outputs $y(t)$, a function G is to be found

$$y(t) = G[\psi(t)] \tag{7.1}$$

such that the choice of G minimizes the error associated with the approximation. While there are many methods available for forming the function G (examples are mentioned in Section 2.1), the one method which is most successful for time-series prediction of chaotic systems is "local modeling" (Abarbanel, 1996).

The idea of local modeling in the context of chaotic prediction is first discussed by Farmer and Sidorowich (1987). Instead of determining a single global function G

relating regressors to future outputs (Figure 7.1), it is suggested that a number of *local predictions* can be made using only regressors which are nearby when forming the prediction model (Figure 7.2). This way, a number of functions which approximate

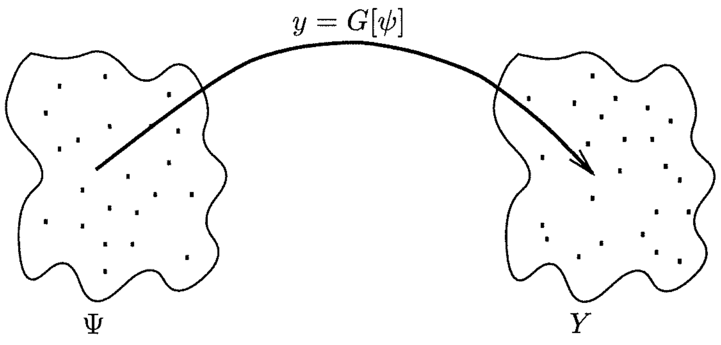


Figure 7.1: Standard global models approximate a single function which is valid for the entire region

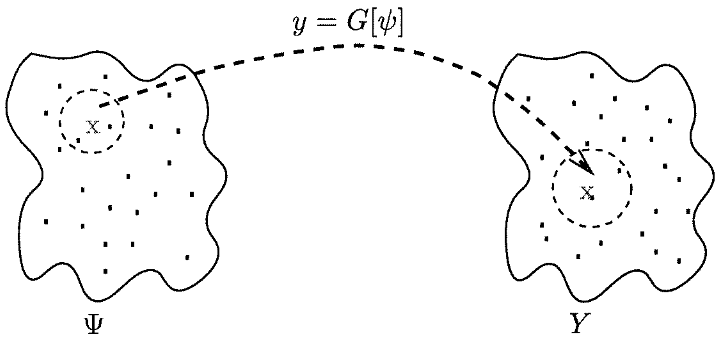


Figure 7.2: Local modeling relies on a number of models which are only valid locally

G in a small region can be utilized. To predict $y(t)$, a collection of regressor vectors $\psi(t_i)$ is found such that $||\psi(t) - \psi(t_i)||$ is small in some sense. Two methods are commonly used in order to determine whether regressors $\psi(t_i)$ are *local.* Either a specific number of the closest vectors or all vectors which are within a certain small distance of $\psi(t)$ are kept.

The simplest approach to local modeling is known as the zeroth-order approximation. In this method, the single regressor $\psi(t^\star)$ which is closest to the regressor of interest is found. The predicted future output for $\psi(t)$ is the future output associated with the nearest neighbor $y(t^\star)$. This method will be highly sensitive to noise, so often a more sophisticated approach known as a first-order approximation is used. In this approach, many local regressors are found and a linear least-squares problem is solved to determine the predicted output. For nearest neighbors $\psi(t_1), \ldots, \psi(t_n)$, θ is found by solving the following least-squares problem

$$\begin{bmatrix} y_d(t_1) \\ y_d(t_2) \\ \vdots \\ y_d(t_n) \end{bmatrix} = \begin{bmatrix} \psi_d(t_1) \\ \psi_d(t_2) \\ \vdots \\ \psi_d(t_n) \end{bmatrix} \theta \tag{7.2}$$

where ψ_d and y_d refer to deviation variables about the mean of the regressors and outputs in the local neighborhood. The predicted output associated with $\psi(t)$ for the first-order approximation is given by

$$\hat{y}_d = \psi_d(t)\theta. \tag{7.3}$$

It is also possible to use higher order polynomials (Casdagli, 1989) and weighted regressions (Sauer, 1993) with local modeling schemes.

The process of finding neighbors for a regressor in a data base of N points will require N computational steps if the data are searched in the most straight-forward manner. However, if the data are presorted into a decision tree only $\log N$ steps are needed to find the neighbors of the single regressor (Friedman, Bentley and Finkel, 1977). Since the neighbors will have to be found every time a prediction is made, a fair amount of computational power is needed for making predictions. This is different from the majority of modeling schemes where significant computational power is needed in order to build the model, but making predictions based on the identified model only involves evaluating a known function.

Some results have been given for bounding the error involved with predictions from local modeling schemes. It has been suggested by Farmer and Sidorowich (1987) that the error associated with prediction using this method is limited only by noise when the typical spacing between data points is $\approx N^{-1/D}$, where N is the number of data points and D is the regressor dimension. When there are less data available, the error for local linear predictors is of $\mathcal{O}(N^{-2/D})$ (Casdagli, 1989). In the prediction of chaotic time-series with large amounts of data, it was observed in a study by Casdagli (1989) that local predictors do a better job than a number of other standard modeling tools (radial basis functions, neural networks, and global polynomials).

A major drawback to local modeling is that in order to build a model, neighboring data points are needed. If data are too sparse, it is impossible to use local modeling schemes. Since local models rely on the training data to build a model, large amounts of data need to be stored and retained in order to make predictions. This is unlike traditional modeling schemes where the training data can be discarded after the model is built.

If a significant amount of noise contaminates the training data, local modeling may be inaccurate. Since local modeling only uses a small number of neighboring regressors in order to build a model, the effect of noise can be more significant than for global modeling schemes where large amounts of data tend to be "averaged" when building the model. For performing local modeling on training sets with significant noise corruption, the data may need to be pretreated by some noise reduction scheme. Grassberger, Hegger, Kantz, Schaffrath and Schreiber (1993) give a comparison of a number of methods of noise reduction for nonlinear systems.

There is one other possible drawback to using local modeling. When local modeling schemes are used for prediction, little to no physical insight can be gained about the system dynamics. However, this is also a problem for many other traditional nonlinear modeling schemes. It is not easy to gain physical insight into the dynamics of a process when modeling with other popular methods including neural networks and radial basis functions.

7.3 Extending previous control ideas

The idea of using numerous models for control of nonlinear system is not new. The idea of using a "stored response modeling" technique was suggested for controlling nonlinear systems by Eichler and Mansour (1979). In this method, a locally valid linear model is developed at a number of discrete grid points in the state-space. These linear models are developed offline using methods similar to the first-order local modeling scheme detailed above. Once the linear models are developed for all grid points, a locally valid solution of the Riccati equation can be found which is the optimal solution for a quadratic performance measure locally. These solutions can be stored for all the grid points and recalled for control of the system. The grid of control solutions in the state-space is called the "Riccati feedback table". When this controller is implemented on the system, the appropriate control move can be found by interpolating the feedbacks given in the Riccati feedback table for the appropriate location in the state-space. The results of this method for a few small nonlinear examples are suboptimal, but comparable to those of a globally optimal nonlinear controller.

Another approach to solving the problem is to approximate the nonlinear system as a piecewise linear system. By using piecewise linear systems to approximate nonlinear systems, it is hoped that some of the tools from linear systems can be modified and utilized for control. An introduction to how a piecewise linear approach might be applied to control of nonlinear systems is given by (Sontag, 1981). Another way to apply methods of linear control to a nonlinear system is to use a "gain scheduled" controller (Shamma and Athans, 1990).

The data-based control ideas which are suggested in the following sections will require both the ability perform online computations and the ability to quickly access large amounts of presorted data. The computational power should not be a problem since Dynamic Matrix Controllers which perform online optimizations for computing future control moves at each sampling time are used widely throughout the chemical processing and petroleum industries (Morari and Lee, 1997). For systems with large sampling times, performing the relatively simple online computations which would be associated with data-based control seem achievable.

The ideas in the following sections are only the first steps in examining how data-based control could be utilized. Obviously, there is a significant amount of work which would need to be completed before this methodology could be successfully applied to industrial problems. Some thoughts on how this can be accomplished and work for the future is detailed in Chapter 10. However, as will be shown in Chapter 9, preliminary results are quite promising.

In the next chapter, a method of computing controllable sets from a data-based methodology is given. A time n controllable set of a given point is defined as the region which can driven to the desired point in n steps by the appropriate control actions. The method used to compute these controllable sets does not require a mathematical description of the system dynamics. In the method presented here the controllable sets are determined directly from identification data of the system.

In Chapter 9, a method of computing an input trajectory that tracks specified reference output trajectory is presented. Once again the trajectory is computed directly from identification data from the system. In the examples presented in this section, the data-based trajectory planning method is implemented in an offline manner and applied to a pair of computational problems and an experimental nonlinear circuit.

A method of control similar to data-based trajectory planning was developed by Schaal and Atkeson (1994). In this article, a "memory-based learning" technique is used to teach a robot how to juggle a devil stick. The controller for the robot "learns" how to perform this repeatitive task by using local modeling techniques. Data are collected from the experiment online and these data are then used by the control algorithm online. It appears that the "memory-based learning" technique can be applied fairly succesfully to this problem in an online fashion.

Chapter 8 Computing controllable sets

Abstract

An algorithm is presented which determines the controllable sets of a system directly from time series data. By determining these controllable sets, it is shown that a minimum time trajectory steering control problem is solved. A global model of the dynamics is not needed for computation since the algorithm exploits local properties of the data. Two algorithms are presented. The more simplistic algorithm relies on a sorting algorithm, while the second algorithm uses local linear models and a continuation scheme to solve problems associated with noise corruption and continuity. Theoretical properties of controllable sets are also analyzed, and a simple example is presented.

8.1 Introduction

In chemical process control, many systems to be controlled are highly nonlinear (due to thermodynamics, reaction kinetics, heat transfer, or other reasons). When designing controllers for nonlinear systems, two methods are commonly used. The first method consists of designing a controller based on a first-principles state-space model. One disadvantage of this method is that determining the parameters of the state-space model can be difficult. In addition, when there is plant-model mismatch it is difficult to determine whether the mismatch is a result of improper parameters or unmodeled dynamics.

The second method of control design commonly used consists of two stages. An input/output model of the dynamics is developed first, and then a controller is designed for that model. A major problem with this method is that no universal nonlinear model structure has been discovered that works well for the identification of all nonlinear systems. Certain identification methods (neural networks, radial basis functions, MARS), which tend to be more accurate for identification purposes, result in models which can be difficult to utilize for analysis and control purposes. Other methods (such as polynomial functions), which give results in a form that can be easily utilized by existing control design methods, tend not to represent the dynamics of nonlinear systems accurately.

A method which could perform control analysis and design directly on the data would allow the control engineer to completely bypass the difficult modeling and identification stage of nonlinear control. In addition, by performing analysis directly on the data rather than the identified model (which is only an approximation of the identification data) more accurate results might be found. What is outlined here is a method to determine the set of points in the time delay coordinates which can be controlled to a reference point in a given number of sampling times directly from time-series data. The exact control moves needed to drive the system to the reference point is also found as a byproduct of this analysis.

One drawback of any data-based analysis scheme is that large amounts of data are needed to arrive at accurate results. As a result, massive amounts of computer storage and high computation speeds might be needed. However with lowering prices of computer storage and recent advances in computational speeds, data-based methods of control could become quite attractive in the near future for analysis and control of nonlinear systems.

8.2 Time delay coordinates

A geometric theorem of Takens (Takens, 1981) states that nonlinear autonomous state-space systems of the form

$$\dot{\mathbf{x}}(t) = f[\mathbf{x}(t)] \qquad \mathbf{x} \in \Re^n \tag{8.1}$$

$$y(t) = h[\mathbf{x}(t)] \qquad y \in \Re \tag{8.2}$$

can be equivalently represented by utilizing what are known as "time-delay coordinates". For the system defined by Equations 8.1 and 8.2, the time-delay description is

$$y(t) = G[y(t-\tau), \ldots, y(t-l\tau)] \tag{8.3}$$

where τ is some constant sampling time and G is some nonlinear relation which cannot be directly determined from f and h in general. However, there is a relation between the dimension of the state-space and the number of terms needed on the right hand side of Equation 8.3 to recreate the state-space dynamics. According to Takens, $l > 2n$ is a sufficient condition for the two dynamical descriptions to be equivalent. This relation is examined and the results are extended for autonomous systems in (Sauer et al., 1991).

Soon after the paper of Sauer *et al.*, it was hypothesized that a similar relation could be found for single-input single-output systems of the form

$$\begin{aligned} \dot{\mathbf{x}}(t) &= f[\mathbf{x}(t), u(t)] && (8.4) \\ y(t) &= h[\mathbf{x}(t)] && (8.5) \end{aligned}$$

where the input u changes only at sampling times of the system (Casdagli, 1992). The description in time-delay coordinates should contain delayed versions of the input as well as the output.

$$\begin{aligned} y(t) = \; & G[y(t-\tau), \ldots, y(t-l\tau), && (8.6) \\ & u(t-\tau), \ldots, u(t-m\tau)]. \end{aligned}$$

This result was formalized recently for systems with a discrete time state-space description (Poncet et al., 1995).

The existence of this relationship alone is not useful for analysis or identification of state-space dynamics from time-series data where the dimension of the state vector is unknown. In this case, the number of time-delay coordinates needed to represent the dynamics of the system is unknown *a priori*. For this reason, algorithms have been developed to determine the minimum number of time-delay coordinates needed to recreate the dynamic behavior using input/output time-series data (Poncet and Moschytz, 1994; Rhodes and Morari, 1995). These algorithms do not build global models to determine the underlying "dimension" of the data. Instead, local properties of the data are exploited to determine the minimum number of delays needed to reproduce the dynamics. Once the number of delayed terms (l and m) needed to represent the dynamics has been determined, identification and analysis can take place in the time-delay coordinates rather than the original state-space coordinates since the two spaces are diffeomorphic.

8.3 Controllable sets

Slight modifications to Equation 8.6 can be made to convert the time-delay description into one which resembles a state-space description where the idea of controllable sets is more intuitive. Let

$$\begin{aligned}\mathbf{z}(t) &= [y(t), \ldots, y(t-(l-1)\tau), \\ &\quad u(t-\tau), \ldots, u(t-(m-1)\tau)]^T.\end{aligned} \tag{8.7}$$

A function F can then be formed, using function G from Equation 8.6

$$\begin{aligned}F[\mathbf{z}(t-\tau), u(t-\tau)] = [G[&\mathbf{z}(t-\tau), u(t-\tau)], \\ &y(t-\tau), \ldots, y(t-(l-1)\tau), \\ &u(t-\tau), \ldots, u(t-(m-1)\tau)]^T\end{aligned} \tag{8.8}$$

such that

$$\mathbf{z}(t) = F[\mathbf{z}(t-\tau), u(t-\tau)] \tag{8.9}$$

$$y(t) = [1, 0, \ldots, 0]\mathbf{z}(t). \tag{8.10}$$

Using this new state-space description of the dynamics, the following definition can be made.

Definition 8.3.1 *The time n* **controllable set** *of a point $\mathbf{z}$, $\mathcal{C}_n(\mathbf{z}) \in \Re^{l+m-1}$, is the set of points for which there exists an input sequence that drives the system to $\mathbf{z}$ in n sampling times.*

What can be said about these controllable sets? First, some characterization can be made concerning the geometry of a controllable set.

Theorem 8.3.1 *$\mathcal{C}_i(\mathbf{z}_{\text{ref}})$ is the projection of an i-dimensional manifold onto the time-delay state-space.*

Proof: If $\mathbf{z} \in \mathcal{C}_i(\mathbf{z}_{\text{ref}})$, $\exists$ an input sequence $\{u_i, \ldots, u_1\}$ such that $F[F[\ldots F[\mathbf{z}, u_1], \ldots], u_i] = \mathbf{z}_{\text{ref}}$ by definition. The left hand side of this equation can be represented as a single function of all its variables, or $H[\mathbf{z}, u_1, \ldots, u_i] = \mathbf{z}_{\text{ref}}$ where $H : \Re^{(l+m-1)+i} \rightarrow \Re^{(l+m-1)}$. By the preimage theorem (Guillemin and Pollack, 1974), if $\mathbf{z}_{\text{ref}}$ is a regular value of H (which is generically true) then the preimage $H^{-1}(\mathbf{z}_{\text{ref}})$ is a submanifold of $\Re^{(l+m-1)+i}$ with dimension i. However, the controllable set consists only of points in the state-space $\Re^{(l+m-1)}$. Therefore, the controllable set is the projection of the manifold $H^{-1}(\mathbf{z}_{\text{ref}})$ onto its first $l+m-1$ components.

Say the system is to be controlled to a steady state ($\mathbf{z}^{\text{ss}} = F[\mathbf{z}^{\text{ss}}, u^{\text{ss}}]$). The following statement can be made concerning the controllable sets of a steady state.

Theorem 8.3.2 *$\mathcal{C}_i(\mathbf{z}^{\text{ss}}) \subset \mathcal{C}_j(\mathbf{z}^{\text{ss}})$ for all $i \leq j$*

Proof: Let $\mathbf{z} \in \mathcal{C}_i$. $\mathbf{z}$ can be controlled to the steady state $\mathbf{z}^{\text{ss}}$ in i control moves by definition. After those i control moves, the controlled input can then be chosen to be u^{ss} for the last $j-i$ control moves. Since $\mathbf{z}^{\text{ss}} = F[\mathbf{z}^{\text{ss}}, u^{\text{ss}}]$, the system remains at state $\mathbf{z}^{\text{ss}}$ after a total of j control moves. Thus $\mathbf{z} \in \mathcal{C}_j$.

This result has implications for control purposes. A simple control algorithm to drive a system from any state $\mathbf{z}$ to a steady state $\mathbf{z}^{\text{ss}}$ can be designed based on an algorithm which determines controllable sets and the related inputs needed to drive the system to that steady state.

Algorithm 8.3.1 Simple control algorithm

```
1. i=1
2. Calculate $\mathcal{C}_i(\mathbf{z}^{\mathrm{ss}})$.
3. Is $\mathbf{z} \in \mathcal{C}_i(\mathbf{z}^{\mathrm{ss}})$? If not, i=i+1 and return to step 2. Else, go to step 4.
4. Apply the needed i control moves to drive the system from $\mathbf{z}$ to $\mathbf{z}^{\mathrm{ss}}$.
```

Thanks to Theorem 8.3.2, it can easily be shown that this control algorithm determines the control moves needed to drive any point $\mathbf{z}$ to a desired steady state in the least time (provided it is possible to drive $\mathbf{z}$ to the steady state).

Another simple result is helpful in computing controllable sets of time greater than 1.

Theorem 8.3.3 $\mathcal{C}_{i+1}[\mathbf{z}_{\mathrm{ref}}] = \mathcal{C}_1[\mathcal{C}_i(\mathbf{z}_{\mathrm{ref}})]$

Proof: If a point $\mathbf{z}$ is able to be driven to the desired reference point $\mathbf{z}_{\mathrm{ref}}$ in $i+1$ steps, then by definition it must be possible to drive $\mathbf{z}$ into $\mathcal{C}_i(\mathbf{z}_{\mathrm{ref}})$ (the time i controllable set) in a single step by definition.

As a result, it is possible to compute controllable sets of any given time recursively using the following algorithm.

Algorithm 8.3.2 Computing controllable sets of $\mathbf{z}_{\mathrm{ref}}$

```
1. $i = 0$. Let $\mathcal{C}_0 = \mathbf{z}_{\mathrm{ref}}$.
2. $\mathcal{C}_{i+1} = \mathcal{C}_1[\mathcal{C}_i]$.
3. i=i+1. Repeat step 2.
```

To compute controllable sets of any time, only a single algorithm to compute time 1 controllable sets must be developed. Two versions of this algorithm are presented in the following section.

8.4 Computing controllable sets

Now that some results concerning controllable sets have been presented, algorithms which compute controllable sets are developed. If the underlying function describing the dynamics is known, computation of the controllable sets is straightforward. Here it is assumed that the algorithm can only access time-series data. Two methods of computing controllable sets will be presented in this section. The first method is a simple box and sort algorithm, while the second method utilizes a local modeling scheme and the continuation program AUTO.

8.4.1 Box and search method

The idea behind the box and search method is to search the time-series data for states which have reached the desired steady state in a single time step previously. If the controllable sets of many different points are to be computed, the data can be presorted into small boxes using the scheme of Grassberger for computational efficiency (Grassberger, 1990). The result of the box and search method is a set of discrete points, each of which lies very close to the actual controllable set.

Algorithm 8.4.1 Box and search algorithm

1. *Presort time-series data in the state-space into ϵ-sized boxes to minimize the search time for points (Grassberger, 1990). ϵ should be small to improve accuracy, but the choice of ϵ will be dependent on the amount of data available.*

2. *Find the set of points $S_{\mathbf{z}}$ in the data set which reached $\mathbf{z}_{\text{ref}}$ (or reached a size ϵ box around $\mathbf{z}_{\text{ref}}$) in one time step previously by searching the data. These points form the set $C_1(\mathbf{z}_{\text{ref}})$. By storing the input needed to drive each $\mathbf{z} \in S_{\mathbf{z}}$ to $\mathbf{z}_{\text{ref}}$, a control algorithm for points in C_1 is defined.*

3. *For the set C_2, repeat the algorithm to find all points $\mathbf{z}$ that reached $\mathbf{z}_{\text{ref}}$ (or reached a size ϵ box around $\mathbf{z}_{\text{ref}}$) in two time steps. Continue to determine all desired control sets.*

It was mentioned previously that controllable sets of time greater than 1 could be computed recursively using an algorithm for time 1 controllable sets. However when using the box and search algorithm, it is much easier to compute $\mathcal{C}_i$ for $i > 1$ directly rather than recursively because of the presorting scheme. The method of direct computation also has less error, since the error involved in ϵ box sorting will propagate during recursive application of the $\mathcal{C}_1$ box and search algorithm.

The box and search algorithm to find $\mathcal{C}_i$ should give an idea of the basic structure of the controllable sets. However, since the time-series data available to the algorithm is of finite length, the computed sets $\mathcal{C}_i$ will consist of discrete points. It would be preferable to find a continuous set of points which describe $\mathcal{C}_1$.

8.4.2 Local modeling

The set of points $\mathbf{z} \in \mathcal{C}_1(\mathbf{z}_{\text{ref}})$ meet the following condition for some input u by definition.

$$F[\mathbf{z}, u] - \mathbf{z}_{\text{ref}} = 0 \qquad (8.11)$$

Notice that when calculating $\mathbf{z}(t+1) = F[\mathbf{z}(t), u(t)]$, all the components of $\mathbf{z}(t+1)$ excluding the first are taken directly from component of the vector $[\mathbf{z}(t), u(t)]$. The first term of $\mathbf{z}(t+1)$ is $y(t+1)$, which can be found using Equation 8.6. Therefore to estimate $F[\mathbf{z}(t), u(t)]$, only an approximation of G from Equation 8.6 is needed.

Calculating the set of points $\mathbf{z}$ which satisfy Equation 8.11 is relatively simple when F is known, however here it is assumed that only time-series data are available. To estimate the behavior of a function at a single point or in a small region around that point, local modeling can be used (Abarbanel et al., 1993). In local modeling, a model is built using only data which are "close" to the point where the dynamics are to be modeled. The implicit assumption of all local methods is that the dynamics are locally continuous.

The simplest local modeling scheme is known as the method of analogs (Lorenz, 1969). When using this method to estimate the function $F[\mathbf{z}, u]$, the data point in the identification data set $[\mathbf{z}(t_{\text{nn}}), u(t_{\text{nn}})]$ which minimizes $||[\mathbf{z}, u] - [\mathbf{z}(t_{\text{nn}}), u(t_{\text{nn}})]||$ is found. The method of analogs predicts the value of the first component of $F[\mathbf{z}, u]$ to be $\mathbf{y}(t_{\mathrm{nn}}+1) = [1, 0, \ldots, 0]F[\mathbf{z}(t_{\mathrm{nn}}), u(t_{\mathrm{nn}})]$, the "closest" example of the desired behavior. While this method does a good job of illustrating the idea of local modeling, it is highly sensitive to noise and may not be accurate when $||[\mathbf{z}, u] - [\mathbf{z}(t_{\mathrm{nn}}), u(t_{\mathrm{nn}})]||$ is relatively large.

Another method of local modeling involves building a linear map. Only data points which are "close" are considered when computing the parameters of the local linear model. Using this method, the parameters are found by solving the least-squares problem

$$\mathbf{Y}_{t+1} = [\mathbf{Z}_t \mathbf{U}_t]\theta \tag{8.12}$$

for θ, where $\mathbf{Z}_t$, $\mathbf{U}_t$, and $\mathbf{Y}_{t+1}$ are matrices consisting of rows $\mathbf{z}(t)$, $u(t)$, and $y(t+1)$ respectively for times where the data are close in the sense of $||[\mathbf{z}, u] - [\mathbf{z}(t), u(t)]||$. Two different methods are traditionally used to determine which data will form the matrices $\mathbf{Z}_t$, $\mathbf{U}_t$, and $\mathbf{Y}_{t+1}$. One method suggests that all points where $||[\mathbf{z}, u] - [\mathbf{z}(t_{\mathrm{nn}}), u(t_{\mathrm{nn}})]|| < \delta$ should be used, where δ is an adjustable parameter. The second method states that data in the identification set should be sorted by distance $||[\mathbf{z}, u] - [\mathbf{z}(t_{\mathrm{nn}}), u(t_{\mathrm{nn}})]||$ and a certain number of the nearest neighbors are used.

For reasons of continuity, neither of these methods are ideal for use in the algorithm about to be presented. Instead, a modified version of the first method is utilized. Instead of solving a simple least-squares problem to form the local model, a weighted least-squares problem is solved. The weighting used takes the form of a radial basis function $(d^2 + c^2)^{-b}$ where $d = ||[\mathbf{z}, u] - [\mathbf{z}(t_{\mathrm{nn}}), u(t_{\mathrm{nn}})]||$ and c and b are positive adjustable parameters. Using this weighting, points close to the reference point are weighted more in the computation of the parameters. The weight is calculated for each neighboring point, and a diagonal matrix of the weights $\mathbf{W}$ is formed. The parameters are then found by solving the following least-squares problem for θ.

$$\mathbf{W}\mathbf{Y}_{t+1} = \mathbf{W}[\mathbf{Z}_t \mathbf{U}_t]\theta \tag{8.13}$$

This process is repeated for each point $[\mathbf{z}, u]$ where a local description of the dynamics is desired.

8.4.3 Using AUTO to determine controllable sets

Now that the local modeling scheme has been detailed, a method for calculating the controllable set $C_1(\mathbf{z}_{\text{ref}})$ utilizing this local scheme is presented. The method should find all values of the vector $[\mathbf{z}(t), u(t)]$ in the range of the identification data such that $\mathbf{z}_{\text{ref}} = F[\mathbf{z}(t), u(t)]$. Because of the form of F, delayed outputs $y(t-\tau), \ldots, y(t-(l-1)\tau)$ and inputs $u(t-2\tau), \ldots, u(t-(m-1)\tau)$ of $\mathbf{z}(t)$ are fixed by the corresponding components of $\mathbf{z}_{\text{ref}}$.

Let the function H be defined as below.

$$H[\mathbf{z}_{\text{ref}}, \mathbf{z}(t), u(t)] \equiv F[\mathbf{z}(t), u(t)] - \mathbf{z}_{\text{ref}}. \tag{8.14}$$

When $H = 0$, the point $[\mathbf{z}(t), u(t)]$ is in the time 1 controllable set. The only free variables in Equation 8.14 are $y(t-(l-1)\tau)$ and $u(t-(m-1)\tau)$, with the rest of the vector components fixed by the structure of F. Thanks to this special structure, the problem of finding the controllable set becomes a search over two free parameters to find where the first component of H is zero. This is equivalent to solving,

$$\begin{aligned}[-1, 0, \ldots, 0]\mathbf{z}_{\text{ref}} + G[y(t), \ldots, y(t-(l-1)\tau) \\ u(t), \ldots, u(t-(m-1)\tau)] &= 0\end{aligned} \tag{8.15}$$

the first component of H found from Equation 8.6. Rearranging the variables of Equation 8.14, the problem to be solved is

$$[1, 0, \ldots, 0]H(\mathbf{v}_{\text{fixed}}, \mathbf{v}_{\text{free}}) \equiv H^1 = 0 \tag{8.16}$$

where H^1 is a nonlinear function, $\mathbf{v}_{\text{fixed}}$ consists of the fixed parameters and $\mathbf{v}_{\text{free}} \equiv [y(t-(l-1)\tau), u(t-(m-1)\tau)]$ contains free parameters. This is an algebraic continuation problem, a problem which AUTO (a standard program for analysis of nonlinear dynamical equations) solves (Doedel, 1986).

A companion program for AUTO was written to presort the data (for determining nearest neighbors) and build local models for the local evaluation of Equation 8.14. Continuation is used to determine the entire controllable set from an initial point contained in the set. The algorithm for determining controllable sets using AUTO takes the following form.

Algorithm 8.4.2 AUTO continuation based algorithm

1. *Find an initial point contained in the controllable set. This is best done using the previously defined box and search algorithm with very small ϵ. Since the initial point is in the controllable set, $H^1 = 0$ when evaluated at this point.*
2. *AUTO makes small steps in the free parameters such that $dH^1/dv_{\text{free}} = 0$. The function H^1 is evaluated using the weighted local modeling scheme outlined previously. AUTO checks to ensure that $H^1(\mathbf{v}_{\text{fixed}}, \mathbf{v}_{\text{free}}) = 0$ for points along the continuation. In this way, all points $\mathbf{z} \in C_1(\mathbf{z}_{\text{ref}})$ and related inputs u which drive the system to $\mathbf{z}_{\text{ref}}$ are determined.*
3. *To find C_2 (for example), the set of state which can be driven to a grid of distinct points in C_1 is computed using the same algorithm.*

8.5 Example

A simple example is presented to show the results of the two algorithms. A random white noise input was used to drive the following discrete time system, and the time-series data were stored.

$$\begin{aligned} 10y(t) &= y(t-1) + 4y(t-1)^2 \\ &\quad +3u(t-1) - u(t-2) \end{aligned} \tag{8.17}$$

The two algorithms were then applied to the 100,000 point data set to determine the time 1 controllable sets for the steady state point $\mathbf{z}^{\text{ss}} = [y(t), u(t-1)] = (0, 0)$.

Since the system is described in the discrete domain, $\mathcal{C}_1(\mathbf{z}^{ss})$ can be computed directly from Equation 8.17. The results of the box and search algorithm with $\epsilon = 0.01$ are presented in Figure 8.1. The theoretical time 1 controllable set is given by the solid line, with the individual points referring to the results of the box and search algorithm.

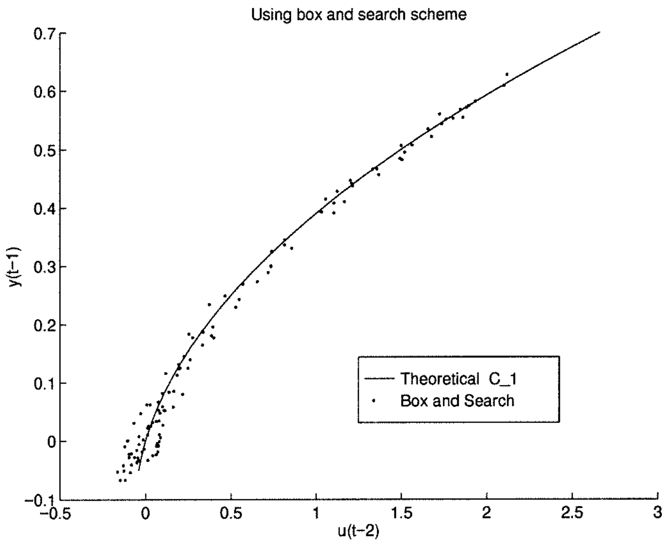


Figure 8.1: Results of box and search algorithm

The results of the AUTO based scheme for computing controllable sets are presented in Figure 8.2. The controllable set found by this algorithm is illustrated by the circles on the figure. Notice that the AUTO continuation scheme gives results which are indistinguishable from the theoretical result. However, the algorithm is not able to continue beyond the end point given since the data in that region becomes sparse and the algorithm has convergence problems. It also might be the case that the adjustable parameters of the local modeling scheme are not optimally selected for this system. Further experience with this algorithm is needed for determining the optimal parameters and the data requirements.

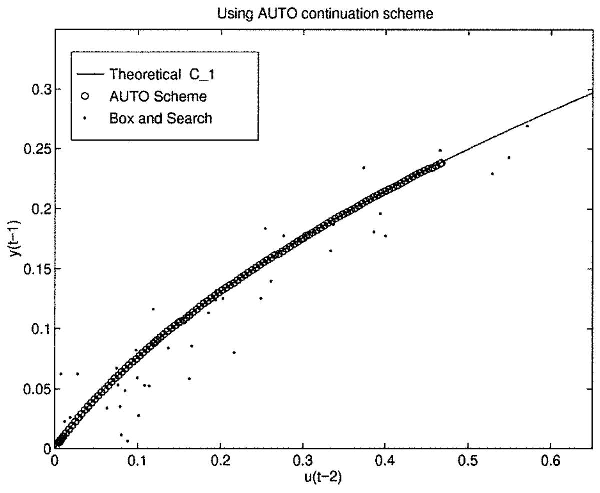


Figure 8.2: Results of AUTO continuation algorithm

8.6 Conclusions

A method to determine controllable sets has been illustrated. A possible advantage of this method is that local properties of data are used rather than a specific model structure. Since the algorithm can only work in the range of identification data, it is also well suited for estimation of controllable sets for systems with constrained inputs.

While the box and search algorithm is easy to implement for any system, the AUTO based continuation scheme has a number of tunable parameters which can be difficult to choose. AUTO normally evaluates nonlinear equations directly when carrying out analysis. For this reason, it can be very sensitive to small errors and tuning of the parameters in the local modeling scheme is very important for convergence. More experience with the algorithm is needed, and it may be necessary to formulate a new algorithm designed specifically for determining controllable sets using these same methods.

The control method outlined here is very simple, however it is only a first step towards control in what could be described as a data-based MPC scheme. In data-based MPC, building a model would not be necessary as the algorithm would have access to large amounts of data for making control decisions. While a data-based controller would possibly require massive amounts of storage, it might be the case that searching the data is more efficient computationally than carrying out a nonlinear optimization problem at each sampling time. While these thoughts are quite interesting, more research in this area needs to be completed before any substantial claims can be made.

Chapter 9 Data-based control trajectory planning for nonlinear systems

Abstract

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

9.1 Introduction

Numerous methods for the control of nonlinear systems have been developed recently. In the control community, some of the more popular methods include geometric control methods based on methods from differential geometry (see (Isidori, 1989) for an introduction), nonlinear model predictive control (Meadows and Rawlings, 1997), and control based on neural networks (Su and McAvoy, 1997). In order to use these methods for control it is necessary to have an accurate description of the system dynamics. This model can be the result of physical knowledge of the system dynamics or the result of system identification. While these methods are popular in the literature of the control community, different methods of control have been pursued for the control of chaotic systems.

Recently published methods of control for chaotic systems also build controllers based on knowledge of the system dynamics. However, most of these methods rely on knowledge of the underlying dynamics of the undriven, autonomous system (Ott, Grebogi and Yorke, 1990; Peng, Petrov and Showalter, 1992; Pyragas, 1992). The chaotic system is then stabilized around an unstable periodic orbit or fixed point using proportional linear feedback control. In the method of Ott, Grebogi, and Yorke (OGY) (Ott et al., 1990) and a number of later modifications, a scalar controlled input is changed at discrete times such that a periodic orbit or fixed point of the system becomes stable. The implementation of the OGY algorithm requires knowledge of the linearized dynamics of the periodic orbit to be stabilized (a fixed point on the Poincaré section) and the linearized dynamics which result from small perturbations to the controlled input about some nominal value. This information can be found by linearizing the uncontrolled system dynamics and making small perturbations in the controlled input. Since the goal trajectory in the state space coincides with an existing unstable trajectory of the uncontrolled system, stabilization is achieved by infinitesimal perturbations of the input.

Other feedback methods (occasional proportional feedback control(Peng et al., 1992), continuous control(Pyragas, 1992)) also stabilize existing trajectories of the unforced system by making small perturbations in the controlled input. While these methods (with finite driving forces) in principle can be used to drive a system towards an orbit which is not a solution of the unperturbed system, they do not provide an algorithm for finding a driving signal necessary for producing and stabilization of a pre-defined orbit.

Another class of chaotic control schemes attempts to drive a system such that an arbitrary goal trajectory is tracked. To this end, open-loop ("entrainment") control schemes have been suggested (Hübler, 1989; Jackson, 1990; Breeden, 1994; Mettin, Hubler, Scheeline and Lauterborn, 1995; Ho, Chern and Wang, 1994; Chen, 1996). Originally, entrainment control was utilized on known dynamical systems where the controlled inputs directly affect each state variable of the system(Hübler, 1989). Later this method was generalized for reconstructed dynamical systems(Breeden, 1994) and an arbitrary combination of inputs. However, once again it was assumed that the inputs are able to entirely specify the state of the dynamical system(Mettin et al., 1995). Clearly, the number of controlled inputs cannot be less than the state dimension of the underlying dynamical system when using this control scheme. Additionally, the stability of this open-loop control scheme cannot be guaranteed unless certain conditions are fulfilled. In particular, goal trajectories must be contained within "convergent" regions(Jackson, 1990; Mettin et al., 1995) of the state-space. A major problem with this method of control is that the problem definition is somewhat artificial. In most physical systems "full state control" (control which directly affects all the states) is not possible since some of the states of the system may not be directly affected by the input.

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

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

Since the trajectory to be tracked may be unstable, it is possible that the computed open-loop control trajectory may not stabilize the system. This is because it is difficult to exactly cancel the instability present in the trajectory of the open-loop system. In this case, additional closed-loop feedback control may be necessary to stabilize the system. It is also possible that the "inverse" mapping which produces the open-loop control law may be unstable. This would lead us to believe that the dynamic system contains non-minimum phase behavior, and the system may exhibit problems very similar to internal instability problems which can be found in linear systems when inverse controllers are used (Morari and Zafiriou, 1989). Both of these problems will be illustrated and examined in more detail in the examples.

9.2 Method

Consider the following nonlinear dynamical system

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, u), \\ y &= h(\mathbf{x}). \end{aligned} \tag{9.1}$$

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

In control theory, a system is called "output controllable" when it is possible to use a controlled input to produce any desired output (Balasubramanian, 1989). While theorems exist which allow us to determine the controllability of nonlinear systems

(Isidori, 1989), these theorems are dependent on a state-space model of the system. In situations where no state-space model (9.1) exists and only an input/output time-series is available, determining whether a system is globally controllable is a problem which has not been solved to the authors' knowledge. However, preliminary results in this area do exist. In the previous chapter, a computational algorithm for computing controllable sets directly from time-series data is outlined. In this work, it is simply assumed that the goal trajectories can be produced by an input trajectory in the examples.

In recent papers (Casdagli, 1992; Poncet et al., 1995), the Takens embedding theorem is extended to deal with input/output systems. Specifically, for system (9.1) future outputs can be generically represented as a function of time-delayed versions of the input and output (assuming the input remains constant between sampling times) as follows:

$$y(t) = P[y(t-T), y(t-2T), ..., y(t-lT), u(t-T), ..., u(t-mT)] \qquad (9.2)$$

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

While this model is guaranteed to exist, for most physical systems only identification is available and the exact form of the state-space dynamics (9.1) is unknown *a priori*. Additionally since $l, m \geq d+1$ is only a sufficient (and not a necessary) condition for a model of the form (9.2) to exist, there may be l and m smaller than $d+1$ such that (9.2) exists. For these reasons, a way of determining the smallest values of l and m from input/output time-series data has been developed using an extended version of the false nearest neighbors (FNN) algorithm (Rhodes and Morari, 1995). Once the proper number of embedded terms on the right hand side of (9.2) has been determined, the function P can be described locally for predictive purposes using nonlinear modeling techniques based on local polynomial predictors (Casdagli, 1992; Hunter, 1992; Abarbanel et al., 1993). Given a known input series, the system output could be predicted by repeated "one-step ahead" prediction.

For purposes of open loop trajectory planning, the input sequence $u(t)$ should be determined as a function of a desired output sequence $y(t)$. Using the Implicit Function Theorem, the equation (9.2) can be locally inverted as

$$u(t) = Q[y(t{+}T), y(t), y(t{-}T), y(t{-}2T), ..., y(t{-}(l{-}1)T), u(t{-}T), ..., u(t{-}(m{-}1)T)]. \tag{9.3}$$

Given the y terms (the values of the goal trajectory), this equation represents an m-dimensional non-autonomous mapping for the desired control u. Just as in case of modeling the output dynamics (9.2), this inverse map can be recovered from the data by using local polynomial models in the space of delayed versions of y and u. Once this map is determined locally, it can be used to calculate the "one-step ahead" control move $u(t)$ that will give the desired reference output $y(t+T)$.

Two problems may be encountered utilizing this local inversion process. First, it is possible that the control needed to produce the desired output is not contained in the data set. In this case, further identification with an input signal which has either a larger magnitude range or a wider frequency range may be needed. Second, the mapping Q from (9.3) is not guaranteed to be unique. For non-unique inverse mappings only the data corresponding to one branch of the inverse map can be used for inverse modeling purposes (see Figure 9.1). If the data from both branches are used for building a model, the model will "average" the data from the two branches and the resulting computed future control move u will lie somewhere between the two branches. In this case, the computed control move will not produce the desired action.

9.3 Computational algorithm

The goal of the computational algorithm presented here is to determine an input trajectory which will produce a desired output trajectory when applied to the system. It will be shown that the proper input trajectory can be computed directly in an open loop fashion from an input/output time-series of the system. Traditionally two

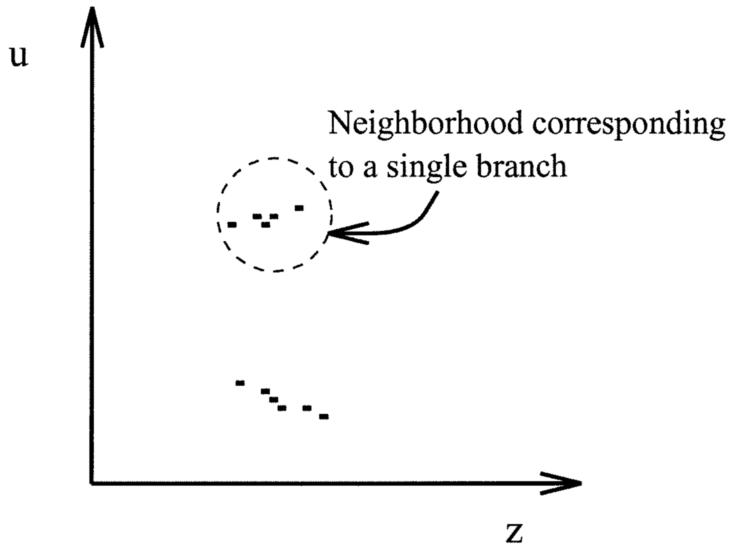


Figure 9.1: Example of a non-unique inverse mapping for the input u. Only a set of points which are close with regards to the input u can be used here for the local linear approximation.

distinct steps are completed for controller design. First, input/output identification data is analyzed and a model of the dynamics is formed. Then a controller is designed using the identified model and the controller is implemented on the actual system. Here, a controller will be designed which determines the input control trajectory *directly* from input/output time-series data of the system. This method of control is more computationally intensive than open loop control schemes which utilize fixed control laws, however it may give better results for systems with complicated dynamics where accurate identification is difficult.

While the computational algorithm does not need a global description of the dynamics, the number of delayed terms needed to recreate the dynamics (l and m) is needed. From this information, a local model of the dynamics in the neighborhood around the desired trajectory is built utilizing time-series data from the training set. By using data which are "near" to the desired dynamics (in the sense of distance in the regression space defined by the right hand side of (9.3)), a locally valid linear model of the dynamics is identified and the proper control move is computed. Once the proper control move is calculated, the process is repeated. This method is different from the OGY method (Ott et al., 1990) since a local linear model is formed for each sampling time of the system. In the OGY method, the control move is implemented periodically and the trajectories to be tracked must also be periodic. The method proposed here is not limited to tracking periodic signals.

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

1. The data set is presorted using the method of Grassberger (Grassberger, 1990) in order to reduce the search time needed by the algorithm. The training data are sorted into a two dimensional grid to save time when searching for neighbors within distance δ of a given point in the space $\Re^{l+m}$. The data are presorted into two dimensional bins in the regression space of the mapping in (9.3) ($[y(t+T), y(t), \ldots, y(t-(l-1)T), u(t-T), \ldots, u(t-(m-1)T)]$).

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

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

$$\mathbf{X} = \begin{bmatrix} \mathbf{z}(\mathrm{NN}_1) \\ \mathbf{z}(\mathrm{NN}_2) \\ \vdots \\ \mathbf{z}(\mathrm{NN}_p) \end{bmatrix} = \begin{bmatrix} y(\mathrm{NN}_1 + T) & y(\mathrm{NN}_1) & \cdots & u(\mathrm{NN}_1 - (m-1)T) \\ y(\mathrm{NN}_2 + T) & y(\mathrm{NN}_2) & \cdots & u(\mathrm{NN}_2 - (m-1)T) \\ \vdots & \vdots & \ddots & \vdots \\ y(\mathrm{NN}_p + T) & y(\mathrm{NN}_p) & \cdots & u(\mathrm{NN}_p - (m-1)T) \end{bmatrix} \tag{9.4}$$

$$\mathbf{u} = \begin{bmatrix} u(\mathrm{NN}_1) \\ u(\mathrm{NN}_2) \\ \vdots \\ u(\mathrm{NN}_n) \end{bmatrix} \tag{9.5}$$

where $\mathbf{z}$ and u are deviation variables about the point $\mathbf{z}_0$ which we are interested in.

To solve for the unknown parameters in the linear model, a weighted least-squares problem is solved. Specifically, the least-squares problem

$$\mathbf{WX}\theta = \mathbf{Wu} \tag{9.6}$$

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

$$u_0(t) = \mathbf{z}_0\theta. \tag{9.7}$$

Note that this is a local linear approximation to equation (9.3).

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

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

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

$$\mathbf{z}_0 = [y_0(t+T), y_M(t), \ldots, y_M(t-(l-1)T), u_0(t-T), \ldots, u_0(t-(m-1)T)] \quad (9.8)$$

where the y_M terms are the measured outputs from the system. With this change, the control law could be calculated online in a "one step ahead" manner. The only limitation to this closed-loop method is that the sampling time must be larger than the computation time needed to determine the next control move $u_0(t)$.

Currently, the algorithm does not account for the possibility that the inverse mapping (9.3) may not be unique. Another problem which could be encountered is an unstable inverse mapping (9.3). This instability may result in an input which becomes unbounded. If the dynamics of the system do not exactly cancel this input, the output behavior will not track the desired trajectory. However, as we will see in the next section, it is possible that inverse mappings which are unstable may lead to acceptable results for open-loop control purposes. In addition, the training set must cover the entire range of inputs needed for the proper control trajectory. There is currently no way to determine the proper range of inputs *a priori*. If the range of inputs is not large enough, there will be no "near neighbors" to the vector $\mathbf{z}_0$ and a local linear model cannot be built.

9.4 Applying the computational algorithm

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

9.4.1 Driven Lorenz model

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

$$\begin{aligned} \dot{x} &= \sigma(y-x), \\ \dot{y} &= -xz + rx - y + \epsilon(u(t) - y), \\ \dot{z} &= xy - bz. \end{aligned} \tag{9.9}$$

The parameter values $r = 45.62$, $b = 4$, $\sigma = 16.0$, which correspond to chaotic behavior of the undriven system, are used. System (9.9) has a controlled input $u(t)$ which appears only in the equation for $\dot{y}$. The output of the system is $x(t)$, and we would like to drive the system such that a desired periodic trajectory $x_0(t)$ is produced. For large values of ϵ, it is expected from studies on synchronization that open-loop implementation of the computed control trajectory will be stable (Kocarev and Parlitz, 1996).

A control signal which causes the output to track the desired trajectory will be constructed from identification data using the methods illustrated previously. First, the Lorenz system is subjected to driving by a random input signal $u_{\text{train}}(t)$ obtained by passing white noise through a low-pass filter (the cutoff frequency of the filter is taken to approximately correspond to the frequency range of intrinsic oscillations of $x(t)$). $u_{\text{train}}(t)$ and $x_{\text{train}}(t)$ are recorded using a sampling time of 0.02 and the time-series is of length 50,000. The time delay T is found using average mutual information analysis (see(Abarbanel et al., 1993)) of the input/output data, and the appropriate embedding dimensions $l = 2$, $m = 2$ are calculated by applying the input/output false nearest neighbors algorithm to the data. The identification data are used by the trajectory planning algorithm to form local inverse maps of form (9.3). A threshold distance of $\delta = 1.0$ is used to determine if points from the time-series are considered as neighbors for the local modeling.

The data-based entrainment algorithm is used to calculate the input for driving the system such that the output trajectory $x_0(t) = 20\sin(.523t) + 5$ is produced. The value of the parameter $\epsilon = 20.0$ (9.9) is used in this example, and the results are presented in Figure 9.2. In Figure 9.2a,b parts of training sets $u_{\text{train}}(t)$ and $x_{\text{train}}(t)$

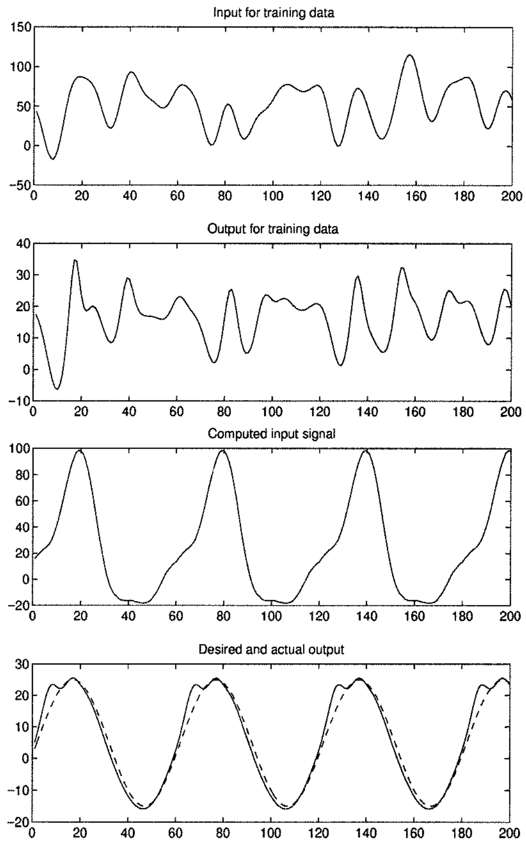


Figure 9.2: Data-based trajectory planning for Lorenz system (9.9) with $\epsilon = 20.0$. The goal trajectory of the output variable, $x_0(t) = 20\sin(.523t) + 5$. Inverted map was reconstructed from test driving the Lorenz system by randomized input. A time-series of length 50000 was used, and the parameters of the model were chosen: $l = 2, m = 2$. *a,b* - simultaneous time-series of the input $u(t)$ and output $x(t)$ from the training time-series; *c* - control sequence calculated using data-based trajectory planning; *d* - output signal resulting from the computed input signal.

are shown. Since the output trajectory to be tracked is periodic, the trajectory planning algorithm is run until the input signal converges to a periodic signal (the initial transient is discarded). Figure 9.2c shows the control signal $u_1(t)$ obtained after the output of the data based trajectory planning algorithm converges to a periodic signal. Figure 9.2d shows the desired output behavior $x_0(t)$ (dotted line) and the output $x_1(t)$ produced by the Lorenz system (9.9) when driven by the control signal

$u_1(t)$ (solid line).

A possible reason that tracking is poor near the "top" of the sine wave is that the local linear mapping for the input is unstable for portions of the trajectory. It is expected that the data based scheme will compute the exact inverse of the Lorenz system; however, if this inversion is not exact the system will not track the desired output exactly. A plot of the output and the location of the single pole of the local inverse mapping as a function of time are given in Figure 9.3. Any time the pole

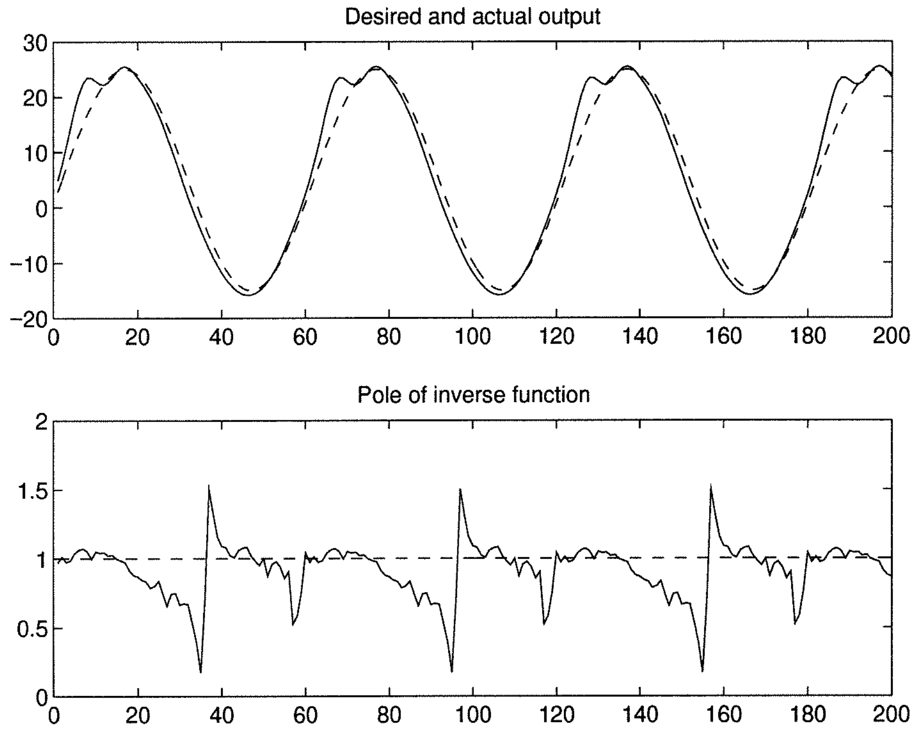


Figure 9.3: Illustration of instability of the inverse mapping. The pole of the local linear inverse is plotted as a function of time.

is greater than 1, the inverse mapping is unstable. Notice that fairly long term instability of the inverse mapping appears to correspond to poor tracking of the desired output (seen from time 60 to 70). However, the other region of instability where the unstable pole is quite large (near time 100) does not seem to affect tracking of the output possibly because the cancelation of the systems dynamics is nearly exact in this region of the dynamics.

Figure 9.4a,b show input and output time-series for tracking of a more complicated signal $x_2(t) = 10\sin(t)+10\cos(0.5t)+5$. Again, reasonably good reconstruction of the

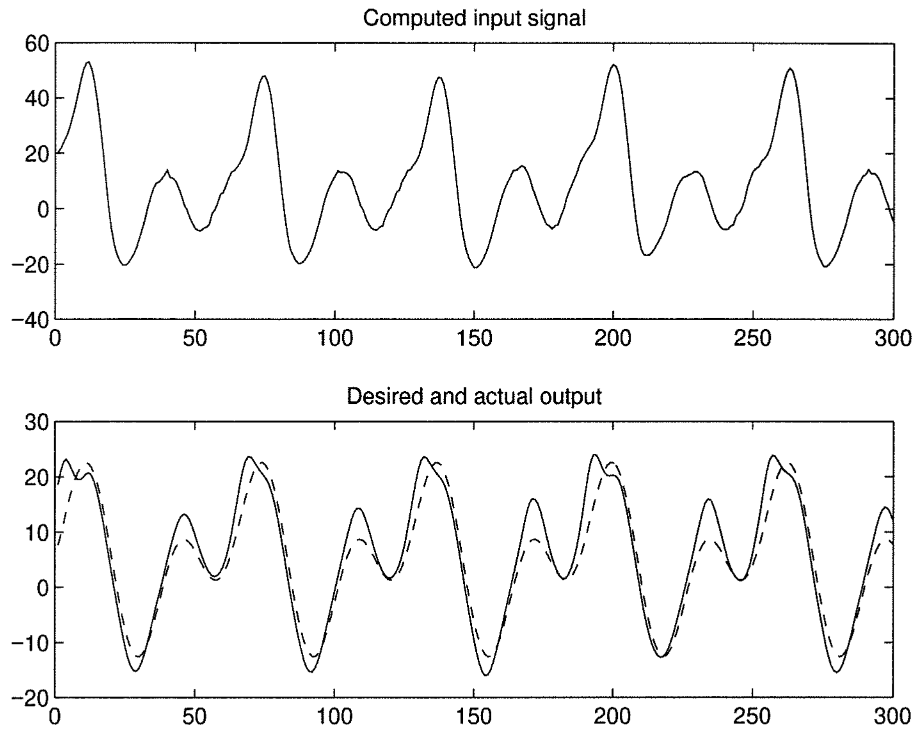


Figure 9.4: Output tracking for goal dynamics consisting of two periodic components.

desired output trajectory is achieved using only this open-loop trajectory planning technique. In Figure 9.5 the previous experiment is repeated for $\epsilon = 5.0$ in system (9.9). Here, the computed control sequence $u_2(t)$ does not accurately track the desired output trajectory. Since the system does not converge to the desired periodic output trajectory, it appears that the linearized dynamics around the desired trajectory may be unstable for this system. A possible remedy for this situation is to use a closed-loop feedback stabilization technique (possibly the OGY method) in a periodic manner to stabilize the dynamics about the desired trajectory.

9.4.2 Electronic circuit

In this section, the computational algorithm described in Sec.9.3 will be applied to control the dynamics of a nonlinear electronic circuit in a physical experiment. In the experiment, a low frequency (about 300 Hz) nonlinear circuit whose diagram is shown

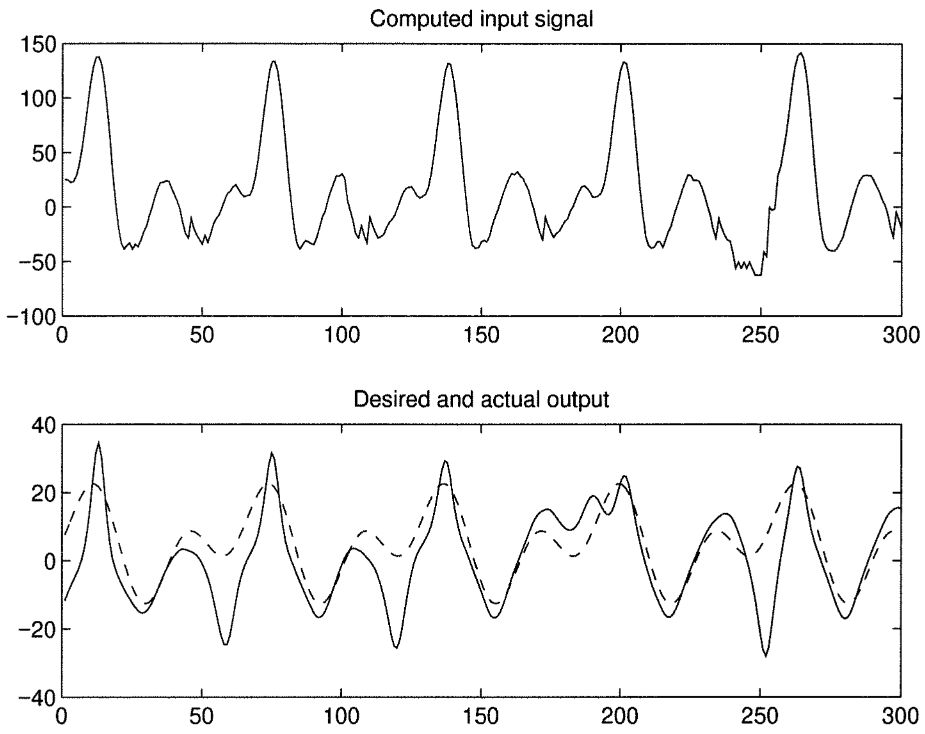


Figure 9.5: Tracking of the output signal of Figure 9.4, for system (9.9) with $\epsilon = 5.0$. In this case, open-loop control cannot successfully track the output signal.

in Figure 9.6 is used. The circuit consists of a nonlinear converter, linear feedback, and an input block. The nonlinear converter is implemented using OP amplifier U1A and the multiplier U2. The shape of the nonlinear function $F(w)$ generated by the converter was measured experimentally (see Figure 9.7a). The linear feedback contains three integrators (U3A, U3B, and U3C) and summers (U3D and U1B) which create three-dimensional phase space (x, y, z) of the nonlinear circuit. The input block is built using OP amplifiers U4A and U4B. This section takes signals $x(t)$ and the external input $u(t)$ to form the output $\epsilon(x(t) - u(t))$ which is applied to terminal A of the switch $SW1$. The value of the parameter ϵ is determined by the resistor R_{cont}.

When the switch $SW1$ is in position B, the circuit operates without external inputs $(w(t) = x(t))$ and generates chaotic oscillations. The projection of this chaotic attractor onto the plane $(x(t), y(t))$ is given in Figure 9.7b. When switch $SW1$ is in position A, the circuit is affected by the external input $u(t)$. In position A, the input to the nonlinear converter is $w(t) = x(t) - \epsilon(x(t) - u(t))$. In this experiment, $\epsilon = 0.8$

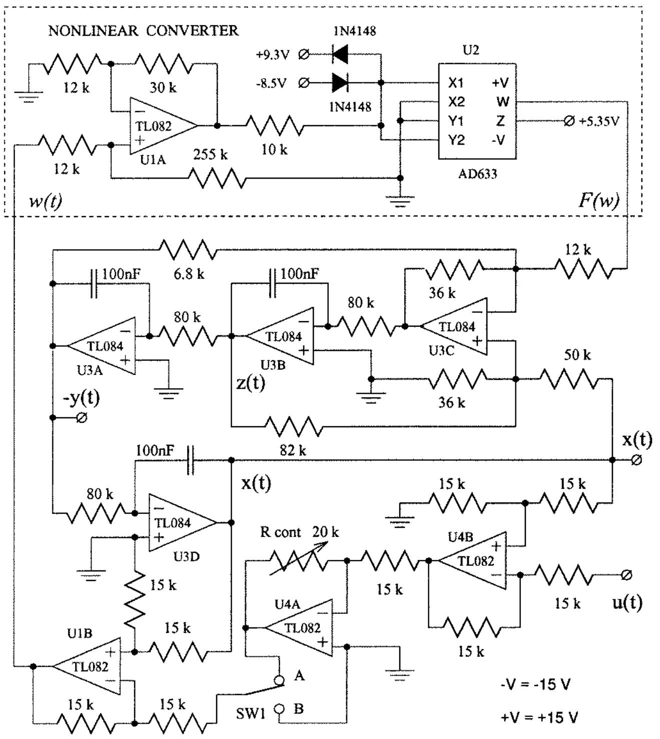
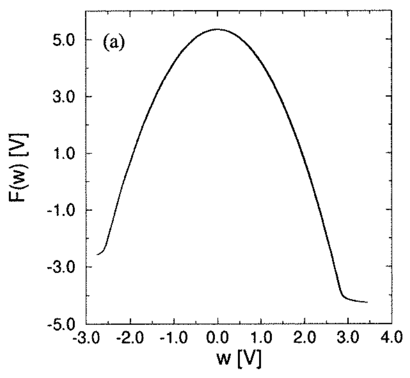


Figure 9.6: The diagram of the nonlinear circuit used in the experimental studies.

is used.

A band-limited random training signal $u_{\mathrm{train}}(t)$ is generated using a sampling rate of 1000 samples per second and this input is applied to the circuit. The input $u_{\mathrm{train}}(t)$ and output $-y_{\mathrm{train}}(t)$ of the system are recorded. This input-output time-series (30000 points) is used by the algorithm proposed before to construct the input $u(t)$ which will cause the nonlinear circuit to track a desired output trajectory $-y_0(t)$. A portion of the training time series is given in Figure 9.8. When the time series is analyzed by the input-output false nearest neighbors algorithm, it is found that the proper number of embedding terms is $l = 3$, $m = 2$.

The trajectory to be tracked for this example given in Figure 9.9 is non-periodic. The goal output trajectory consists of a sine wave with varying amplitude (time 50–400 and 900–1100), a constant value (400–700), and a piecewise linear signal (700–900). When this goal trajectory and the training set are input to the trajectory planning algorithm using a neighbor threshold distance $\delta = 0.25$, the result is the input sequence shown in Figure 9.10. Computing the entire input sequence (including



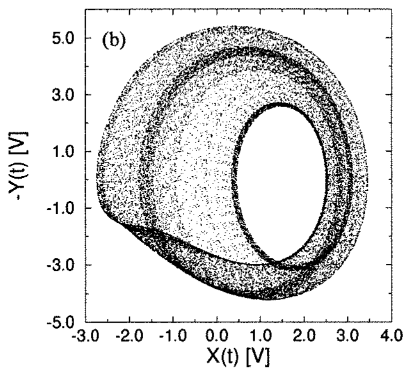


Figure 9.7: (a)-the shape of the nonlinear converter. (b)-the projection of the chaotic attractor measured from the circuit with the sampling rate 1000 sample/sec ($SW1$ in the position B)

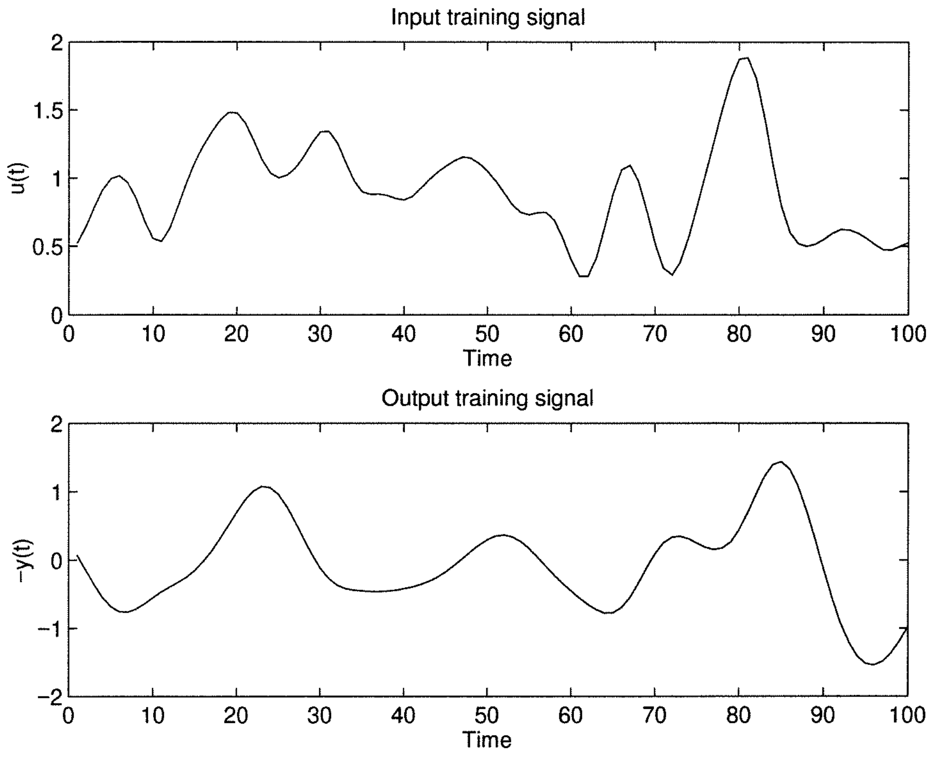


Figure 9.8: Portion of the training time-series for the nonlinear circuit.

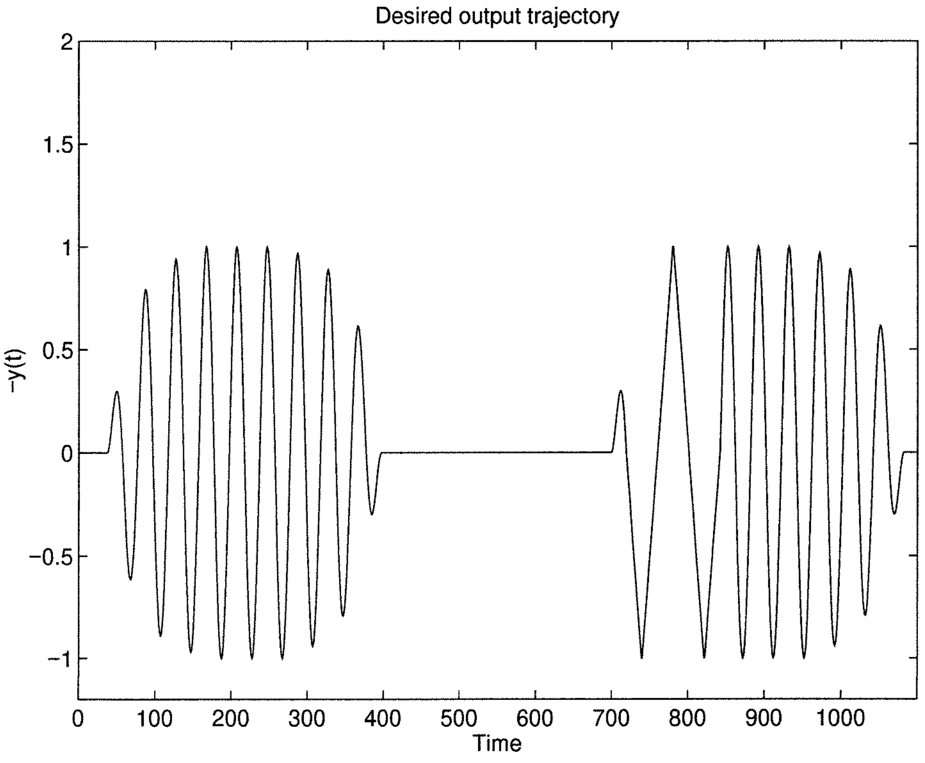


Figure 9.9: The desired goal trajectory for the nonlinear circuit.

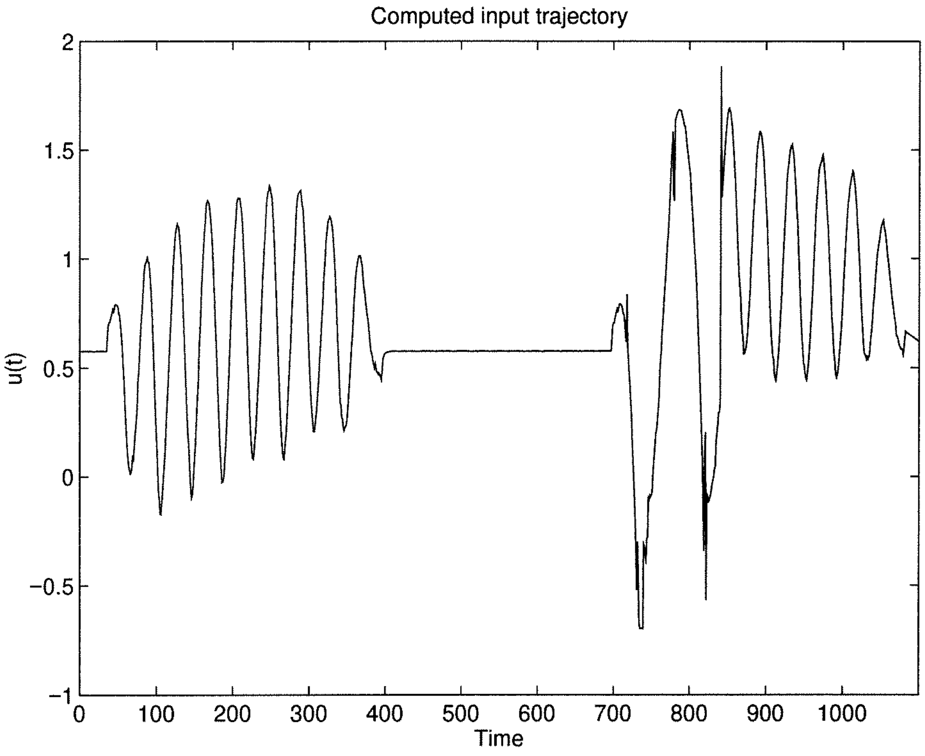


Figure 9.10: The computed input from the trajectory planning algorithm.

the sorting of the training data which is only necessary one time) takes less than 1 minute on a Sun SPARCstation 20. When this input sequence is used to drive the circuit, the measured output of the circuit tracks the goal trajectory quite well (see Figure 9.11). The major differences between the goal trajectory and system outputs occur during the transition which starts and ends the piecewise linear signal at times 700 and 850. Since closed loop control is not implemented in this example, the poor tracking at these times is not corrected online.

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

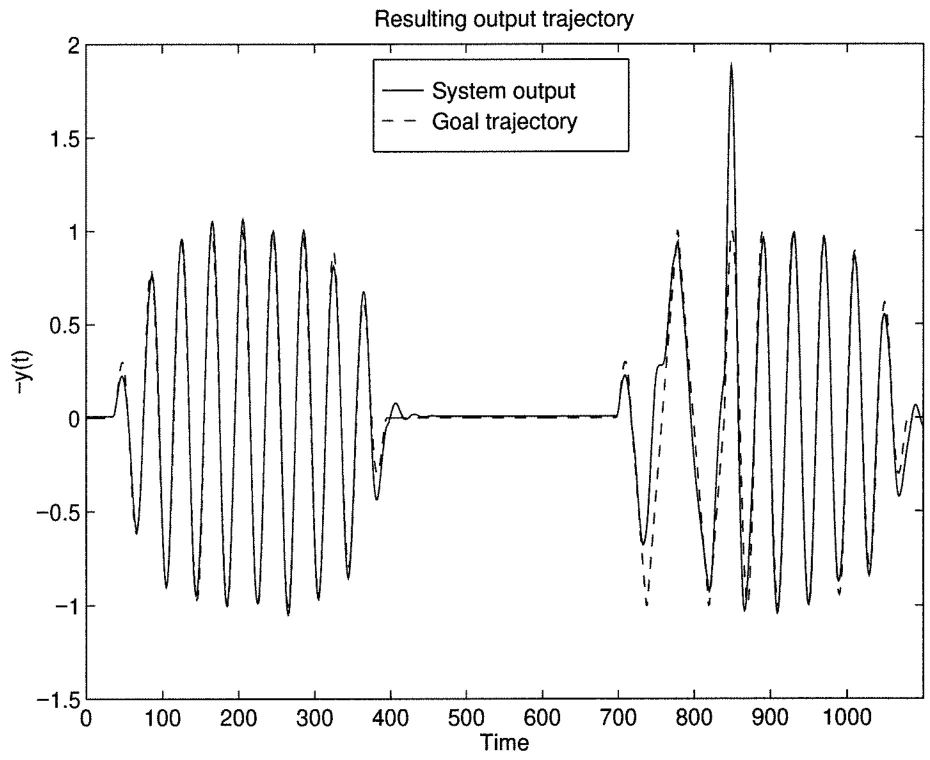


Figure 9.11: A comparison of the goal trajectory and the output of the circuit when the computed input from the trajectory planning algorithm is applied.

9.5 Inverse modeling and control of the Henon map

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

$$\begin{aligned} x_{n+1} &= 1 - ax_n^2 + y_n + u_n \\ y_{n+1} &= bx_n + cu_n, \end{aligned} \tag{9.10}$$

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

$$x_{n+1} = 1 - ax_n^2 + bx_{n-1} + u_n + cu_{n-1}. \tag{9.11}$$

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

$$u_{1,2} = (1 - c^2)^{-1}[x_{2,1} - 1 + ax_{1,2}^2 - bx_{2,1} - c(x_{1,2} - 1 + ax_{2,1}^2 - bx_{1,2})]. \tag{9.12}$$

Suppose we use the inputs $u_{1,2}$ from (9.12) and input them to the system in an open-loop fashion. With this choice of $u_{1,2}$ the periodic orbit of interest is a fixed point of the map

$$\mathbf{Y}_{2m+2} = \mathbf{P}_2(\mathbf{Y}_{2m}, u_{2m+1}, u_{2m}, u_{2m-1}) \equiv \mathbf{P}(\mathbf{P}(\mathbf{Y}_n, u_{2m}, u_{2m-1}), u_{2m+1}, u_{2m}) \tag{9.13}$$

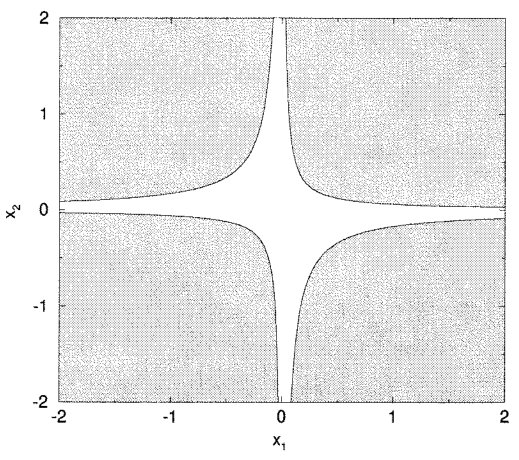


Figure 9.12: Region of stability of the open-loop control of Henon map (9.10) in the plane of parameters x_1, x_2 of the goal trajectory for $a = 1.4, b = 0.3$. (unshaded).

with $u_{2m\pm1} \equiv u_1$, $u_{2m} \equiv u_2$, $\mathbf{Y_n} \equiv \{x_n, x_{n-1}\}$. This solution can either be stable or unstable depending on the eigenvalues of the Jacobian of the system (9.13), or equivalently (9.11), calculated over a periodic orbit,

$$\mathbf{DP_2} = \begin{pmatrix} -2ax_1 & b \\ 1 & 0 \end{pmatrix} \begin{pmatrix} -2ax_2 & b \\ 1 & 0 \end{pmatrix}. \tag{9.14}$$

The region of stability on the plane (x_1, x_2) is defined by two sets of hyperbolas (Figure 9.12). Inside this region the map will be stable and the periodic control sequence (9.12) can be used in open-loop fashion. This region of stability is equivalent to the convergent regions of Ref.(Jackson, 1990). Outside the stability region, an additional closed-loop stabilization technique is needed.

A modification of the pole placement method can be used to stabilize this system (see, e.g. (Hong, Jie, Wang and Wang, 1996)). In addition to the two components of vector $\mathbf{Y}$, an additional linear equation for u_{2m} is added so the system to be analyzed (9.13) becomes:

$$\mathbf{Y}_{2m+2} = \mathbf{P}(\mathbf{Y}_{2m}, u_{2m+1}, u_{2m}, u_{2m-1})$$

$$u_{2m+2} = u_1 + \mathbf{K} \cdot \mathbf{Y}_{2m} + g\,(u_{2m} - u_1). \tag{9.15}$$

The control coefficients ($\mathbf{K} = (k_1, k_2)$ and g) can be calculated by specifying the eigenvalues of the Jacobian of this system(Hong et al., 1996). In this example, all three eigenvalues are placed at zero.

In Figure 9.13, two examples of controlling a period-2 orbit in the driven Henon map are shown. Figure 9.13a demonstrates the application of analytic open-loop control for the periodic trajectory $x_1 = 0.3, x_2 = 0.2$. The eigenvalues of the open-loop system about the desired trajectory are .978 and .092, so the open-loop system is stable. Figure 9.13b shows that open-loop control fails for the period 2 signal $x_1 = 0.3, x_2 = 0.7$. The eigenvalues of the open-loop system are 2.206 and .041, so the open-loop system is unstable and will not produce the desired output trajectory. However, by using pole placement feedback (closed-loop) control the desired trajectory can be stabilized as seen in Figure 9.13c.

In the previous examples, initial conditions in the neighborhood of the trajectory to be tracked were utilized. Under this assumption, the presented linear stability analysis should be valid. If the Henon map is driven under open loop control from arbitrary initial conditions, the system could settle on an undesired orbit as happen in Figure 9.13b where the map settles on a period-7 orbit. Simply waiting to start the control trajectory when the system comes close to the desired orbit may not be feasible, since the desired trajectory may not belong to the attractor of the undriven system. For these reasons, special care must be taken when bringing the system from arbitrary initial conditions to a desired trajectory.

The system could be brought from arbitrary initial conditions to the desired trajectory by computing the controllable sets of a point on the trajectory. The time n controllable set of a reference point in the embedded space consist of the set of points which can be controlled to that reference using exactly n input moves (Rhodes and Morari, 1996). By finding the smallest time controllable set of a point on the reference trajectory, the inputs needed to drive the system to a point on the reference trajectory in the shortest time are found.

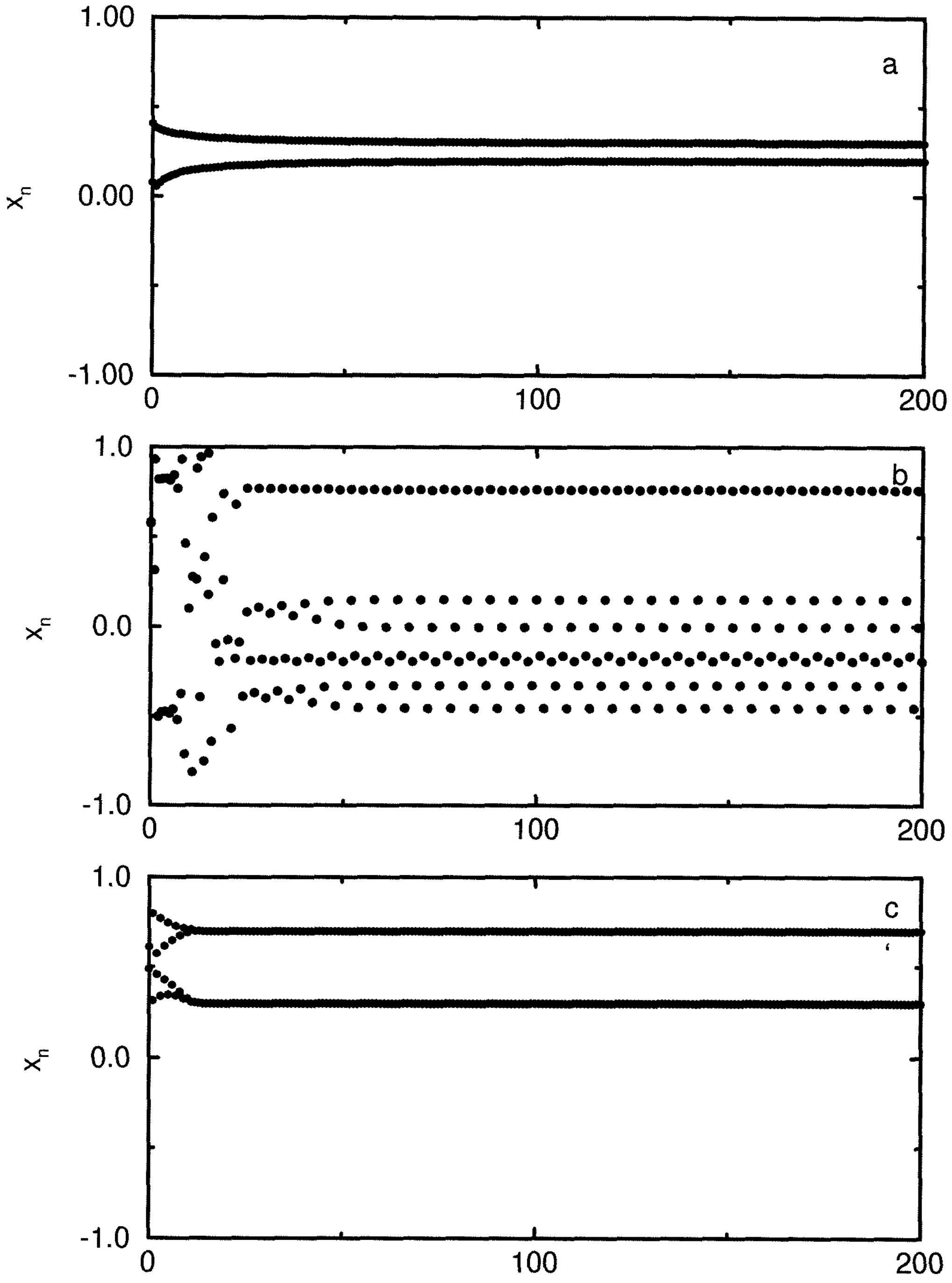


Figure 9.13: *a* - successful open-loop control of a period-2 orbit of the Henon map with $x_1 = 0.3, x_2 = 0.2$; *b* for $x_1 = 0.3, x_2 = 0.7$ - open loop control fails. *c* successful closed loop control for $x_1 = 0.3, x_2 = 0.7$ with feedback corrections calculated by the pole placement method using (9.15) to stabilize the goal orbit.

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

9.6 Conclusions

In this chapter an algorithm was presented which computes a control trajectory which will drive a nonlinear system such that a specified output trajectory is produced. This output trajectory does not need to coincide with a trajectory of the undriven system, and the algorithm which was presented for computing the trajectory relies only on time-series data. The main difference between our approach and other previous published "entrainment control" methods is that no *a priori* knowledge of the system is assumed. In addition, it is not assumed that the inputs directly affect all states of the system. We are interested in tracking a goal trajectory specified only in terms of a (scalar) output signal. Accordingly, instead of a full state control (which is rare in most physical systems of interest), it is assumed that only a single input to the system is available. When determining the proper input trajectory, a locally valid inverse model is constructed utilizing time-series data from a training set. The inverse model (9.3) is then used to determining the proper input trajectory in an open-loop one-step-ahead fashion.

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

Chapter 10 Work for the future in data-based control methods

10.1 Introduction

As the work detailed in this section of this thesis introduces the concept of data-based control, some thoughts and suggestions for future work in this field along with some ideas on how data-based control could be used in industry will be outlined. While the examples of Chapter 9 showed how data-based control methods could be used in an offline fashion, the ultimate goal of this research direction is to apply data-based control methods in an online fashion. In order to implement a data-based control scheme, two distinct computational pieces are needed. The first piece of the overall control scheme would use local models and a simple control strategy to track a known trajectory. This piece would be similar to the data-based trajectory tracking control scheme detailed in Chapter 9.

The second piece of the overall control scheme is a long-term trajectory computation scheme that would compute the trajectory to be tracked by the previously mentioned control method. The planned trajectory could be computed in a way that it satisfies conditions of optimality or feasibility if constraints happen to be present. This trajectory would also be determined using local models, and the method of computing controllable sets presented in Chapter 8 provides some ideas on how this could be achieved.

After these pieces are developed, it is possible that the overall process performing data-based control would be much simpler than the traditional control design process. The controller design process for traditional controllers involves the following steps:

1. Perform an identification experiment.

2. Form a model based on data from the identification experiment on the process.

3. Design a controller using the model as a reference.

4. Implement the controller on the process.

For systems exhibiting nonlinear behavior, it can be quite difficult to develop a parsimonious model of the dynamics. The process of identifying a model involves "compressing the data" in an attempt to capture the most important features of the data. While this is important for forming a model of reasonable size and may discard some of the noise associated with the data, it is always possible that valuable information is lost when forming a mathematical model of the dynamics in this manner. In addition, the field of nonlinear identification is not yet mature. Because there is no single accepted method of systematically identifying a model of a system, the process of identification for nonlinear models is an art as well as a science.

On the other hand, the process of developing data-based controllers could be much simpler.

1. Perform an identification experiment.

2. Implement data-based control using identification data as a reference.

One advantage of this method is that the difficult problem of identifying a nonlinear model could be skipped. Additionally, by allowing the control algorithm to utilize the data directly for control it may be possible to achieve better performance since more information about the process dynamics is available. However, data-based control also has some drawbacks. The question of computational time needed for online data-based controllers has not yet been addressed. Also, noise corruption and outliers could be problematic if not properly accounted for in the control scheme.

Since data-based schemes do not utilize a global model of the process, time-delay coordinates take the place of the state-space coordinates of the traditional approach. By assuming the discrete-time dynamics of the system can be represented in the following form:

$$y(t) = G[y(t-\tau), \ldots, y(t-l\tau), u(t-\tau), \ldots, u(t-m\tau)] \tag{10.1}$$

the time-delay coordinates of the system are defined as

$$\mathbf{z}(t) = [y(t), y(t-\tau), \ldots, y(t-(l-1)\tau), u(t-\tau), \ldots, u(t-m\tau)]. \tag{10.2}$$

As in Chapter 8, the dynamics of in the time-delay coordinates are governed by the relationship

$$\mathbf{z}(t+1) = F[\mathbf{z}(t), u(t)] \tag{10.3}$$

where F is defined in 8.10. Here it is shown that $\mathbf{z}(t{+}1)$ can be computed from $\mathbf{z}(t)$ and the control move $u(t)$. The trajectory tracking problem involves computing the input $u(t)$ needed to track a given reference trajectory $\mathbf{z}_{\text{ref}}(t)$. The trajectory computation problem involves finding a trajectory $\mathbf{z}(t)$ which meets certain conditions (endpoint constraints, the ability to satisfy constraints on u and y, optimality, etc.).

10.2 Using local models for trajectory tracking

It was shown in Chapter 9 that data-based methods can be used to calculate a control trajectory that tracks a desired output trajectory in an offline fashion. However, because of external disturbances and possible model/plant mismatch, it would be preferable to implement data-based trajectory tracking in an online fashion. In order to perform data-based tracking online, it is necessary to search the data for nearest neighbors, build a local model, and use this model to determine the next control move during each sampling time of the system. While this seems like quite an ambitious task, it should be possible since the data can be presorted and only relatively simple computations are needed for the control computations.

The first step in testing this online implementation of the data-based tracking scheme would be to implement the method presented in Chapter 9 as an online controller for a simulated system. To perform this study, computer code would have to be written to link the program that computes the data-based control moves to a simulation of the process. Since the data-based control method involves a number of data searches, MATLAB may not be the best choice for these implementing the control algorithm. In my experience, MATLAB is somewhat slow in performing searches and this is one reason why the code for the offline data-based control planning example problems is written in FORTRAN. The implementation is relatively fast; as an example, the computation of the control trajectory for the electrical circuit experiment takes less than 1 minute on a Sun SPARCstation 20. This time includes the presorting algorithm and the computation of the control trajectory which consists of 1100 samples. In practice, the data could be stored in a presorted manner and sorting would need to be performed only one time for each data set.

If the results of these trials are successful, the next step would involve testing the method on an experimental system. Trials on an experimental system would involve writing software that could operate in real time. These trials would be important, since they would indicate the computational time needed to compute a single control move using the data-based control algorithm along with a real-time control processing system. In addition, in these trials the suitability of using this method in conjunction with data from physical processes could be further tested. Realistic problems associated with noise corruption and data outliers would be encountered and possible solutions to these problems could be tested. It is expected that systems which might be suitable for data-based control would have a fairly large sampling time, and this should be taken into account when choosing a trial process for this experimental work. The work of Schaal and Atkeson (1994) in "memory-based learning" control leads us to believe that real time application of these control schemes should be successful.

Some theoretical issues also should be examined for the proposed trajectory tracking algorithm. The first problem to be addressed is the question of stability. As was shown in Chapter 9, the data-based controller in its present formulation may cause the system to become unstable if the process exhibits nonminimum-phase behavior. This is a major drawback, especially since a model of the process dynamics does not need to be explicitly derived before applying the control scheme. However, it might be possible to modify the control objective slightly such that the closed-loop behavior of the system is stable.

Some hints in this direction are given in a recent paper of Tan and Cauwenberghe (1996). In this paper a stable one-step-ahead control scheme is built which uses neural network models. It is assumed that the neural network model of the dynamics is accurate, and the following cost function is considered:

$$J = [y_{\text{ref}}(t+1) - y_m(t+1)]^2 + \alpha[\Delta u(t)]^2 \tag{10.4}$$

where α is an adjustable parameter, $\Delta u(t) = u(t) - u(t-1)$, $y_{\text{ref}}(t+1)$ is the trajectory to be tracked, and $y_m(t+1)$ is the prediction of the neural model. This is different than the objective considered in the previously presented method for computing controllable sets which only considers the error associated with the control move.

It is then shown in Tan and Cauwenberghe (1996) that the control move can be computed by using a gradient descent method, and that the algorithm converges using this method. While the discussion is specific to neural network models in the method of computing the gradient, the same quantities could be computed from local models. Stability is then shown for the one-step-ahead control scheme using a standard Lyapunov function. While the results given in Tan and Cauwenberghe (1996) are specific to neural networks, it is likely that the results could be modified and extended to systems described by local models under the proper assumptions.

Another area of theoretical research to be examined for control trajectory tracking algorithms is the problem of robustness. The problem of robustness is a bit different than the situation considered in other control methods. Traditionally, a major source of plant/model mismatch arises from attempting to describe a nonlinear system by a linear model. Attempting to represent a system by a simplified model is another source of plant/model mismatch. However, these issues may not be problematic for local models unless the process dynamics happen to change from those encountered during the identification experiment. When using local models it is hoped that enough data are available to fill out the space such that local linear approximations are valid. For data-based control problems, the major reason to consider robustness is the presence of noise and outliers in the data.

Since a data-based control scheme uses local models, only a small portion of the data is used when forming any single local model. The presence of noise or outliers in this small data set could cause large errors in the prediction. Probably the most effective way to deal with this problem would be some type of preconditioning of the data. Outliers could be removed from the data and the noise could be reduced using nonlinear methods which involving projections (Grassberger et al., 1993). This should be the most efficient way of dealing with the problem of "robustness". Making changes in the control algorithm for robustness would likely involve some type of worst-case analysis, and would probably add significantly to the computational time needed by the controller for computations.

10.3 Trajectory computation

The trajectory computation portion of the data-based controller would compute the reference trajectory needed by the tracking scheme outlined in the previous section. In the data-based methodology, the reference trajectory is computed using identification data from the system as a reference. As will be shown, the trajectory could be computed such that it will meet certain optimality conditions or such that it is a feasible trajectory in the presence of constraints.

There are two problems which can be defined for trajectory generation. The first problem, which was discussed in Chapter 8, is determining the trajectory which ends at a given reference location in the time-delay coordinates. In order to compute this trajectory, the method of computing controllable sets which was presented in Chapter 8 can be used. Say it is desired that the system should be driven to $\mathbf{z}_{\text{ref}}$ in the time-delay coordinates. The set of trajectories which are driven to $\mathbf{z}_{\text{ref}}$ in n sampling times is defined as the time n-controllable set of $\mathbf{z}_{\text{ref}}$, $\mathcal{C}_n(\mathbf{z}_{\text{ref}})$. The trajectory that produces the desired output $\mathbf{z}_{\text{ref}}$ is also computed by the algorithm which determines the controllable sets. The set of trajectories associated with that controllable set can be defined as $c_n^j(\mathbf{z}_{\text{ref}})$ where $j = 1, \ldots, p$ and p is the number of trajectories found by the algorithm (the number of trajectories will depend on the step size used by the algorithm).

The controllable sets are computed in a recursive fashion, which makes the problem a good candidate for harnessing the power of parallel processing. The controllable set $\mathcal{C}_{n+1}(\mathbf{z}_{\text{ref}})$ is found by computing the time 1 controllable set of a grid of points contained in $\mathcal{C}_n(\mathbf{z}_{\text{ref}})$. Using code which is able to harness the power of parallel processors, the computational time associated with computing the controllable sets could be dramatically reduced when compared to the current implementation. Computational time would be a major concern if the controllable sets were to be calculated online.

A possible extension of the method of computing controllable sets involves considering an objective function. Assume that the the system is currently located at $\mathbf{z}_0$ in the time-delay coordinates. Then the set of trajectories such that $\mathbf{z}_0$ is driven to $\mathbf{z}_{\text{ref}}$ is defined by $c_j^i(\mathbf{z}_{\text{ref}}) = [\mathbf{z}_0, u_1, \ldots, u_i]$ for $i = n, \ldots, \infty$ where n is the smallest time controllable set which contains $\mathbf{z}_{\text{ref}}$ and j is the index for trajectories of a given time. Note that the point $\mathbf{z}_0$ might also be located in any controllable sets with a time greater than n. In addition, there may be numerous trajectories associated with $\mathbf{z}_0$ in a given controllable set. With these trajectories which all have $\mathbf{z}_{\text{ref}}$ as an endpoint, how should the trajectory to be implemented be chosen?

It is possible to search these trajectories can such that some condition is optimized. The time n controllable set trajectory of $\mathbf{z}_{\text{ref}}$ associated with $\mathbf{z}_0$ as the initial condition is the trajectory which reaches $\mathbf{z}_{\text{ref}}$ in least time. Another method would be to search the trajectories in the controllable set such that an objective of the form

$$J_r(\mathbf{z}_0) = \sum_{k=0}^{r-1} e^T(k) Q e(k) + \sum_{k=0}^{r-1} R u^2(i) \tag{10.5}$$

is minimized where Q and R are weights, $e = \mathbf{z} - \mathbf{z}_{\text{ref}}$, and r is the maximum horizon of interest which will be searched.

Since controllable sets are computed in a recursive fashion, it is possible to efficiently compute the objectives associated with the trajectories. Trajectories in the time $n+1$ controllable set are computed directly from a grid of points contained in the time n controllable set, so only the last term in the objective function would need to be added to the time n objective function. Further computational improvements could be gained by searching for the smallest objective in the following manner. By sorting the objective functions associated with trajectories in the time i controllable set, only those trajectories with small objective functions would have the associated $i+1$ controllable set trajectory computed. In this way, those trajectories with large objective functions could be discarded and the entire controllable set would not need to be computed.

Another feature of the data-based method of computing controllable sets is that input and output constraints can be easily met. The continuation for computing the controllable set involves solving the a nonlinear equation in terms of $[y(t-(l-1)\tau), u(t-(m-1)\tau)]$ (see Section 8.4.3). By only searching the region in the space of $[y(t-(l-1)\tau), u(t-(m-1)\tau)]$ which satisfy the constraints, only feasible trajectories will be found. Another approach for systems with inputs constraint is to gather identification data from an experiment where the input constraints are enforced. Since local model require nearby data, only feasible trajectories can be found by the algorithm.

The second problem which could be considered in trajectory computation is the problem of computing reachable sets. The time n reachable set of a reference $\mathbf{z}_{\text{ref}}$ is defined as the set of points in the time delay coordinates which can be reached by making n control moves. This is essentially the dual to the previously considered problem, and many of the same continuation methods could be used for computation of these reachable sets. This algorithm might be useful for determining feasible trajectories for optimizing certain quantities of the process. One example of where reachable sets could be used is in the control of batch processes. The output trajectory which optimizes some final conditions relating to the final product could be identified in this manner. Once again, the system constraints can be easily enforced.

10.4 Other possible extensions

The previously described schemes rely exclusively on open-loop identification data for control computation. However, the conditions under which processes are operated may change with time. These new conditions may bring about new process dynamics which don't match the dynamics of the identification experiment. If the data-based control scheme relies on the old identification data to control this system, there may be major problems of model/plant mismatch.

However, as the process is running it is also producing dynamic data. For this reason, it might be feasible to collect data from the process online and continuously update the database which the local models rely upon for prediction. The one problem with this idea is that closed-loop identification can be much more difficult than open-loop identification because of the small amount of signal present in closed-loop data (Söderström and Stoica, 1989). However, this type of approach would be very appealing to industry since the down-time associated with data-producing identification experiments needed by the data-based control scheme would be dramatically reduced. In addition, the data-based control scheme may be able to "adapt" to changing process conditions by continually updating the reference data.

Another field in which the data-based approach could be applied is process monitoring. Since the dynamics of the system can be represented by Equation (10.1), the exists a surface (1-dimensional manifold) in the space $[y(t), y(t-\tau), \ldots, y(t-l\tau), u(t-\tau), \ldots, u(t-m\tau)]$ on which the dynamics should lie. If the dynamics should happen to change as a result of some disturbance to the process conditions, the function G may no longer describe the system dynamics. In this case, the data from the new perturbed process dynamics no longer lie on the surface defined by $[y(t), y(t-\tau), \ldots, y(t-l\tau), u(t-\tau), \ldots, u(t-m\tau)]$ from the identification data. By analyzing the distance of the process data from this surface in an online fashion, abrupt changes in the process dynamics could be detected.

To gain an insight as to how these process monitoring methods would work, literature in the field of nonlinear noise reduction would provide some insight. In noise reduction, the surface $[y(t), y(t-\tau), \ldots, y(t-l\tau), u(t-\tau), \ldots, u(t-m\tau)]$ is used as a reference. Outliers are then projected onto this surface using local linear projection methods (Kostelich and Yorke, 1988). In process monitoring, a different problem would be analyzed, but the insight needed to solve the problem is similar. The classification of outliers would need to be made online, and if outliers are detected some type of classification system for the faults would be needed.

10.5 Conclusions

Some possible future work in the field of data-based control was presented in this chapter. While there is still a great deal of work to be completed in this area before any industrial application of these methods is possible, the underlying motivation is quite strong. Data are now widely available in industry, and should be put to better use for control purpose. With advances in computational speed and storage capacity, researching new ways to use this data for control purposes is necessary. Continually, new classes of control methods are be developed such that better performance is achieved and the control design and maintenance process is simplified. Possibly data-based control methods can play an important role in the future in the push for better, more efficient means of control.

Part V

Summary and suggestions for future work

Chapter 11 Summary and suggestions for future work

11.1 Summary of contributions

In the first portion of this thesis, the problem of forming models suitable for process control is examined. These models are needed so that newly developed methods of nonlinear control can be applied to industrial problems. The methods presented here are specifically designed to work with nonlinearities, as a majority of real world processes to be controlled exhibit nonlinear dynamics. Problems in the fields of black-box and first-principles modeling are addressed. Three distinct problems are examined in the thesis.

First, the false nearest neighbors algorithm is presented as a method for determining the proper regression vector size for nonlinear black-box modeling. The algorithm is computationally efficient, and relies on a geometric criterion for determining the proper regression vector. Because a global model of the dynamics is not needed, this method can be used in conjunction with any functional modeling scheme. Problems specific to input/output systems and noise corrupted time-series are analyzed for the first time and a new threshold test is proposed in order to correct problems associated with noise corruption. In addition, a review of time-delay embedding theorems for autonomous and input/output system is given.

The problem of nonlinear model reduction is considered next. This subject is important since models that are derived from first-principles are commonly of high order. At the same time, these high-order models can exhibit behavior consistent with low-order systems. In order to be useful for nonlinear control design algorithms, these models need to be reduced in some way. The specific problem of model reduction for singularly perturbed systems is examined in this thesis. The algorithm of Maas and Pope (1992) is presented as a method of computing the reduced-order manifold of slow dynamics, and it is proven rigorously for the first time that this algorithm correctly identifies this manifold. A new method of model reduction that uses the results of the Maas and Pope (1992) algorithm is then introduced.

In the last portion of the thesis, the concept of data-based control is introduced. The idea behind this concept is to use a new breed of models known as local models which utilize large amounts of data. With recent advances in computer storage and speed, industry is able to collect large amounts of process data. The challenge is to determine how that data can be used effectively. The data-based control ideas presented here attempt to use the data directly in the control process. Methods of trajectory tracking and computing controllable sets are presented, and some thoughts on possible future research in this field are presented. However, there is still a great deal of work to be done in this area before these methods can be successfully applied to industrial problems.

11.2 Future work

There are still a number of open research problems in the area of process modeling and identification for control purposes. An overview of the work for the future in the area of data-based control was presented in Chapter 10, so those points will not be repeated here. Some other subjects requiring future work include:

Computationally efficient and accurate nonlinear model reduction The method of model reduction presented in Chapter 6 is a fairly computationally efficient method; however, it is not completely accurate method for forming reduced models as was shown in the two-phase CSTR example. On the other hand, the method of Roussel and Fraser (1991) is very accurate since the algorithm converges to the proper reduced system. The problem with this method is the large computational effort required for performing the necessary symbolic manipulations. In order to combine the best qualities of these two algorithms the following approach is suggested.

1. Use the algorithm of Maas and Pope (1992) to compute an initial guess for the slow manifold.
2. Approximate the algorithm of Roussel and Fraser (1991) by representing the slow manifold using a numerical approximation rather than a symbolic description. In this way, the method could be made more computationally appealing

As mentioned before, there is some current work in this direction (Davis, 1997).

Identifying functional relationships While the problem of determining the proper model order for nonlinear black-box models was addressed in Chapters 2-5, the problem of determining the functional relationship between the regression vector (past) and the output (future) was not addressed. Mathematically, the function G in the relationship

$$y(t) = G[\psi(t)] \tag{11.1}$$

should be estimated from identification data of the system.

There have been numerous papers on this topic using a variety of approaches to solve this problem including neural networks, splines, recursive partitioning, radial basis functions, and other methods; however, none of these methods take the approach of constructing an optimal set of orthogonal basis functions which best approximate the mapping G. This would be the best approach to solving the problem, and some ideas on how an optimal orthogonal basis might be constructed are presented in Mao and Billings (1996). This is the most appealing way to solve the problem.

Building models that include uncertainty descriptions All models will have some amount of uncertainty associated with them depending on both the model structure used and the quality and quantity of the data available for forming the identified model. The amount of uncertainty associated with the model may also be dependent on the location of the regression vector. A model which included some type of bounds on the uncertainty associated with the model may be very useful for developing robust controllers.

Understanding results of existing nonlinear identification methods Many methods of nonlinear identification result in models which are difficult or impossible to interpret after they are constructed. While these models may reproduce the dynamical behavior of the process extremely well, it would be preferable to gain insight into the process dynamics as a byproduct of modeling the system. This insight can be helpful when designing controllers, so performing these studies should prove valuable. New methods of nonlinear identification should also be developed such that insight is gained during the modeling process.

The MARS identification scheme (Friedman, 1991) is one example of a modeling scheme that gives some understanding of the dynamics during the modeling phase. As the model is developed, the basis functions and variables utilized by these basis functions are given to the user. In addition, the predictive capability of each of the variables contained in the regression vector is given and unnecessary portions of the model are "trimmed" away in the final stage to reduce the complexity of the model. These are the types of features which can be helpful when attempting to understand the system dynamics.

Bibliography

Abarbanel, H. D. I. (1994). Tools for analyzing observed chaotic data, *in* A. Guran and D. J. Inman (eds), *Stability, Vibration, and Control of Structures*, Kluwer Acedemic Publishers.

Abarbanel, H. D. I. (1996). *Analysis of Observed Chaotic Data*, Institute for Nonlinear Science, Springer-Verlag, New York.

Abarbanel, H. D. I., Brown, R., Sidorowich, J. J. and Tsimring, L. S. (1993). The analysis of observed chaotic data in physical systems, *Rev. Mod. Phys.* **65**: 1331–1392.

Balasubramanian, R. (1989). *Continuous Time Controller Design*, IEE Control Engineering Series, Peregrinus, London.

Beiman, L. (1993). Hinging hyperplanes for regression, classification, and function approximation, *IEEE Transactions on Information Theory* **39**: 999–1012.

Box, G. E. P. and Jenkins, G. M. (1970). *Time Series Analysis Forecasting and Control*, Holden-Day, San Francisco.

Boyd, S. and Chua, L. O. (1985). Fading memory and the problem of approximating nonlinear operators with volterra series, *IEEE Transactions on Circuits and Systems* **32**: 1150–1161.

Breeden, J. L. (1994). Open-loop control of nonlinear-systems, *Physics Letters A* **190**: 264–272.

Brogan, W. L. (1991). *Modern Control Theory*, Prentice Hall, Enlgewood Cliffs, New Jersey.

Casdagli, M. (1989). Nonlinear prediction of chaotic time series, *Physica D* **35**: 335–356.

Casdagli, M. (1992). A dynamical systems approach to modeling input-output systems, *in* M. Casdagli and S. Eubank (eds), *Nonlinear Modeling and Forecasting*, Vol. XII of *A Proceedings Volume in the Santa Fe Institute Studies in the Sciences of Complexity*, Addison-Wesley, pp. 265–281.

Chen, C. C. (1996). Direct chaotic dynamics to any desired orbits via a closed-loop control, *Physics Letters A* **213**: 148–154.

Chen, S. and Billings, S. A. (1989). Representations of non-linear systems: the NARMAX model, *Int. J. Control* **49**(3): 1013–1032.

Cutler, C. R. and Ramaker, B. L. (1980). Dynamic matrix control– A computer control algorithm, *Joint Automatic Control Conf.*, San Francisco, California.

Davidson, J. F. (1956). The transient behavior of plate distillation columns, *Trans. Inst. Chem. Engrs.* **34**: 44–52.

Davis, M. J. (1997). Personal Communication.

del Re, L., Kraus, F., Schultheiss, J. and Gerber, H. (1994). Self-tuning PID controller for lower-limb FES with nonlinear compensation, *Proceedings of the American Control Conference*, Vol. 2, pp. 2015–2019.

Doedel, E. J. (1986). *AUTO: Software for Continuation and Bifurcation Problems in Ordinary Differential Equations*, Concordia University: Montreal, Canada.

Doyle, F. J., Ogunnaike, B. A. and Pearson, R. K. (1995). Nonlinear model-based control using second-order Volterra models, *Automatica* **31**: 697–714.

Eichler, J. and Mansour, M. (1979). A sub-optimal control scheme of non-linear systems based on the stored response modelling technique, *Regelungstechnik* **27**: 220–224.

Enns, D. (1984). *Model Reduction for Control System Design*, PhD thesis, Stanford University.

Farmer, J. D. and Sidorowich, J. J. (1987). Predicting chaotic time series, *Phys. Rev. Lett.* **59**: 845–848.

Fenichel, N. (1979). Geometric singular perturbation theory for ordinary differential equations, *Journal of Differential Equations* **31**: 53–98.

Fenichel, N. (1983). Oscillatory bifurcations in singular perturbation theory, II. fast oscillations, *SIAM Journal on Mathematical Analysis* **14**: 868–874.

Finesso, L. (1990). *Consistant Estimation of the Order for Markov and Hidden Markov Chains*, PhD thesis, University of Maryland.

Franklin, G. F., Powell, J. D. and Emami-Naeini, A. (1986). *Feedback Control of Dynamic Systems*, Addison-Wesley, Reading, Mass.

Fredkin, D. R. and Rice, J. A. (1995). Method of false nearest neighbors - a cautionary note, *Physical Review E* **51**: 2950–2954.

Friedman, J. H. (1991). Multivariate adaptive regression splines, *The Annals of Statistics* **19**(1): 1–141.

Friedman, J. H., Bentley, J. L. and Finkel, R. A. (1977). $K-d$ tree, *ACM Trans. Math. Software* **3**: 209–226.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1994). User's guide for NPSOL 5.0: A FORTRAN package for nonlinear programming, *Technical report*, Stanford University.

Grassberger, P. (1990). An optimized box-assisted algorithm for fractal dimension, *Physics Letters A* **148**: 63–68.

Grassberger, P., Hegger, R., Kantz, H., Schaffrath, C. and Schreiber, T. (1993). On noise reduction methods for chaotic data, *Chaos* **3**: 127–141.

Guillemin, V. and Pollack, A. (1974). *Differential Topology*, Prentice-Hall.

Harg, K. (1997). Computers, models, and the real world. Plenary talk at PSE'97-ESCAPE7.

Harris, T. J., MacGregor, J. F. and Wright, J. D. (1980). Optimal sensor location with an application to a packed bed tubular reactor, *AIChE J.* **26 (6)**: 910–916.

Ho, M. C., Chern, J. L. and Wang, D. P. (1994). Creating the desired output waveform in nonlinear oscillators with modulations, *Physics Letters A* **194**: 159–164.

Hong, Z., Jie, Y., Wang, J. and Wang, Y. H. (1996). General-method of controlling chaos, *Physical Review E* **53**: 299–306.

Horn, R. A. and Johnson, C. R. (1985). *Matrix Analysis*, Cambridge University Press, Cambridge.

Hübler, A. (1989). Adaptive-control of chaotic systems, *Helvetica Physica Acta* **62**: 343–346.

Hunter, Jr., N. F. (1992). Application of nonlinear time-series models to driven systems, *in* M. Casdagli and S. Eubank (eds), *Nonlinear Modeling and Forecasting*, Vol. XII of *A Proceedings Volume in the Santa Fe Institute Studies in the Sciences of Complexity*, Addison-Wesley, pp. 467–491.

Hwang, Y. (1991). Nonlinear wave theory for dynamics of binary distillation columns, *AIChE Journal* **37**(5): 705–723.

Hwang, Y. (1995). Wave propagation in mass-transfer processes: From chromatography to distillation, *Ind. Eng. Chem. Res.* **34**: 2849–2864.

Isidori, A. (1989). *Nonlinear Control Systems*, Springer-Verlag.

Jackson, E. A. (1990). The entrainment and migration controls of multiple-attractor systems, *Physics Letters A* **151**: 478–484.

Jones, C. (1994). Geometric singular perturbation theory, *in* R. Johnson (ed.), *Dynamical Systems*, number 1609 in *Lecture Notes in Mathematics*, Springer, Montecatini Terme, pp. 45–118.

Joseph, B. and Brosilow, C. B. (1978). Inferential control of processes: Part I. steady state analysis and design, *AIChE Journal* **24**(3): 485–492.

Kennel, M. B., Brown, R. and Abarbanel, H. D. I. (1992). Determining embedding dimension for phase space reconstruction using a geometrical reconstruction, *Phys. Rev. A.* **45**: 3403–3411.

Khalil, H. K. (1996). *Nonlinear Systems*, second edn, Prentice Hall, Inc., Upper Saddle River, NJ.

Kocarev, L. and Parlitz, U. (1996). Generalized synchronization, predictability, and equivalence of unidirectional coupled dynamical systems, *Physical Review Letters* **76**: 1816–1819.

Koprowski, G. (1997). NBA teams look for help from the digital sixth man, Wall Street Jounal Interactive Edition (http://www.wsj.com). September 13.

Kostelich, E. J. and Yorke, J. A. (1988). Noise reduction in dynamical systems, *Phys. Rev. A* **38**: 1649–1652.

Lee, J. H. and Morari, M. (1991). Robust measurement selection, *Automatica* **27**: 519–527.

Lévine, J. and Rouchon, P. (1991). Quality control of binary distillation columns via nonlinear aggregated models, *Automatica* **27**: 463–480.

Ljung, L. (1986). *MATLAB System Identification Toolbox User's Guide*, The Mathworks, Inc., Sherborn, Massachusetts.

Ljung, L. (1987). *System Identification: Theory for the User*, Prentice-Hall Information and System Sciences Series, Prentice-Hall, Inc., Englewood Cliffs, NJ.

Lorenz, E. N. (1969). Atmospheric predictability as revealed by naturally occurring analogues, *J. Atmos. Sci.* **26**: 636.

Maas, U. and Pope, S. B. (1992). Simplifying chemical kinetics: Intrinsic low-dimensional manifolds in composition space, *Combustion and Flame* **88**: 239–264.

Mao, K. Z. and Billings, S. A. (1996). Algorithms for minimal model structure detection in nonlinear dynamic system identification, *Technical Report 635*, University of Sheffield, Sheffield, United Kingdom.

MathWorks, Inc. (1992). *MATLAB Reference Guide*, Natick, Mass.

Meadows, E. S. and Rawlings, J. B. (1997). Model predictive control, *in* M. A. Henson and D. E. Seborg (eds), *Nonlinear Process Control*, Prentice-Hall Inc., Upper Saddle River, NJ, pp. 233–310.

Mejdell, T. and Skogestad, S. (1991). Estimation of distillation compositions from multiple temperature measurements using partial-least-squares regression, *Ind. Eng. Chem. Res.* **30**: 2543–2555.

Mettin, R., Hubler, A., Scheeline, A. and Lauterborn, W. (1995). Parametric entrainment control of chaotic systems, *Physical Review E* **51**: 4065–4075.

Moczek, J. S., Otto, R. E. and Williams, T. J. (1965). Approximation model for the dynamic response of large distillation columns, *Chem. Eng. Progress Symp. Series* **61**: 136–146.

Moore, B. C. (1981). Principal component analysis in linear systems: Controllability, observability and model reduction, *IEEE Trans. Autom. Control* **26**: 17–32.

Moore, C., Hackney, J. and Canter, D. (1987). Selecting sensor location and type for multivariable processes, *in* D. M. Prett and M. Morari (eds), *Shell Process Control Workshop*, Butterworth publishers, Boston, pp. 291–308.

Morari, M. and Lee, J. H. (1997). Model predictive control: Past, present, and future, *PSE'97-ESCAPE7 Symposium.*

Morari, M. and Zafiriou, E. (1989). *Robust Process Control*, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Ogunnaike, B. A. (1995). Personal Communication.

Ott, E., Grebogi, C. and Yorke, J. A. (1990). Controlling chaos, *Phys. Rev. Lett.* **64**: 1196–1199.

Packard, N. H., Crutchfield, J. P., Farmer, J. D. and Shaw, R. S. (1980). Geometry from a time series, *Phys. Rev. Lett.* **45**: 712.

Peng, B., Petrov, V. and Showalter, K. (1992). Controlling low-dimensional chaos by proportional feedback, *Physica A* **188**: 210–216.

Poncet, A. and Moschytz, G. S. (1994). Optimal order for signal and system modeling, *Proc. IEEE International Symposium on Circuits and Systems*, Vol. 5, London, pp. 221–224.

Poncet, A., Poncet, J. L. and Moschytz, G. S. (1995). On the input-output approximation of nonlinear systems, *Proc. IEEE International Symposium on Circuits and Systems*, Vol. 2, Seattle, pp. 1001–1004.

Pyragas, K. (1992). Continuous control of chaos by self-controlling feedback, *Physics Letters A* **170**: 421–428.

Rhodes, C. and Morari, M. (1995). Determining the model order of nonlinear input/output systems directly from data, *Proceedings of the American Control Conference*, Seattle, pp. 2190–2195.

Rhodes, C. and Morari, M. (1996). Determining controllable sets from a time-delay description, *13th World Congress of International Federation of Automatic Control*, Vol. M, San Francisco, pp. 13–18.

Rosenbrock, H. H. (1957). An investigation of the transient response of a distillation column, Part I: Solution of the equations, *Trans. Inst. Chem. Engrs.* **35**: 347–351.

Roussel, M. R. and Fraser, S. J. (1991). On the geometry of transient relaxation, *J. Chem. Phys.* **94**: 7106–7113.

Ruelle, D. and Takens, F. (1971). On the nature of turbulence, *Comm. Math. Phys.* **20**: 167–192.

Sågfors, M. F. and Waller, K. V. (1995). Dynamic low-order models for capturing directionality in nonideal distillation, *Ind. Eng. Chem. Res.* **34**: 2038–2050.

Sandberg, I. (1991). Approximation theorems for discrete-time-systems, *IEEE Transactions on Circuits and Systems* **38**: 564–566.

Sauer, T. (1993). Time series prediction by using delay coordinate embedding, *in* A. S. Weigend and N. A. Gershenfeld (eds), *Time Series Prediction: Forecasting the Future and Understanding the Past*, Vol. XV of *Santa Fe Institute Studies in the Sciences of Complexity*, Addison-Wesley, pp. 175–193.

Sauer, T., Yorke, J. A. and Casdagli, M. (1991). Embedology, *Journal of Statistical Physics* **65**: 579–616.

Schaal, S. and Atkeson, C. G. (1994). Robot juggling: Implementation of memory-based learning, *IEEE Control Systems* **14**: 57–71.

Scherpen, J. M. A. (1996). $\mathcal{H}_\infty$ balancing for nonlinear systems, *International Journal of Robust and Nonlinear Control* **6**: 645–668.

Shamma, J. and Athans, M. (1990). Analysis of gain scheduled control for nonlinear plants, *IEEE Trans. Auto. Cont.* pp. 898–907.

Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P., Hjalmarsson, H. and Juditsky, A. (1995). Nonlinear black-box modeling in system identification: A unified overview, *Automatica* **31**(12): 1691–1724.

Skogestad, S. and Morari, M. (1987). The dominant time constant for distillation columns, *Comp. and Chem. Eng.* **11**(6): 607–617.

Skogestad, S. and Morari, M. (1988). Understanding the dynamic behavior of distillation columns, *Ind. Eng. Chem. Res.* **27**: 1848–1862.

Söderström, T. and Stoica, P. (1989). *System Identification*, Prentice Hall International Series in Systems and Control Engineering, Prentice Hall, New York.

Sontag, E. (1981). Nonlinear regulation: The piecewise linear approach, *IEEE Trans. Aut. Control* **26**: 346–358.

Strassel, K. A. (1997). Companies find gold in mining their data, Wall Street Journal Interactive Edition (http://www.wsj.com). September 8.

Su, H. T. and McAvoy, T. J. (1997). Artificial neural networks for nonlinear process identification and control, *in* M. A. Henson and D. E. Seborg (eds), *Nonlinear Process Control*, Prentice Hall Inc., Upper Saddle River, NJ, pp. 371–428.

Takens, F. (1981). Detecting strange attractors in turbulence, *in* D. A. Rand and L. S. Young (eds), *Dynamical Systems and Turbulence, Warwick 1980*, number 898 in *Lecture Notes in Mathematics*, Springer-Verlag, pp. 366–381.

Tan, Y. and Cauwenberghe, A. V. (1996). Nonlinear one-step-ahead control using neural networks: Control strategy and stability design, *Automatica* **32**: 1701–1706.

Trainham, J. A. (1996). The environment, process design, and control: Better things for better living, *13th IFAC World Congress*, Vol. Plenary and Index, pp. 83–85.

Wahl, E. F. and Harriot, P. (1970). Understanding and prediction of the dynamic behavior of distillation columns, *Ind. Eng. Chem. Process Des. Dev.* **9**: 396–407.

Weigend, A. S. and Gershenfeld, N. A. (eds) (1994). *Time Series Prediction: Forecasting the Future and Understanding the Past*, Santa Fe Institute Studies in Complexity, Addison-Wesley Publishing Company.

Whitney, H. (1936). Differentiable manifolds, *Ann. Math.* **37**: 645–680.

Zhou, K., Doyle, J. C. and Glover, K. (1996). *Robust and Optimal Control*, Prentice Hall Inc., New Jersey.

Ziegler, J. G. and Nichols, N. B. (1942). Optimum settings for automatic controllers, *Transactions of the A.S.M.E.* **64**: 759–768.