Combining Prior Knowledge and Nonparametric Models of Chemical Processes

by
Michael Lewis Thompson

S.M.Ch.E., Massachusetts Institute of Technology
June, 1984

B.S.Ch.E., Northwestern University
June, 1982

Submitted to the Department of Chemical Engineering
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1996

Author

Department of Chemical Engineering
March 19, 1996

Certified by

Mark A. Kramer
Thesis Supervisor

Accepted by

Robert E. Cohen
Chairman, Committee for Graduate Students

Combining Prior Knowledge and Nonparametric Models of Chemical Processes

by

Michael Lewis Thompson

Submitted to the Department of Chemical Engineering
on March 19, 1996,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Keywords: Systems engineering, process modeling, probability theory, probabilistic inference, Bayesian estimation, Bayesian statistics, artificial intelligence, neural networks, semiparametric modeling, nonparametric modeling, chemical industry.

Abstract

Modeling is an important task in all fields that benefit from quantitative descriptions of the systems being studied. These fields include engineering, the sciences, applied mathematics and econometrics; all of which require accurate models despite the complexity and uncertainty inherent in the problems they address. In particular, engineers and scientists in the chemical process and biotechnology industries must often resolve problems complicated by the underlying biological, chemical, and physical phenomena. Additionally, the process data are often sparse and corrupted by noise, sensor faults and missing values. This inherent uncertainty prevents a modeler from synthesizing accurate models solely from first principles. Consequently, development is often limited to only a partial specification of the underlying phenomena. The resulting models may not be robust enough to satisfy the demands of many systems engineering applications such as process control, design and optimization.

This thesis presents a probabilistic approach for synthesizing robust models despite the uncertainty present in complex systems typical of the chemical, biotechnology, and petroleum industries. By applying probability theory, the methodology unifies advances in artificial intelligence, e.g., radial basis function neural networks, with statistical methods. This allows a modeler to combine prior knowledge of first principles with flexible models derived from data (i.e., nonparametric models). Importantly, the thesis also presents the implementation of the methodology as a computationally efficient suite of computer routines that performed well in predicting process behavior despite the existence of unmeasured variables and corrupted data.

At the heart of this approach is the uniform representation of knowledge as probability density functions. Probability density functions (pdfs) are capable of capturing the full range of knowledge about a quantity from absolute certainty to complete ignorance. This

thesis identifies the appropriate pdfs for the domain of chemical process modeling and formulates the general state estimation problem in a decision theoretic framework. This general framework expresses the systems engineering tasks of state prediction, data rectification, and fault diagnosis as siblings descended from the same state estimation family tree.

Despite the concise problem formulation, finding the optimal estimate of the state remains a difficult task because of the mathematical complexity of the pdfs involved. However, this thesis proposes approximating pdfs by finite mixtures of normal distributions and using techniques such as expectation-maximization (EM) to solve the resulting Bayesian estimation problem. Additionally, it is shown that the framework justifies an empirically motivated approach that synthesizes hybrid models combining radial basis function neural networks and first principles. This hybrid approach is shown to be an approximation to a special case derived from the probabilistic framework. This helps explain the hybrid approach's good performance in practical applications.

The results have been promising. The methodology has been successfully applied in modeling fed-batch penicillin fermentation and vinyl acetate polymerization. The synthesized models have been more accurate and more robust, hence reliable, than models synthesized from either first principles alone or completely nonparametric (fully data-derived) models. In light of these positive results, broader usage of this probabilistic paradigm for systems engineering is advocated.

Thesis Supervisor: Prof. Mark A. Kramer

Acknowledgments

The lessons I have learned during the course of this research have stretched far beyond chemical engineering and academe. In the most positive and wonderful ways, they have forever shaped my life. As considerate advice and, most importantly, by active example, these lessons were taught to me by the many people with whom I have shared my years at MIT. I am eternally grateful to the men, women and families who chose to teach me. Many of whom are listed below. All of whom have given greatly.

My thesis advisor:	Prof. Mark Kramer (*service above and beyond the call of duty*).
... "academic" advisor:	Prof. Charles Cooney (*guidance, support, and pats on my back since 1982*).
... thesis committee:	Profs. George Stephanopoulos and Michael I. Jordan (*invaluable input always*).
... research colleagues:	Dr. Carlos Rojas-Guzmán, Lloyd Johnston and Dr. Jon Tan (*continuous stimulation and fun! 10.001 TA, anyone?*); Dr. Glen Ko (*insightful expertise and judgment*); Dr. Tor Arne Johansen (*Internet-borne ideas from Norway*).
... roommates:	Dr. John Wlassich (*admirable through the ages*); Carlos and Javier Rojas-Guzmán (*roommates from heaven*); and Sung Joon Yoon (*through thick (smoke) and thin (soot) -- "I love you, man!"*).
... Kyoto/MIT friends:	Yu and Mark Hasegawa-Johnson (*much loved "Solutions" to "the Problem"*); Kush, Yoshiko, Ouma, Iliana, Boo Ho, Ross, and the ten students from Kyoto with whom I shared the summer of 1993 (*Watashi no tomadachi desu ne!*).
... first officemates:	Cathryn Shaw-Reid (*perfection*), Dr. Eddie Koury (*companionship/competition*), Arthur Lue (*wonderment*), Drs. Mark and Erin Johnson (*happiness*).
... lunchmates:	Cathryn Shaw-Reid, Prof. Paula Hammond, Gerald Prioleau, Dr. James Oliver (*1991 "Talented Tenth" at MIT Chem. E.*).
... El Paso families:	Sergio and Marci Alvarado (*Best folks I'll ever know! "R-r-right!"*); Mrs. Ines Alvarado and family (*generations of love and respect*); Charlotte Vines (*beloved sister*), Dr. Colin, Gareth and Charles Bill (*treasured cousin and little nephews*).
... SPES family:	Director John Riccobono (*paragon of Christian character*); Tom Haferd and counselors Michael Julian and Aaron Mbowa (*men dedicated to our future*); Francisco Errold, Reggie Charles, Francois Exilhomme, Marcus Richards (*valuable teachers all*) and especially, Hadley Camilus (*he'd be the one, if I had a son*).
... Boston families:	Michael and Djene Morris (*God's examples for my life*). Reggie Remy (*potent promise*) and the Camilus and Bobos families (*homes of honor and love*).
... friends afar:	Christine, Jodye, Atsuko, Sue and Diane (*teachers of wisdom, faith, patience and passion; humor, strength, love and compassion*).
... Amoco/Dow friends:	Dr. Sanjeev & Kavita Katti, Lawrence & Vivian Zhang, Dr. Joe Harrington, Dr. Wolf Koch, Dr. Dave Sabo, Dr. Dan Sajkowski, Dr. Craig Vaughn & Dr. Duane Lehman and their families (*friendship, support and priceless memories*).
... Chem. E.s of '96:	Celia (*"down to Earth," yet so admirably tall*); Cindy, Ethel, Yvette (*Methyl, Ethyl, Propyl; my "Alkyl groupies"*); Anita Z. & Anthony N. (*excellence incarnate*).
... Chem. E. staff:	Elaine Aufiero-Peters, Carol Phillips, Linda Mousseau (*caring when many weren't*).
... financial support:	the NSF and Amoco Vice President, Dr. Al Kozinski (*assistance, with vision*).

Most of all, my family: Willie and Viola (*the heroes and nurturers of my life*); Bernard, Roslyn, Ernie, Cheryl, Keitron and Freddy (*best friends: older, younger, dearest forever*); Candice, Cody and Michael (*darling future engineers!?*).

Biographical Information

Born May 24, 1960 in El Paso, Texas.

Post-Doc, '96-97	MITSUBISHI CHEMICAL CORPORATION	Mizushima, Japan
EDUCATION	MASSACHUSETTS INSTITUTE OF TECHNOLOGY	Cambridge, MA
1990 to 1996	Ph.D. Chemical Engineering, March 19, 1996. Minor: Statistics & Artificial Intelligence.	
1982 to 1984	MASSACHUSETTS INSTITUTE OF TECHNOLOGY	Cambridge, MA
	S.M. Chemical Engineering. Advisor: Charles Cooney. Thesis: System Analysis, Simulation, Control and Optimization of the Fed-Batch Penicillin Fermentation. Football team.	
1978 to 1982	NORTHWESTERN UNIVERSITY	Evanston, IL
	BS in Chemical Engineering Varsity track: school record-holder in discus.	
CONSULTING	ASPEN TECHNOLOGY INC.	Cambridge, MA
1993 to 1995	Expertise in neural networks and statistics.	
INDUSTRY	AMOCO OIL COMPANY	Naperville, IL
1987 to 1990	Research Engineer. AI/Knowledge-based systems.	
1984 to 1987	DOW CHEMICAL COMPANY	Freeport, TX
	Research Engineer. Epoxy resins polymerization modeling.	
Summers	DOW CHEMICAL COMPANY Freeport, TX and Midland, MI; Modeling, simulation, control.	
1979 to 1983	ALUMINUM COMPANY OF AMERICA (ALCOA) Pittsburgh, PA; Modeling and simulation.	
RECRUITING	AMOCO OIL COMPANY	Cambridge, MA
1990 to 1995	Campus Liaison for Recruiting at MIT	
TEACHING	MASSACHUSETTS INSTITUTE OF TECHNOLOGY	Cambridge, MA
1993 to 1995	Teaching Assistant for undergraduate courses (6 courses total).	
Summer	KYOTO SCHOOL OF COMPUTER SCIENCE (Summer Program)	Boston, MA
1993	Taught "American Technology" to undergraduates from Kyoto, Japan.	

PUBLICATIONS

Journals: Thompson, M. L. and Kramer, M. A., "Modeling Chemical Processes Using Prior Knowledge and Neural Networks," *AIChE J*, 40(8) 1328, 1994.

Proceedings: Thompson, M. L. and Kramer, M. A., "A Hybrid Modeling Methodology to Combine Prior Knowledge and Neural Networks," *Proc. IJCNN '93-Nagoya*, 1993.

Kramer, M. A., Thompson, M. L., and Bhagat, P. M., "Embedding Theoretical Models in Neural Networks," *Proc. Amer. Control Conf.* 475, 1992.

In press: Thompson, M. L. and Kramer, M. A., "Efficient Cross-Validation Using Sensitivity Analysis for Models Nonlinear in the Parameters," for *J. Comp. and Graph. Statistics*

Thompson, M. L. and Cooney, C. L., "Optimal Substrate Addition for Secondary Metabolite Production in Fed-Batch Fermentations," for *Biotech. Bioeng.*

Johnston, L.P.M., Thompson, M. L. and Kramer, M. A., "An Efficient Algorithm for Data Rectification," for *AIChE J.*

HONORS National Science Foundation Creativity Award ('90); Graduate Engineering for Minorities (GEM) Fellow ('82); Mortar Board Senior Honor Society ('81); National Merit/Achievement Scholar ('78). Gold Medals in Discus: Houston Corporate Track Meet ('85) and Mass. Bay State Games ('92).

COMMUNITY SPES Counselor/tutor to boys in inner-city Boston, 1991-95, year-round (6 hr/wk).

LANGUAGES Japanese (<1yr). Computer: Matlab, Maple, C, FORTRAN, Pascal, Smalltalk, Lisp, Prolog.

Table of Contents

List of Figures

List of Figures (continued)

List of Tables

Chapter 1

Introduction

Model synthesis is the topic of this thesis. The primary focus is modeling applications in bioprocess and chemical engineering for purposes such as process design, control and optimization. However, this research is applicable as well to model synthesis in other engineering and scientific disciplines, applied mathematics, econometrics, and any other area that requires quantitative modeling.

Several factors complicate model synthesis:

- Uncertainty – processes contain phenomena that are manifested by unmeasured variables or ambiguous relations between the process variables; e.g. reaction rate constants, constitutive relations, and physical properties.
- Complexity - biological and chemical processes exhibit multiple phenomena that often require complicated models involving many variables and equations.
- Competing models - often multiple models with differing qualitative behaviors over different operating regimes or under different assumptions must be selected or combined.
- Sparse data - process data are usually obtained from a relatively narrow operating regime and, thus, only sparsely cover the potential variable space, especially in high-dimensional problems.

- Corrupted data - process data are corrupted by process and sensor noise, and possibly also by sensor malfunctions and missing values.

These problems make model synthesis a difficult task. Uncertainty and complexity make it impossible to fully specify the true structure of the underlying process relations. Sparse and corrupted data yield models that are inaccurate or that extrapolate poorly. And, choosing the single model from among several candidates or finding a means to combine several models is difficult if neither candidate is accurate over the intended domain which the full model is to be applied.

1.1 Technological Vision

This thesis applies probability theory to the problem of combining prior knowledge with nonparametric (data-derived) models in systems engineering. In its grandest sense, the technological vision of the thesis is the advancement of a probabilistic paradigm for systems engineering. A return to the foundations of scientific inference yields insights into formulating and subsequently solving the difficult problems associated with the systems engineering tasks of process modeling, data rectification, and fault diagnosis. These tasks require reasoning under uncertainty in the presence of empirical data. As such, they pose typical problems in scientific inference, which are readily cast in a decision theoretic framework and solved using the probability calculus.

The modeler's knowledge, the available empirical process data, and any case-specific evidence constitute the basis for all inferences and decisions made concerning the process. Traditionally, approaches to apply this information in model synthesis have leaned heavily towards either end of the knowledge-based versus data-derived spectrum. This is depicted in Figure 1.1, which illustrates the prevailing modeling approaches for each regime.

At the knowledge-based end are "parametric" modeling approaches that exploit prior knowledge of the process and first principles to specify fixed form mathematical models. A parametric model has a fixed structure yet is characterized by parameters that are usually estimated with the help of data. At the extreme of the knowledge spectrum, the true model structure and even the true parameter values are known, completely eliminating the need for data.

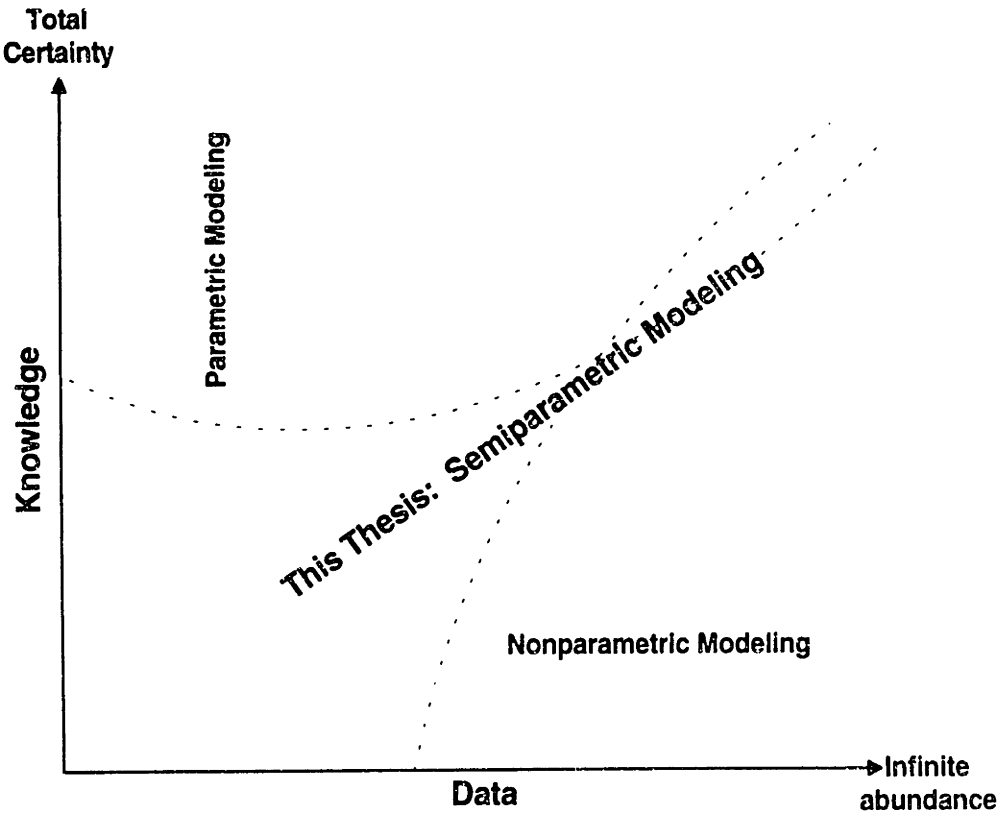


Figure 1.1: Parametric, semiparametric and nonparametric modeling regimes

At the data-derived end are "nonparametric" modeling approaches that exploit an abundance of data to induce flexible form mathematical models whose parameter set is defined only after presentation of the data. In the unrealizable limiting case, all possible process states are completely and exactly measured and prediction (or state estimation) becomes a simple lookup from the tabulated measurements. Before this extreme, though, the reliance upon the data becomes strong enough that only the structural family of models

need be specified, and the model form is determined during parameter estimation by selection of a specific structure from the family. This is called a nonparametric modeling approach, in that a finitely parameterized model structure is not specified *a priori*.

It is the nature of the real world that many problems fall well between these limiting cases: the knowledge is not complete enough to fully specify a parametric model, and the data are either not of high enough quality, broad enough spread or abundant enough to yield a reliable nonparametric model. Currently, approaches in this realm are called semiparametric because they strive to combine some *a priori* fixed model structure with flexible components that rely upon the data to dictate their final form.

This thesis addresses the gulf between the parametric and nonparametric approaches which remains to be satisfactorily filled with pragmatic solutions. Aside from the direct practical reason of synthesizing more accurate process models, this research is necessary because there is a need to unify the many splintered approaches to modeling systems from which data have been collected. Though all such approaches have a common objective and hence a common motivational pedigree, they differ in their treatment of the knowledge a modeler possesses and the data that have been gathered from the process. Taken together, the process knowledge and the data constitute the available information. The splintering of approaches can be attributed to the different emphases put upon this available information in the scientific fields from which these approaches have sprung.

For example, statistical modeling approaches, which explicitly address the stochastic nature inherent in systems, tend to be more data analytic than the functional analytic approaches born from applied mathematics, which concentrate more on the deterministic aspects of the process. Take the case in which little is known about a system in terms of model structure but a wealth of data exists. In such a case, while the (classical) statistician tends to use nonparametric density estimation techniques to generate data-derived models and uses probability density functions to describe the residual information uncompensated for by the model, the applied mathematician tends to use series approximations of the

underlying "true" function and describes the residual as an approximation error due to series truncation.

These different approaches are usually viewed as diametrically opposed modeling philosophies. However, greater insight is afforded if they are viewed as belonging to a continuous spectrum measuring the strength of the bias injected into the inductive process. The statistician, when adopting a nonparametric approach, injects minimal bias about model structure, allowing the data to dictate the model's form, while the applied mathematician injects considerable bias by presuming a model of a particular functional class approximable by a series summation. In other words, what really differs between the two approaches is the extent to which each admits certainty/ignorance of the assumed model form and hence, the degree to which each source of information – prior knowledge and data – is weighted. Yet, in practical application, both approaches view the problem from a function approximation (or curve-fitting) standpoint. I denote this paradigm as a function-oriented paradigm.

By "function-oriented paradigm" we mean a modeling paradigm that focuses on the model structure first and foremost with an intent to approximate a supposed true function. In doing so, these approaches offer an assortment of techniques to modify the model structure and to augment the parameter estimation objective function according to the available information. The problem with the function-oriented paradigm is that there does not exist a well-formed theory for uniformly incorporating the various types of prior knowledge into the model synthesis process and for accounting for common shortcomings in the empirical data, like outliers and missing values. Consequently, researchers in the many fields that have addressed this problem have invented a wide variety of often *ad hoc* methods. In sum, the uniqueness and optimality of function-oriented paradigm approaches are often questionable.

An alternative is the probabilistic modeling paradigm, which puts the available information at the center of attention and focuses upon representing the degree of uncertainty

associated with it. In the probabilistic paradigm, making plausible inferences about the process state in light of the uncertainty in the available information is central. The functional structure of the model, any assumptions or preferences imposed by the modeler, and the available empirical data all have complementary and well-defined roles. Probability density functions represent the inherent uncertainty in each. And, inferences or estimates about the process are made using the probability calculus, which has been proven to be the only coherent paradigm (i.e., only paradigm that is rational and consistent with the modeler's values and beliefs) for scientific reasoning under uncertainty (Howson and Urbach, 1983; Bernardo and Smith, 1994; Berger, 1985).

But, what does the modeling paradigm matter if both approaches yield acceptably accurate models? Aside from the issue of uniformity, the paradigm matters most when the modeler is confronted with an atypical situation or a situation that pushes an approach to its breaking point. Such a situation arises when attempting to use all of the knowledge and all of the data to maximum benefit in synthesizing a model when neither alone is sufficient to produce an accurate model. This is the problem confronted in semiparametric modeling and dealt with here.

This thesis examines semiparametric modeling in both the function-oriented and probabilistic paradigms. First, it derives and demonstrates an intuitive and empirically proven function-oriented approach. The approach combines artificial neural networks with first-principles models and offers substantial improvements in prediction accuracy and robustness over purely parametric and purely nonparametric methods. This function-oriented approach synthesizes hybrid neural networks based on a rational assignment of prior knowledge to the model structure and the parameter estimation problem. However, the function-oriented approach has shortcomings that limit its applicability, namely the lack of an explicit theoretical basis for combining models, treating data from multiple sources, and managing data with outliers or missing values.

To rectify this, our research turned to the derivation of a more general methodology grounded in probability theory. This research shows that difficult issues faced in process modeling can be handled readily by adopting the probabilistic paradigm, which avoids the case-specific extensions common in the function-oriented paradigm. In this way a single, uniformly consistent methodology for synthesizing models in an extremely wide variety of cases is provided. Moreover, pragmatic implementation and solution methods are investigated.

1.2 Plan of the Document

This document is laid out in the following fashion. This chapter concludes with the motivation of the research. Several examples are presented that capture the essence of the problems this research addresses. Also, the objectives and significance of the research are stated. Chapter 2 then gives the background context of the thesis. Prior work in a wide variety of fields is briefly described. This serves to define the state of the art in semiparametric modeling and to help create an appreciation for the breadth of this research. It also sets the stage for the development of the two approaches investigated herein: (1) the empirically motivated function-oriented paradigm of Chapter 3 and (2) the theoretically based probabilistic paradigm that comprises the remaining chapters of the document.

Chapter 3 presents a function-oriented approach for semiparametric modeling. The chapter describes a good pragmatic method to combine prior knowledge with nonparametric models (specifically radial basis function neural networks). The intuition, derivation, and implementation of the approach are documented there. The results of applying this approach to a vinyl acetate polymerization process and a fed-batch penicillin fermentation are also presented. Also a novel method for estimating prediction error of nonlinear models is presented and evaluated. Although a good starting point that proved better than strictly parametric and strictly nonparametric approaches, the semiparametric modeling approach of Chapter 3 still required a theoretical justification and a basis to

address tougher problems such as corrupted data. These were provided by the research presented in Chapters 4 and 5.

Chapter 4 obeys the mandates of completeness and generality to present a formal theoretical basis for combining prior knowledge and data in model synthesis. In this chapter, the probabilistic paradigm of process modeling is presented by formulating the modeling problem in a decision theoretic framework and applying the axioms of probability theory. The result of this derivation is a Bayesian estimation problem that is general. Also, the subproblems of model synthesis, parameter estimation and hypothesis selection are discussed. Given the probabilistic problem formulation, Chapter 5 illustrates the representation of process knowledge as probability density functions, which are the basic building blocks of the paradigm. The definition and importance of finite mixture distributions in model synthesis are also explained there. Notably, the specific assumptions and approximations that must be applied to this problem to arrive at the function-oriented method of Chapter 3 are presented in Chapter 5. Thus, a theoretical basis is provided for the empirically motivated approach of Chapter 3.

Chapter 6 follows with a description of solution methods for the Bayesian estimation problem. Included are the specific efficiency enhancements that we derive for process modeling by exploiting finite mixture distributions and applying the expectation-maximization (EM) algorithm. Chapter 6 also presents a robust state estimation solution that is capable of determining model parameters and predictions despite the increased uncertainty due to data corrupted by sensor noise, gross errors, and missing values. The chapter concludes by describing the implementation in computer software of these solution methods. The implementation is a set of Matlab functions we call the "Probabilistic Modeling Toolkit for Matlab."

Chapter 7 presents the results of applying the probabilistic methodology and is particularly important from an applied standpoint. In the chapters prior to it, the basic problem formulation was stated and methods to solve the problem were described and

implemented. What remained was a clear demonstration of how to apply the modeling methodology derived in this research to the real problems a modeler faces. This is accomplished in Chapter 7. It demonstrates the modeling methodology in a step-by-step manner and applies the methodology in three case studies. Additionally, Chapter 7 presents the diagnostic measures available as a result of this framework. It discusses how the increased interpretability afforded by these diagnostics facilitate the evolutionary process of model synthesis.

The case studies presented in Chapter 7 are the state estimation of corrupted data from an industrial fermentation process and the probabilistic estimation of the fed-batch penicillin fermentation originally solved in Chapter 3. As the methodology is applied in each case, the reader is provided with heuristics for transforming the given prior process knowledge into appropriate probability density functions. Also, the solution of common problems presented by data such as missing values and corruption by sensor failures are provided. At each step of the methodology, references are made to the specific applicable functions of our "Probabilistic Modeling Toolkit for Matlab." Chapter 7 presents and discusses the results of the case studies. The practical utility of the diagnostic measures and plots are demonstrated. Moreover, the prediction performance in light of data corrupted by missing values and gross errors are presented.

Chapter 8 signals the end of the main body of this thesis. It summarizes the contributions of the research and mentions possible future research. Finally, the document concludes with references and the appendices, which contain a summary of the mathematic notation, some useful mathematical formulas for manipulating normal distributions, and the computer software source code.

1.3 Motivating Issues

To provide more concrete motivation for this research, examples of several problems faced by process modelers are listed below. Each represents important issues confronted

by a modeler. To resolve each, it is imperative that all available knowledge and data be applied optimally.

1.3.1 Process Constraints

Often it is impossible to prescribe accurate parametric models that obey known process constraints. Yet, sufficient data may exist to derive nonparametric models, which the modeler must ensure obey the process constraints globally over all input conditions. In general, the inclusion of prior knowledge in the form of inequality constraints is difficult to achieve with nonparametric models. Merely ensuring that during parameter estimation the model obeys the constraints at the data points is not sufficient to guarantee that the model will obey the constraint at all points in input space. A model synthesis methodology that is capable of guaranteeing global satisfaction of the constraints would be of great value in producing models for control and optimization purposes. These engineering applications often require models that predict accurately in operating regimes not covered by the data used in parameter estimation.

1.3.2 Unmeasured Phenomena

In a typical process, many variables are unmeasured and are inferred from the available measurements. Also, explicit functional relations may not exist to describe the dependence of all state variables. These shortcomings in the process knowledge prevent a modeler from applying a solely parametric model. In such cases, the modeler must revert to the data to induce a suitable nonparametric estimator of the unmeasured quantities.

For example, in the information flow diagram of the fed-batch penicillin fermentation of Figure 1.2, the current biomass and substrate concentrations, $X(t)$ and $S(t)$, respectively, affect the future biomass concentration $X(t+\Delta t)$ indirectly through the specific growth rate $\mu(X,S)$. Unfortunately, μ cannot be readily measured and must be estimated. Though parametric correlations have been posed to relate μ to X and S, in general they are plagued by considerable uncertainty. Yet, without presupposing μ, the modeler cannot take

advantage of the beneficial inductive bias offered by the biomass balance around the fermentor. The balance is a parametric model that has the potential to improve the predictive accuracy and extrapolative capability of the model. In other words, by imposing the structure implied by the existence of the intermediate unmeasured μ, a more accurate model can be built despite the uncertainty in μ; but the dependence of μ on X and S is best represented nonparametrically to account for the uncertainty in model form. A model synthesis methodology must be robust enough to infer reasonable estimates for such unmeasured intermediates so that they can be exploited by the inclusion of other dependent forms of prior knowledge.

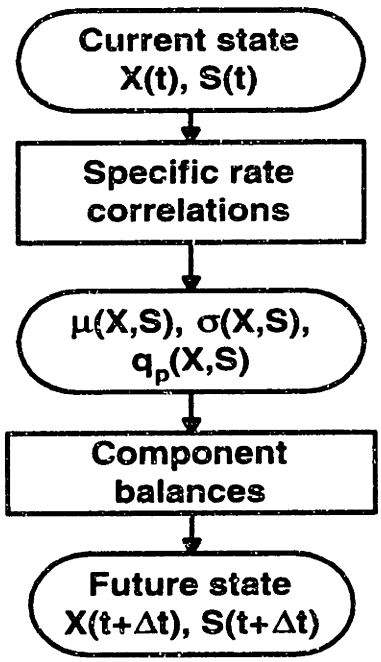


Figure 1.2: Fed-batch penicillin fermentation information flow diagram

1.3.3 Multiple Models

Often a single model – without undue complexity – cannot accurately describe the full range of process operating conditions. For example, in the pH neutralization process depicted in Figure 1.3, the outlet pH response to changes in base flow rate Q_3 changes

with respect to the current values of pH and Q_3. Therefore, multiple operating regimes with different process behavior must be accounted form. To resolve this issue, a model synthesis methodology must be able to combine multiple models into a single robust model that predicts accurately and smoothly even during transitions from one regime to another.

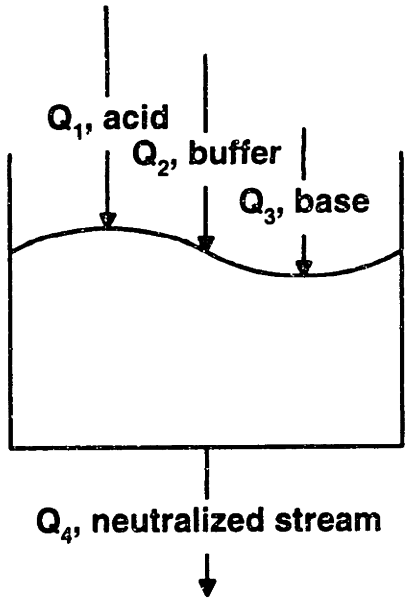


Figure 1.3: pH neutralization process flowsheet.

1.3.4 Extrapolation

In the process industries, there is a continuous push for optimal process operations. To achieve this, new control systems must be designed and operating conditions discovered that have not been previously investigated. This is particularly true of the process control, design and optimization tasks of systems engineering. The starting point for new advancements is usually the store of knowledge in the form of models that exist for the

process. However, exploration of new operating regimes requires that extrapolations be made by these models.

For example, in the pH neutralization process shown in Figure 1.3, data may have been gathered from a limited process region characterized by intermediate pH values. To design a robust control system for the process, an engineer requires that the model perform accurately over the full range of pH values. Thus, in the absence of data in the high and low pH regimes, a model synthesis methodology must still produce an accurate model.

1.3.5 Process Faults

Along with sensor failures, all manner of process disturbances, equipment malfunctions, and faults may plague a process. These may or may not be manifested in the data, yet the ability to accurately predict process performance during these atypical periods is still essential.

For example, in the pH neutralization process, if a leak in stream S_3 were to arise, the process model should be robust enough to still yield accurate estimates for the outlet pH. To achieve this, a model synthesis methodology must incorporate prior knowledge about candidate faults or at least about potential gross deviations in process measurements.

1.3.6 Partitioned Data Sets

In some modeling problems, the process data available may have arisen from multiple independent sources, and therefore can be partitioned into independent sets. An example of this is in the dynamic modeling case when data have been collected during dynamic process operation and at steady-state operation. It is not always obvious how to combine this information in order to get the best process model.

For example, in the pH neutralization process, we may have only limited data gathered during dynamic operation at moderate pH values. Therefore, as discussed above in the instance of extrapolation, it may be difficult to achieve accurate state estimation in the low and high pH operating regimes. One way of infusing additional information is to supply the titration curve information, i.e. steady-state operating data, gathered across the full range of pH. Though these data are only a few points, they have the potential to greatly influence parameter estimation through the certainty we ascribe to them.

1.3.7 Small Data Sets

As shown in Figure 1.1, the less prior process knowledge a modeler has the more reliant upon the process data the modeler becomes. Even if the data are sufficient to estimate the parameters of a parametric model or abundant enough to synthesize a reasonable nonparametric model, there still may not be sufficient data to validate the model or assess the suitability of one model versus another. This is because the standard model validation methods require estimates of prediction error, which are notoriously variable and are provably accurate only asymptotically with respect to the data, i.e., only as the data abundance approaches infinity. A robust modeling methodology must be able to synthesize models and assess their validity or at least choose the best model from a set of candidates even when data sets are small.

1.3.8 Corrupted Data

As testament to Murphy's law, process measurement systems sometimes fail. Sensors may fail in a variety of ways, and each failure mode gives rise to a particular pattern of corrupted data. Therefore, the detection of and compensation for sensor faults becomes important. Also, data may be corrupted by missing values or outliers due to transmission errors, human error in transcribing results, or unavailability of records with all relevant conditions that affected the process operations. A model synthesis methodology should

use all available knowledge and measurements of other variables to produce models robust in the face of such shortcomings.

1.4 Outstanding Issues

Despite the large variety of approaches to process and data modeling, there still remains an incomplete prescription for solving modeling problems that are characterized by both lack of knowledge and lack of data, such as those listed in the previous section. It is clear that when knowledge of the process is fairly complete, the reliance on the data is minimized and the model structure can be fully specified. Conversely, as the abundance of data (or information content contained in the data) exceeds that of the prior process knowledge, modeling approaches become more data analytic and statistical in nature. However, a sound semiparametric modeling methodology that is capable of synthesizing models robust enough for chemical process engineering when both the knowledge and data are deficient in some way is still an unfulfilled necessity.

The function-oriented paradigm focuses on the equality and inequality relations specified by the modeler and attempts to determine how to best combine them. However, due to the infinite number of possible methods, criteria are needed to zero in on those methods that are most promising. In most cases it is pragmatism that achieves this: select the method that works well in practice. Empiricism motivates these approaches. This thesis research attempts to find more satisfactory criteria to refine these methods.

On the other hand, the probabilistic paradigm addresses the more general inferential search for a scientific means of conducting plausible reasoning using information drawn from multiple sources. It defines guidelines for model synthesis in order to maintain consistency and completeness. In other words, a methodology is valid if it makes sense within the prescribed axiom system of probability theory. However, at the levels of eliciting prior probability density functions and of implementing practical computational methods, there must be a retreat from pure theory. This thesis research presents a methodology within

this theory and then investigates reasonable means of implementing it efficiently and practically.

1.5 Research Objectives

The objective of this project is to develop a methodology for synthesizing robust process models. Robustness is characterized by the accuracy, consistency, and generalization capability of the models despite shortcomings in the available process knowledge and data. Models generated by the methodology should be more accurate than conventional models synthesized from data sets of the same size. The new models should be consistent with the physical constraints of the real process. And, the models should yield reasonable predictions in a larger region of variable space or more general context of process operation. Also, the methodology should be formalized with sound theoretical underpinnings. The remainder of this section lists the specific research objectives.

1.5.1 Define Formal Problem in Plausible Reasoning

The first step in addressing the issues described above is to formally state the process modeling problem. It is clear that model synthesis requires inferences drawn from and decisions based on uncertain information in the form of process knowledge and data. As such, the problem requires a formal statement as the starting point for a methodology with a well-founded theoretical basis.

> *First, this project will formally define chemical process modeling as a problem in plausible reasoning.*

1.5.2 Characterize Prior Knowledge in Process Modeling

Within a rational framework for plausible reasoning, we seek a modeling methodology that produces robust process models. This methodology must categorize the types of knowledge commonly found in chemical process modeling and provide a basis for

integrating each type with information provided by empirical data. The appropriate types of prior process knowledge must be identified.

Second, this project will characterize the sources and types of prior knowledge in chemical process modeling.

1.5.3 Represent Prior Knowledge

After characterizing each type of prior knowledge, we will develop a means of exploiting the knowledge. To achieve this, we must determine how best to represent and integrate the varied types of prior knowledge. In model synthesis, the uncertainty present in prior knowledge and data must be accounted and compensated for. Parametric models specified by prior knowledge must be combined with nonparametric models derived from data. A methodology that represents both types of models as an integrated semiparametric model is sought.

Third, this project will develop a methodology that incorporates prior process knowledge and data-derived models into a semiparametric model.

1.5.4 Quantify Model Reliability

Prior knowledge may also influence the objective function, constraints and algorithm of the parameter estimation method. We seek a modeling framework that can accommodate prior knowledge in each of these estimation elements. Furthermore, the existence of a modeling methodology does not imply that models designed using it will have desirable properties. The methodology must provide metrics that quantify model reliability and then demonstrate that it yields reliable models. A framework that provides a tight link between parameter estimation and model analysis is necessary.

Fourth, the methodology will have the capability to evaluate and compare the effects of different types of prior knowledge in synthesizing models that are provably reliable.

1.5.5 Implement as an Efficient System

The methodology must be practically implementable as a computer system. As such, the system must provide an efficient protocol for interacting with the user. This includes specifying the model synthesis problem and model structure so that they are conducive to efficient parameter estimation. To achieve this, model regression algorithms that exploit prior knowledge in improving the efficiency of parameter estimation must be developed.

> *Fifth, the methodology will be implemented in the form of a computational system that includes efficient algorithms for parameter estimation, prediction and assessment of model performance.*

1.6 Research Significance

This research addresses the difficult systems engineering task of process model synthesis. It attempts to present a formal methodology for solving a broad class of modeling problems in which neither process knowledge nor data alone contain sufficient information to produce accurate process models. Noting that reasoning under uncertainty is the domain of decision theory and probabilistic inference, we adopt a probabilistic paradigm in which to derive a formal methodology. Therefore, this research has several contributions:

- *Formalization of process modeling to explicitly account for the uncertainty inherent in the modeler's knowledge and the available data.* This provides a modeler with a solid framework for model synthesis, assuring correctness and completeness, thus providing more reliable models and subsequently more reliable processes designed and operated based on these models.

- *Characterization and probabilistic representation of the specific types of prior knowledge in the domain of process modeling.* This allows a modeler to take full advantage of applying probability theory when reasoning under uncertainty. The modeler can incorporate more

information into the model synthesis process and tie the model more closely to the decision-making process served by the model in design, control and fault diagnosis applications.

- *Implementation of solution methods for model synthesis, parameter estimation and state estimation for a large class of problems involving multiple operating regimes and corrupted data.* This allows a modeler to solve problems that require combining diverse models of complex process, using data gathered during dynamic and steady-state operation, and rectifying errors and missing values in the measurements of the independent and dependent variables.

Chapter 2

Background and Literature

Prior work relevant to this project spans the fields of chemical engineering, probability theory, statistics, econometrics, computational learning theory, cognitive science, applied mathematics and mathematical programming. In these fields, the role of prior knowledge in improving both the model synthesis process and the generalization capabilities of the resulting models has been identified. The literature describes approaches to achieving these goals and developing computational systems to implement them. This section reviews the most relevant work.

The model synthesis problem has several elements that comprise the knowledge base or modeling hypothesis H. Here they are listed generally, and later, in Chapter 4, they will be framed probabilistically. The elements are the following:

- The *input (x) and output (y) domain definitions*, which specify the dimensionality and variable types of the input-output space: $x \in D_x \subset \Re^n$, $y \in D_y \subset \Re^m$.

- A *data model*, which defines the relationship between the true process variables and the measurements and specifies the noise distributions (e_x, e_y): e.g., $e_x \sim N(0,\eta^2)$; $e_y \sim N(0,\sigma^2)\}$, where "$e_y \sim N(0,\sigma^2)$" means "e_y is distributed according to a normal distribution with mean zero and variance σ^2.

- A *set of constraint relations* ($\mathcal{M}$), which defines the model and its parameterization: e.g., $\mathcal{M} = \{f(x;\theta)=y;\ g(x,y)\geq 0\}$.

- The *performance metric* ($F(\theta)$), which is used to rank the parameter sets of a given candidate model: e.g., $F(\theta;x,y)=\| y - f(x;\theta) \|^2$.

- A *model synthesis (learning) algorithm* ($\mathcal{A}(.)$), which either accepts or determines the model structure and estimates the optimal model parameters θ^* according to the performance metric: e.g., $\theta^* = \mathcal{A}(\mathcal{H},F,D)$.

Prior knowledge defines the modeling hypothesis H. Typical process operating conditions, physical limitations of process equipment, first principles, and the intended application of the model all help in dictating the input-output domain boundaries and constraints upon the candidate models. Often, substantial knowledge about the functional behavior of the relationships between variables will be known within this domain or subregions of the domain. These behaviors include continuity, smoothness, flatness, monotonicity, concavity, and convexity. Also, features like asymptotes, extrema, zeros, and inflections may be known. Behavioral knowledge provides constraints upon the model and its derivatives. These constraints define the candidate model forms in the hypothesis set and provide insight into choosing an appropriate performance metric and learning algorithm.

Prior work has focused upon optimally integrating this prior knowledge:

- Probability theory combines prior knowledge and data as information represented by probability density functions, which capture the uncertainty inherent in each and provide a means for plausible reasoning and decision-making.

- Statistical modeling accepts prior knowledge of the data with respect to its noise behavior and density function; it provides the theoretical basis for posing the model, formulating the objective function, and analyzing model reliability.

- Computational learning theory includes prior knowledge as inductive bias in model structure and algorithms for learning; it provides the theory for quantifying this knowledge in concept learning.

- Neural network research applies prior knowledge in designing model structure (network architecture) and parameter estimation algorithms.

- Mathematical programming provides the algorithmic machinery to efficiently optimize the objective function subject to prior knowledge posed as constraints.

These fields are intertwined at several levels, but may be broadly assigned separate roles in the context of real-valued functional approximation:

- Computational learning theory and cognitive science supply the philosophical basis for justifying the use of prior knowledge, identifying types of prior knowledge, its role in learning new concepts, and how preferences such as simplicity may improve generalization.

- Probability theory defines a rational inference framework, uniformly representing uncertainty in discrete concepts and real-valued variables, and provides models that incorporate prior knowledge and quantify its impact on generalization capabilities.

- Mathematical programming takes the metrics and constraints, derives equivalent unconstrained optimization problems, and supplies provably convergent algorithms to solve them.

Because these fields emphasize different objectives, rarely are they used in an orderly manner in the following steps of model synthesis: specify the problem domain, identify the relevant prior knowledge, represent this knowledge appropriately, regress the most suitable model efficiently, and prove the reliability of the model. Frequently, when grabbing techniques from several fields, modelers move from one step to the next neglecting to make explicit the assumptions associated with each technique. Violating the implicit assumptions of the techniques results in suboptimal or invalid models.

Recently, as neural networks have become ever more popular as nonlinear models, researchers in many scientific and engineering fields have proposed a variety of approaches to exploit prior knowledge in neural network modeling. However, without close adherence to the foundations laid in the fields above and without reassessing the underlying assumptions of the modeling paradigms they have adopted, these researchers often propose *ad hoc* techniques that lack solid rationale. Furthermore, much effort is then lost attempting to iron out the idiosyncrasies of each particular method before finally backtracking to the grounding theories upon which these techniques are based.

The approach proposed here attempts to avoid these pitfalls in the domain of chemical process modeling. In formalizing the characterization and incorporation of prior knowledge in process models, this approach identifies the theoretical pillars that support the methodology and, in attempting to address them, directs attention to the practical and theoretical gaps that remain to be filled.

2.1 Probabilistic Inference and Decision Theory

Model synthesis is a decision problem clouded by uncertainty due to incomplete knowledge and imperfect data. Decision theory represents a formal means of making optimal decisions in the face of such uncertainty. Thus, it provides a natural, and arguably the best, means of posing problems of model synthesis and inferences about models (e.g. Jaynes, 1993; Berger, 1985; Bernardo and Smith, 1994). By exploiting the probability calculus, a decision theoretic framework allows the decisionmaker's prior knowledge to influence inferences drawn from data and assures that these inferences are consistent with the decisionmaker's assumed state of knowledge, beliefs and preferences. At the heart of (probabilistic) decision theory is probability theory, which provides the axioms that dictate how probabilities are combined. Thus, probability theory provides the rules necessary to perform inference under uncertainty, which is called plausible reasoning.

There is a wealth of literature on plausible reasoning using probabilistic inference, more commonly referred to as Bayesian inference. Following Jaynes' (1993), we will use the broader term "probabilistic" throughout this thesis rather than "Bayesian" when referring to the paradigm as a whole, because the probabilistic paradigm encompasses far more than the mere application of Bayes' Theorem. In this thesis, the literature cited on the topic provides (1) the philosophical motivation for probability theory in inference and decision making (Jaynes, 1993; Howson and Urbach, 1988), (2) the axioms and mathematics of the probability calculus (Cox, 1946, 1961; Bernardo and Smith, 1994), and (3) the application of probability theory in modeling (Gatsonis, *et al.*, 1991; Berger, 1985; Box and Tiao, 1974). Each addresses arguments for and against probabilistic reasoning (esp. Howson and Urbach, 1988; and Berger, 1985) and provide extensive bibliographies of probabilistic and non-probabilistic ideas and research (esp. Bernardo and Smith, 1994). Here, we will summarize the motivation for adopting this broader view of probability as the calculus of plausible reasoning, hence also of scientific inference. Our brief treatment is based on Jaynes (1983), who was motivated by Pólya (1954) for providing the qualitative arguments and Cox (1946, 1961) for providing the mathematical derivations.

In making decisions, we must reason despite incomplete information. Especially in the realms of science and engineering, we must account for how plausible events or propositions are if we are to reason most effectively. For example, to determine the cause of a process shutdown, we may not be able to obtain all the information we need to conclude with certainty what caused the shutdown. However, based on the available evidence, we can judge how plausible possible events, such as, say, the failure of a reactor thermocouple, are.

Therefore in reasoning under uncertainty, we desire to (1) quantify the plausibility of a proposition or event, (2) obey common sense in updating our knowledge when assigning plausibilities, and (3) be consistent in arriving at the plausibilities we assign. The first criterion is a practical consideration that helps us to communicate, manipulate, and automate how plausible we think a proposition is. The second criterion appeals to our

intuitive sense of reasoning correctly, forcing us, for example, to admit an increased plausibility in a proposition if the things that we know increase the proposition's plausibility are believed to be true. And, the third criterion keeps us honest by making us use all relevant information available, come to the same conclusion if given the same plausibilities regardless of our path of reasoning, and assign the same plausibilities to propositions that we believe to be equally plausible.

Cox (1946, 1961) proved that a unique axiomatic system of plausible reasoning arises given the above three criteria and correspondence to Aristotelian logic in the limit of certainty. The system dictates the use of probabilities to quantify the plausibilities, and Cox's axioms define the core of probability theory. As a consequence, the broadened view of probabilities as a measure of our beliefs in the plausibility of propositions rather than merely frequencies of events can be exploited in a formal system of reasoning under uncertainty. The axioms that represent the foundations of probability theory are the Product Rule and the Sum Rule. From them the famous Bayes' Theorem and all other valid probability statements can be derived. For discrete propositions A and B and an assumed state of knowledge (or hypothesis) H, the axioms are as follows:

Product Rule: $\mathrm{P}(\mathrm{AB}|H) = \mathrm{P}(\mathrm{A}|\mathrm{B},H)\mathrm{P}(\mathrm{B}|H) = \mathrm{P}(\mathrm{B}|\mathrm{A},H)\mathrm{P}(\mathrm{A}|H)$

Sum Rule: $1 = \mathrm{P}(\mathrm{A}|H) + \mathrm{P}(\text{not A}|H)$,

where "AB" denotes "A AND B", i.e. logical conjunction; and "P(A|H)" denotes "the probability of A given that the state of knowledge H holds", i.e. the probability of A conditioned upon H.

Bayes' Theorem arises by simple manipulation of the Product Rule:

Bayes' Theorem: $$\mathrm{P}(\mathrm{A}|\mathrm{B},H) = \frac{\mathrm{P}(\mathrm{B}|\mathrm{A},H)\,\mathrm{P}(\mathrm{A}|H)}{\mathrm{P}(\mathrm{B}|H)}.$$

Also, the useful operation of marginalization arises by combining the Product and Sum Rules (for mutually exclusive and exhaustive propositions A_i, i=1,...,N):

Marginalization: $$\mathrm{P}(\mathrm{B}|H) = \sum_{i=1}^{N} \mathrm{P}(\mathrm{B}|\mathrm{A}_i,H)\mathrm{P}(\mathrm{A}_i|H).$$

Chapter 4 presents analogous formulas for the Product Rule, Sum Rule, Bayes' Theorem and marginalization when applied to the probability density functions of continuous variables.

The theoretical underpinnings of probability theory are sound, and many complex real-world problems in a variety of fields are being solved by application of probability theory (e.g., Gatsonis, *et al.*, 1991; Rojas-Guzmán, 1995). Aside from philosophical debates – which are thoroughly discussed in the vast literature, e.g. Howson and Urbach (1988) – the slow adoption of the probabilistic paradigm for inference in engineering can be attributed largely to the difficult pragmatic issues of representing our knowledge probabilistically and implementing the calculational tools needed to perform probabilistic inference. Thus, what currently occupies researchers in this field are mainly (1) casting prior knowledge as appropriate probability distributions for specific problems, and (2) deriving efficient computational methods for solving the resulting mathematical problems. Research in both of these areas is discussed in the following two sections. Appropriately, our research described in Chapters 4 through 7 addressed these areas in the context of process modeling.

In almost all of the other fields discussed below, there are researchers who have proposed probabilistic approaches to combining prior knowledge with data-derived models. However, they have usually either restricted the interpretation of probabilities as frequencies or merely tacked on Bayesian estimation methods to function-oriented approaches rather than used the probabilistic paradigm to its fullest. For example, the research in Bayesian learning networks (MacKay, 1992a and Neal, 1995) formulates the model synthesis problem from a function-approximation standpoint, specifying the neural network *a priori*, then attempting to inject prior probability density functions on the network weights to invoke Bayes' Theorem. Such approaches do not take full advantage of the probabilistic paradigm. Probability theory provides a basis for specifying the form of a nonparametric model so that the parameters involved have meaningful prior distributions (see Chapter 5), and it explicitly accounts for shortcomings in the data by

posing probability distributions to capture the stochastic behavior and potential faults of the data model. Moreover, probability theory provides a uniform treatment of the uncertainty inherent in both our knowledge and the data. It maximally exploits the information contained in each to facilitate model synthesis and state estimation. The research in this thesis demonstrates these points.

2.1.1 Representation of Prior Knowledge

Applied generally, probabilities can describe subjective beliefs and preferences as well as more objective sampling frequencies. Though classical frequentist statistics restrict probabilities to frequencies, the broader interpretation of probability as general measures of uncertainty provides great power in scientific inference (Howson and Urbach, 1988; Pearl, 1988). Using subjective probabilities allows a modeler to make belief statements (i.e., pose assumptions) about arbitrary propositions instead of being limited to only those propositions pertaining to the frequency of an event. Thus, probabilities allow a modeler to uniformly and quantitatively represent his or her knowledge about the process being modeled as well as the stochastic nature of the process and the measurement system.

In order to inject objectivity into the representation of knowledge as subjective probabilities, researchers have proposed applying principles such as mathematical invariance (e.g. translational or rotational invariance) to probability assignments. Bernardo and Smith (1994) present some of these derivations, each of which leads to the family of exponential distributions, which we refer to as canonical distributions. One basis for applying these canonical distributions is the Principle of Maximum Entropy. Jaynes (1993) and Skilling and Gull (1988) demonstrate how the Principle of Maximum Entropy leads to objective probability assignments and probability density functions. Simply put, the Principle dictates the unique probability assignment that is least committal yet obeys the constraints imposed by prior information. In this way, only the information available or assumed about the process is incorporated in the probabilities: nothing more, nothing less. Kapur and Kesavan (1993) provide a good overview of Maximum Entropy and

applications of entropic measures. Finally, Templeman and Xingsi (1989) demonstrate the connection between entropic priors and the representation of inequality constraints in the Lagrangian of optimization problems.

Bernardo and Smith (1994) also show that the discrete and continuous canonical distributions are the only distributions that have parameters θ with corresponding sufficient statistics. A statistic τ is any function of the data *D*. The statistic τ is sufficient with respect to θ if $p(\theta|D,\tau)=p(\theta|D)= p(\theta|\tau)$, i.e., the probability of any hypothesis about θ conditioned on both *D* and τ is equal to the probability of that hypothesis conditioned on either *D* or τ alone (Bernardo and Smith, 1994; Howson and Urbach, 1987). This also means that τ summarizes all information about θ contained in *D*. For example, the sample mean *m* of a data set assumed to be from a normal (Gaussian) distribution is a sufficient statistic for the mean of the distribution μ. All inferences based on *D* concerning μ can be made equally as well as and will be identical to those based on *m*.

Appealing to principles of invariance such as Maximum Entropy allow us to transform process knowledge into probability density functions. The gaps remaining to be filled are the characterization of the appropriate knowledge in process modeling, the representation of it as pdfs, and the derivation of tractable solution methods. The research described in this thesis strives to fill these gaps.

2.1.2 Probabilistic State Estimation

In the probabilistic paradigm, the task of model synthesis is motivated by our wish to estimate the state *S* of our process based on the data *D*, any additional evidence *E*, and our prior knowledge *H*. An example is the prediction of $S=y$ (the dependent variables) given $E=x$ (the independent variables at which we wish the prediction). Then, $D=\{X,Y\}$ are matrices of *x* and *y* data gathered from experiments; and *H* is our assumptions about the

dependence of y on x and the procedure used to generate D. (More detailed examples of S, D, E and H and their broader implications are discussed in Chapter 4.)

The interdependence of these quantities is represented using probability density functions (pdf) because they best capture our uncertainty about the process due to incomplete knowledge and stochastic phenomena. The pdf $p(S|E,D,H)$ summarizes this information and uncertainty; and it is well-known that, when optimality is defined as minimum prediction squared-error, the optimal estimate of the state S^* is given by the conditional expectation: $S^* = \mathrm{E}[S|E,D,H] \equiv \int S\, p(S|E,D,H) dS$ (e.g., Berger, 1985).

Often our knowledge describing the relationship between S and E is a function $S=f(E;\theta)$, where the parameters θ are independent of E and summarize the information in the data D. Therefore, as we shall see in later chapters, S^* is expressed in terms of the parameters θ as follows: $S^* = \int f(E;\theta)\, p(\theta|D,H) d\theta$.

Two approaches are then adopted to compute S^*:

Approach 1: Estimate parameters θ and predict S assuming $S=f(E;\theta)$.

Approach 2: Repeatedly sample $p(\theta|D,H)$ to get many values of θ and estimate S as the average of $f(E;\theta)$ evaluated at the sampled θ values.

Approach 1 maximizes $p(\theta|D,H)$ to estimate the parameters, then simply evaluates the model $f(E;\theta)$ given E and the single point estimate of θ. It avoids the computation of the integral by instead putting its energy in the maximization of $p(\theta|D,H)$, which often requires less computation, and using a single θ value in predicting S^*. However, unless $p(\theta|D,H)$ is highly peaked at its maximum, inaccuracy is introduced because the $f(E;\theta)$ evaluated at the point estimate may differ considerably from $f(E;\theta)$ averaged over $p(\theta|D,H)$. Regardless, this is a common practice because usually the estimates can be obtained using

conventional optimization software, the estimates are often accurate enough, and it is convenient to have a single estimate of θ.

Approach 1 involves Bayesian estimation because $p(\theta|D,H)$ is the posterior distribution of θ given by Bayes' Theorem in terms of the likelihood $p(D|\theta,H)$ and a prior $p(\theta|H)$. The research on this approach is set apart from that of more typical maximum likelihood estimation approaches by its focus on finding suitable prior distributions (e.g. Gatsonis, *et al.*, 1991), which we discussed in the previous section. Aside from concerns with the prior, research also attempts to find optimization methods that take advantage of specific forms of $p(\theta|D,H)$. The research most relevant to our work is that involving finite mixture distributions, which represent the pdf of a quantity u by a series summation of simpler pdfs, for example:

$$p(u|H) = \sum_{j=1}^{K} q_j \, h_j(u;a_j),$$

where the $h_j(.)$ are the component pdfs parameterized by a_j, the q_j are the prior probability of u being from the j^{th} component, and the q_j sum to one. (Chapter 5 discusses finite mixture distributions in more detail.)

Titterington, *et al.* (1985) provide a good overview of the theory and applications of finite mixture distributions. These distributions arise quite naturally when the variable space can be partitioned into separate, possibly overlapping regimes. In that case, each component pdf of the mixture represents the information and uncertainty associated with the process in its particular operating regime. Mixture distributions can also be applied as nonparametric probability density estimators (e.g., Mueller, *et al.*, 1992) and have been shown to be able to approximate any arbitrary pdf for certain families of kernel distributions, such as normal distributions (Titterington, *et al.*, 1985).

Solution methods to estimate the parameters of the component pdfs are varied, but a particularly promising tactic is the application of Expectation-Maximization (EM) (Little and Rubin, 1974). EM is a optimization method for "missing" data problems and has been

applied in a variety of fields (e.g., Jordan and Jacobs, 1992; Titterington, *et al.*, 1985; Abraham and Chuang, 1993). In the problems we consider, the missing data are the assignments of each data point to each component of the mixture. So, given the data D and an initial guess for θ, EM iteratively estimates θ by first taking an Expectation Step (E-Step) to determine from which component each data point is most probably sampled. Then, EM takes a Maximization Step (M-Step) to find the most probable θ given the estimated knowledge of which points were sampled from which component. Chapter 6 gives a more detailed description of these methods and shows that EM usually is an efficient method because of the simplicity of the calculations in both the E-Step and M-Step when specific forms of the components, such as normal distributions, are used.

Approach 2 attacks the approximation of the integral head on. It uses the simple idea that the expectation of a function can be approximated by the sample mean. However, generating a sample that has significant probability mass as measured by $p(\theta|D,H)$ can require extensive calculation, especially in cases when θ-space is high-dimensional. So, the success of this approach relies on an efficient means of sampling $p(\theta|D,H)$ to yield a sample representative of the distribution.

The second approach is most commonly implemented as Markov chain Monte Carlo (MCMC) sampling methods. Neal (1993) gives an excellent review of these methods, which include Gibbs sampling and variants of stochastic dynamic simulation and simulated annealing. Gibbs sampling (e.g., Mueller, *et al.*, 1992; Casella and George, 1992) solves the problem by successively sampling from the full conditional pdfs with respect to each parameter rather than directly sampling the joint pdf $p(\theta|H)$, which is usually more complicated than the full conditionals. For example, if $\theta=\{\theta_1,\theta_2,...,\theta_p\}$, then the full conditional for θ_1 is $p(\theta_1|\theta_2,...,\theta_p,H)$; and at each iteration $p(\theta_1|\theta_2,..,\theta_p,H)$ is sampled, then the new θ_1 is used in sampling $p(\theta_2|\theta_1,\theta_3,...,\theta_p,H)$, and so on. However, Gibbs sampling has drawbacks in that the convergence to an appreciable probability mass can take many samples and, for some problems, it is difficult to determine or compute the full conditionals. To get around these problems, extensions to method have been proposed

such as using a Metropolis algorithm variant that accepts a point only if $p(\theta|H)$ is greater than its value at the previous point, otherwise it rejects the sample. Still, this may not work well because the rejection rate may be unreasonably high and the algorithm will take a long time to accumulate a representative sample (Neal, 1993).

Neal (1995) proposes a hybrid MCMC method that avoids the problems of Gibbs sampling by imposing a stochastic dynamic simulation (statistical physics) framework. He found the method improved over the Gibbs sampling algorithms by allowing greater distances between sampled points (as measured in θ-space), thus avoiding random walks and traversing the space more thoroughly and quickly.

In Chapter 6, we present a summary of our brief experimentation with MCMC methods. Generally, the parameter estimation methods of Approach 1 yield estimates much more quickly than the methods of Approach 2. However, MCMC methods are the state-of-the-art for solving probabilistic problems for which a simple point estimate is inadequate.

2.1.3 State Estimation from Corrupted Data

In industry, inferences about the state of a process are made by interpreting vast quantities of data gathered from sensors. These inferences dictate the control actions and planning strategies chosen to maximize profit. However, data corruption due to sensor malfunctions or other stochastic phenomena that cause gross errors often compromises the quality of these inferences. In order to improve the quality of the data and hence inferences based upon the data, data rectification (or reconciliation) is performed. Rectification removes gross errors and adjusts data to more probable values. When applied to dynamic process data, rectification can be viewed as a means of detecting and compensating for additive outliers in time series.

The problem of simultaneously estimating the parameters of the model, rectifying the data (estimating current and past states), and predicting future states is a difficult one that has been addressed largely by research in robust state estimation (e.g., robust Kalman filtering)

and outlier detection in time series analysis. The literature on this topic is substantial. Here, we briefly review the research most relevant to ours. The research of Abraham and Chuang (1993), Schick and Mitter (1994), Ljung (1993), and Chen and Liu (1993) serve as examples of the state of the art.

Abraham and Chuang (1993) have applied EM to the simultaneous estimation of model parameters and outlier effects in time series. Like the approach proposed here, Abraham and Chuang (1993) use a Gaussian mixture model to represent the error pdf. However, they simultaneously estimate the parameters, the prior probability ε, the variance of the noise σ^2, the location (mean) of the outlier pdf, and the conditional posterior probability of each observation being an outlier given the observations x_t for all times t=1, 2, ..., N.

Despite the seemingly comprehensive nature of their approach to outlier detection and parameter estimation, the formulas of Abraham and Chuang (1993) are only applicable to the case of all outliers having the same sign and the outlier pdf having the same variance σ^2 as the noise pdf. Their approach would overestimate the variance of the noise in cases in which outliers have positive and negative signs and are more variable than the observation noise. Also, the type of outliers must be identified prior to applying this method because different EM formulas apply depending upon whether the outliers are additive or innovational. Moreover, the computation necessary at each M-Step of the EM algorithm is much greater than that required by the approach in this work and becomes prohibitive for even low-dimensional multivariate problems and for models with high-order autoregressive representations (e.g., finite moving-average (MA) models).

A number of researchers have used state space representations and investigated Bayesian methods to estimate the process state and outlier effects assuming that mixture models adequately represent the observation errors. Much of this work has been done in the context of robust Kalman filtering. Of this research, the most relevant to ours is that of Dr. Irvin Schick, a researcher in the Laboratory for Information and Decision Systems (LIDS) here at MIT. Schick and Mitter (1994) apply a Bayesian approach for recursive

state estimation in the face of stochastic outliers. They represent the pdf of the observation noise as a mixture model. The approach generates an (m+1)-component mixture model to account for uncertainty in the decision of whether each observation in a data window of m observations was an outlier or not. As a result, their estimator is a parallel bank of m+1 weighted Kalman filters: the first represents the single event that no outliers were present, and the remaining m filters represent the events that a single outlier was present in each of the respective m observations. This is but a small subset of the 2^m possible events in the full m-dimensional binary event space. Regardless, each filter acts as a smoother in estimating an observation by accounting for information present in the prior and subsequent observations within the window. Given ε, which is the prior probability of an observation being drawn from the outlier distribution, the contribution of the first Kalman filter to the estimate of the current state has weight $(1-\varepsilon)^m$, which is the prior probability of the event that there were no outliers in the window of m observations. The remaining m filters are each weighted by $\varepsilon(1-\varepsilon)^{m-1}$, which is the prior probability of the event that only a single outlier occurred among the m observations. The window width m increases with decreasing ε, but it is not clear how large it would be for a given multivariate process.

While the approach introduced in this work and the two mentioned above use a probabilistic model for the outliers, the methods proposed by Ljung (1993) and Chen and Liu (1993) assume that the outliers are deterministic. Using a deterministic model allows the outliers to be readily estimated by maximum likelihood. However, before the outliers can be estimated they must be detected. Ljung (1993) presents test statistics that are appropriate for detecting additive outliers for the cases of known and unknown model parameters. In her approach, Ljung estimates an outlier at time t as the difference between the observed value and the smoothed value $E[x_t \mid x_{[t]}]$, where $x_{[t]}$ is the complete time series except without x_t and $E[x_t \mid x_{[t]}]$ denotes the expectation of $p(x_t \mid x_{[t]})$. Hence the estimate takes advantage of observations at previous and subsequent times. Chen and Liu (1993) simultaneously estimate the model parameters and additive outliers, innovational outliers, level shifts and temporary changes in the time series. Their approach is an iterative

technique that alternately (*1*) detects and removes the effects of outliers and (2) regresses the model parameters. They also present a variety of test statistics to detect the different kinds of outliers. However, applying the test statistics to individual and sets of observations can be combinatorially explosive.

Probabilistic error models, as proposed by Schick and Mitter (1994) and Abraham and Chuang (1993), and deterministic outlier models, as proposed by Chen and Liu (1993) and Ljung (1993), have their advantages and disadvantages. The deterministic models allow quick calculation of outliers once they are found, but suffer from the need to apply special-purpose test statistics and the combinatorial problem of locating the possibly multiple outliers. The probabilistic models allow joint estimation of the probability of each observation being an outlier, but suffer from the complexity of the resulting joint posterior pdf (or likelihood) and the computation necessary to maximize it.

2.2 Statistical Modeling

Statistical theory provides a basis for model synthesis and validation of stochastic processes. The research described in this section is set apart from that of the probabilistic paradigm described above because most of this research adopts a "frequentist" or classical view of probability rather than exploiting the full interpretation of probability theory. Regardless, the research in statistics offer many relevant methods for model formulation and parameter estimation; and many of them are proven to be special cases of the more general probabilistic paradigm and Bayesian methods (e.g., Gruber, 1990). The research in this field also defines metrics and algorithms to assess model reliability using confidence intervals and predictive error criteria and to detect extrapolation by identifying the degree of novelty of the inputs. For example, nonlinear statistical modeling addresses many of the key issues of this project, particularly in model analysis and rigorous proofs of model properties. Gallant (1984) presents the theory for synthesizing and analyzing nonlinear

models. Also, research in nonparametric and semiparametric modeling offers some insight into types of prior knowledge and the incorporation of this knowledge into models.

2.2.1 Nonparametric Modeling

The theory behind nonparametric models is extensive. Generally, nonparametric models are a series sum with a data-dependent length. They include kernel estimators and splines (Eubank, 1988; Härdle, 1990) and multivariate methods such as generalized additive models (Hastie and Tibshirani, 1990; McCullaugh and Nelder, 1989), projection pursuit (Friedman and Stuetzle, 1981), multivariate adaptive regression splines (MARS) (Friedman, 1992) and sliced inverse regression (Duan and Li, 1991). Also, smoothing splines; minimax estimators; nearest neighbor estimators and other clustering techniques are considered to be nonparametric methods. Additionally, neural network models – discussed below in Section 2.4 – can also be framed in a nonparametric context if the network architecture (number of nodes) is selected according to the training set data. The inclusion of prior knowledge into nonparametric models has also been investigated in nonparametric modeling (Härdle, 1990). Prior knowledge may take several forms: regularizers (to smooth or flatten the model); fixed points such as limit points, zeros and inflections; isotonic regression, e.g. shape-invariance and monotonicity; and semiparametric models.

2.2.2 Semiparametric Modeling

In the absence of a parametric component, nonparametric models suffer from the several disadvantages: they are overly dependent upon the data, they may produce predictions that are inconsistent with process constraints, and they suffer from unreliable extrapolation. A semiparametric model combines parametric and nonparametric models to form a composite model. Wahba (1990) has investigated semiparametric splines, which are motivated from regularization methods. McCullagh and Nelder (1989) and Hastie and Tibshirani (1990) have explored generalized additive models, which add a "link" function

serving as a nonlinear transformation on the output of a linear model. Such an approach has similarities to the use of a parametric output model in the function-oriented approach we propose below in Chapter 3.

2.2.3 Prediction Error Criteria

In nonparametric modeling, the data play a central role in determining the number of parameters as well as the parameter values. A nonparametric model has the flexibility to conform to the present data. As a result, often model selection is guided by the desire to choose the simplest model that adequately describes the data. Imposing dual objectives of simplicity and fit to the data helps to avoid over-parameterization and hence overfitting. However, these are often competing objective that results in the bias versus variance tradeoff described well by Geman, *et al.* (1992). Quantifying this tradeoff is critical in an attempt to derive measures of predictive or generalization capability.

Resampling techniques, such as leave-one-out and S-fold cross validation, are commonly used for determining the generalization capabilities of a model. Described more fully in Section 3.5, resampling methods estimate the model parameters multiple times over different partitions of the same data set. The most popular are leave-one-out cross validation (CV) (Stone, 1974), variants such as S-fold cross validation (SCV) (Zhang, 1993b), and repeated learning and testing (RLT) methods (Burman, 1989). Cross validation and jack-knifing (Efron, 1982) techniques avoid excessive variance in model predictions due to overfitting and yield measures of prediction error to determine the generalization capability of the models. These measures have limited utility from a synthetic view, where a means of selecting or emphasizing the specific knowledge that most greatly improves model generalization is sought. Moreover, Davison and Hall (1992) have shown that resampling methods suffer from high variance in their estimates of prediction error, and CV and SCV have been proven to be highly biased for small to moderate sample sizes (N<50 for univariate data) (Burman, 1989; Eubank, 1988).

There are also alternative choices for a measure of prediction accuracy. Härdle (1991) lists several squared error measures. Along with cross validation, there are complexity-penalized measures such as general prediction error (GPE) (Moody, 1992), FPE (Akaike, 1974), AIC (Akaike, 1970), and generalized cross validation (GCV) (Craven and Wahba, 1974), which require an estimate of the effective number of parameters in the model (i. e., the model degrees of freedom) but do not require repeated sampling and re-estimation of the parameters. Presented in Chapter 3, part of our research derives such a complexity-penalized metric that is more efficient than resampling methods and is potentially more accurate than the criteria listed above.

2.3 Computational Learning Theory

Computational learning theory (machine learning theory) attempts to define models of learning and implement them in computer technology. At the heart of this research is the role of inductive bias in learning. Buntine (1991) provides an overview of some of the problems in this field. Major issues concerning prior knowledge include determining (1) the relationships between the specific types of inductive bias in a problem domain, (2) the representation of these relationships in the model structure and the learning metrics, and (3) the algorithms used to train the model. The project proposed here addresses these issues in the domain of chemical process modeling.

2.3.1 Vapnik-Chervonenkis Dimension

Computational learning theory provides a metric for quantifying inductive bias in Boolean concept learning and inductive classification problems (Haussler, 1988). Concept learning involves the selection of a concept (e.g. a classification rule, pattern matcher, or real-valued function) from a concept class or set of concepts usually with similar structure. The metric is the Vapnik-Chervonenkis dimension (VC-dim):

> In learning a target concept in the concept class C of candidates, using examples drawn from an instance space X: VC-dim(C) m_{max}, where m_{max}

is the largest m such that there exists a set of m examples $\{x_1, x_2, ..., x_m\}$ drawn from X that can be labeled in all possible ways by the concepts in C.

To understand the implications of the VC-dim($\mathcal{C}$), note that as long as the number of examples $m \leq$ VC-dim($\mathcal{C}$) a hypothetical example set $\{x_1, x_2, ..., x_m\}$ can be labeled in all possible ways, and therefore could have been generated by any one of the concepts in $\mathcal{C}$. At the point that the number of examples m exceeds VC-dim($\mathcal{C}$), the examples can no longer be labeled in all possible ways. It is at this point that the inherent biases (constraints) within the concept class become evident and some of the concepts within $\mathcal{C}$ may be ruled out as the generating source behind the example set. However, because its underlying assumptions consider only the size of the sample and not the contents or prior knowledge about the underlying distribution of the sample, the VC-dim provides only a worst-case lower bound on the number of training examples needed by an algorithm. In the worst case, a learning algorithm will require at least VC-dim($\mathcal{C}$) examples to be able to distinguish between the candidate concepts in $\mathcal{C}$.

The more restrictive the inductive bias within $\mathcal{C}$, the fewer the examples required to manifest it, and the lower the VC-dim($\mathcal{C}$). Simpler, more restrictive concept classes, such as Boolean conjunctions, have low VC-dim; while richer, more expressive concept classes, such as trigonometric function space, have high or even infinite VC-dim. As VC-dim increases more data are required to learn the underlying concept or function of the domain and, generally, learning becomes more difficult.

VC-dim may be used to quantify the effects of inductive bias on the minimum number of examples required to learn concepts using neural networks. Baum and Haussler (1989) and Cohn and Tesauro (1992) are examples of researchers who have applied VC-dim sample size estimation to neural networks in order to determine relations between the network architecture (number of nodes), domain dimensionality (number of variables), and data set size (number of examples). Baum and Haussler (1989) present a means of determining the minimum data set size necessary to learn the underlying classification

relationship of the examples using a neural network. They found that a lower bound on VC-dim is the number of weights in the network, which provides a useful rough estimate of VC-dim.

Despite their utility in rather simple concept learning problems, where the concept classes are well-defined and usually finite and the example set is usually noise-free, VC-dim techniques cannot be readily extended to function approximation applications such as process modeling. Quantifying the inductive bias in real-valued function approximation problems in the presence of noisy data remains an open research issue. Furthermore, because VC-dim is a worst-case bound, it has limited utility in practice. Cohn and Tesauro (1992) performed numerical experiments that measure the generalization capability of neural networks to determine the tightness of the VC-dim. They found that in some cases the approach to perfect performance is exponential in the number of examples m, rather than the $1/m$ approach typical of VC-dim calculations for PAC ("probably approximately correct") learning. In a practical application, modifications to the learning problem may improve upon the VC-dim bounds by adopting an alternative to the PAC learning framework. An example of this is the "reliably probably almost always usefully" learning framework of Rivest and Sloan (1988), which benefits from relaxing the uniform convergence constraint by allowing an algorithm to exploit partial recognition of a concept. Thus, the VC-dim is a broad guideline for choosing sample size and comparing the influence of different inductive biases in classification problems. For process modeling, other more suitable metrics of inductive bias are sought, such as average generalization capability put forth by Tishby, *et al.* (1989).

2.3.2 Generalization and Occam's Razor

A central point in machine learning is to derive a learning algorithm that when given as input a data sample of observation tuples consisting of inputs $x \in D_X$ and outputs $f(x)$ will produce as output a hypothesis $h(x)$ that is accurate in predicting $f(x)$ over the domain D_X, from which the sample was taken. In any nontrivial case, the sample inputs are a proper

subset of D_X, and it is desired that the hypothesis h(x) still be accurate for those inputs outside of the sample data set. This generalization capability is important and is at the crux of synthesizing accurate process models for control and optimization applications.

Occam's razor, the principle of parsimony, has been invoked to derive learning algorithms that prefer simpler hypotheses to more complex ones in order to achieve good generalization (predictive accuracy). Such a preference for simplicity can be rationalized on the grounds that if two hypothesis have equal training accuracy the simpler one should have greater predictive accuracy because it is less likely to have been trained to the specific idiosyncrasies of the training sample. Although empirical evidence supports this, only recently has any theoretical basis been developed to validate that this principle holds for specific hypothesis classes (such as when the target function is assumed to be of a known limited size (Blumer, *et al.*, 1987). However, as argued by Buntine (1991), if the true function itself is complex, there is reason to believe that the more complex of equally accurate hypotheses would generalize better. Furthermore, determining the appropriate metric for simplicity (or complexity) for a specific problem domain is an open issue. Yet, the facts that simpler hypotheses tend, in general, to be easier to train, apply and interpret provide motivation for exploiting Occam's razor in process model synthesis. The question is "how best to do so?"

Wolpert (1990a) directly addresses the issue of selecting appropriate simplicity measures and concludes that the "optimal simplicity measure" must be a function of the predictive ("guessing") distribution, and thus of the architecture, novel inputs, training data and restrictions (such as training error as a function of noise distribution). Once provided with such a measure, one may determine the applicability of Occam's razor to a specific problem domain by examining the "Occam error." Occam's razor is applicable to a problem if the Occam error is negative.

Though Wolpert's treatment of Occam's razor is mathematically rigorous, it does not directly address the formulation of the optimal simplicity measure for any particular

example and the exact form of the Occam error is strictly a theoretical quantity, which must be accompanied by simplifying assumptions to be applied in practice. In fact, Wolpert admits that "just calculating guessing distributions ..., never mind simplicity distributions, is a calculational nightmare." It requires extensive Monte Carlo simulation for any nontrivial architecture, such as neural networks. Regardless, Wolpert provides a starting point for critically analyzing the value of imposing Occam's razor in model synthesis and its impact on the generalization capabilities of the resultant models. Further research to wrest this theory into a practical framework is required.

2.3.3 Shifting Inductive Bias and Iterative Induction

Research in computational learning theory also presents several approaches to exploiting and evaluating the effects of prior knowledge for the purpose of iteratively improving the induction process itself (e.g., Buntine and Stirling, 1991) – in other words, "learning to learn." Such work serves as the basis for exploring an interactive model synthesis methodology that relies upon refining the biases by acquiring and discarding user-specified prior knowledge. In a process modeling context, the biases are the model constraints, the objective function, the prior distributions (in a Bayesian approach), and the training algorithm itself. Shifting bias amounts to iteratively training new models and adjusting the biases at each iteration in accord with the user's satisfaction with model predictions. This is a potentially powerful approach because each training yields a model that provides additional information about the problem space and the "true" underlying process model.

A primary example of work in this area is that of Utgoff (1986). Utgoff demonstrates that (1) search for a better bias is a fundamental part of the learning task and (2) it can be mechanized. His "Shift To A Better Bias" (STABB) system was used in concert with LEX, an integral calculus equation solver. STABB improves learning by dynamically adjusting the equation solver's concept description language. Central to this approach is the choice of an appropriate representation of the biases and a means of determining when and what shifts in the bias should be made. Utgoff represents the bias as a concept

description language based upon predicate calculus. Shifts are identified when the version space for the hypothesized concept becomes empty. STABB then shifts the bias by either deducing new operators or syntactically manipulating existing concepts to form new concepts.

Applying computational learning theory in econometric modeling, Freedman and Stuzin (1991) have proposed an interactive system for model tuning. This method is designed to improve forecasting with dynamic models. After synthesizing the model, the iterative method incorporates qualitative suggestions from the modeler into the next iteration of model refinement. The tuning process combines the model error history at past times, subjective suggestions from the user, and a tentative model prediction to form the current prediction. It may be possible to automate this approach to iteratively improve the performance of dynamic process models.

2.3.4 Model Discovery

Researchers in computational learning theory have also investigated knowledge-based approaches to model discovery, primarily with applications in engineering design. Rao *et al.* (1991) have developed KEDS, a knowledge-based equation discovery system, which uses candidate models to drive the selection of subsets of process data and iterative regression of process models specific to subregions of process input space. The approach attempts to automatically learn the mathematical relationships underlying process data by recursively regressing the parameters of a fixed set of functional templates. The major disadvantage to this approach is that the combinatorial nature of the problem quickly renders it intractable. For example, Rao *et al.* report the results of KEDS on a 4-input, 3-output process with 200 data points: KEDS generated 2000 candidate formulae and required 16 hours of run-time on a Texas Instruments Explorer II. Moreover, the fixed set of templates presents a bias that limits the applicability of the resultant models. This is not a problem in our approach since the nonparametric component compensates for any adverse effects of the bias introduced by the parametric component. Additionally, the lack

of prespecified domains on candidate models forces KEDS to search for candidates and to determine domains only with regard to the data. The NP-completeness of the search problem results in the eventual intractability of the approach. Finally, if the data are noisy and hard constraints are present, the KEDS methodology will synthesize models that could yield predictions inconsistent with the real process.

Another approach, applied to engineering design, is AIMS (Adaptive Interactive Modeling System) developed by Tcheng, *et al.* (1991). AIMS is intended to induce fast simulation models for use in interactive design. The system uses recursive splitting to divide the input space into multiple regions and fit linear models in each region. Tcheng *et al.* focus upon selecting the best inductive bias in the form of the induction algorithm used. For this purpose, AIMS uses a multiple objective optimizer to choose among alternative induction techniques (from ID3 to back-propagation networks) by determining the Pareto optimal values for the objectives of predictive accuracy, model formation time, and model evaluation time. Though, AIMS does not explicitly use qualitative prior process knowledge, the approach might be generalized to incorporate qualitative knowledge represented as multiple, competing objective functions.

2.3.5 Summary

Though computational learning theory is a starting point for exploration in combining prior knowledge with neural networks for process modeling, its usefulness is diminished by the fact that process modeling is a problem in approximating real-valued functions. Computational learning theory focuses almost solely upon Boolean classification problems. Much of the theory that can be readily proven in the classification problems typical of machine learning cannot be readily transformed into a real-valued function approximation context. A bridge between or unification of statistical regression theory and computational learning theory is needed, as discussed in Buntine (1991). The closest

thing to a complete theory of scientific inference that is capable of achieving this unification is probability theory (Jaynes, 1993; Howson and Urbach, 1988).

2.4 Artificial Neural Networks

As nonparametric models, artificial neural networks (ANNs) are receiving much attention in process model synthesis. ANNs such as three-layer backpropagation networks and radial basis function networks have been proven to be universal function approximators (Cybenko, 1989; Poggio and Girosi, 1990a; Hornik *et al.*, 1989). This ability to approximate arbitrarily complex functions has been exploited in applying ANNs as models of chemical processes (e.g., Di Massimo *et al.*, 1992; Bhat *et al.*, 1990; Pollard *et al.*, 1992). The major advantage of neural network models is that they can be synthesized without detailed knowledge of the underlying process. Only after presentation of the data does the behavior of a neural network conform to the specifics of the particular process. This property is common to the larger class of flexible functional form models, which are the nonparametric models mentioned in Section 2.2.1.

However, like other nonparametric models, the lack of a process-based internal structure is a liability for the neural network when faced with sparse, noisy data. For example, if a neural network is used to predict the composition of a reactive mixture, first principles dictate that the outputs are non-negative and that they sum to one if expressed as mole fractions. Due to the limited amount of training data and noise in the data, the network outputs may not conform to these process constraints. This inconsistency becomes more severe as the network makes predictions beyond the limits of the training data (i.e., extrapolates). For mole fractions, the problem can be corrected by applying a normalization transform upon the network outputs. In general, though, a more elaborate fixed form or parametric model must be applied.

Insufficient data hampers the accuracy of a neural network because the network relies completely upon the data when inducing process behavior. However, qualitative

knowledge of the function to be modeled may be helpful in overcoming data sparsity. For example, an engineer may know that a process variable increases monotonically in approaching an asymptotic limit. Both the monotonicity and the asymptote are prior knowledge that should be enforced upon the neural network model. Unfortunately, sparse data may prevent a neural network from capturing either of these behaviors.

Neural network researchers have attempted to formalize the intuitive notion that the *a priori* inclusion of more information facilitates model synthesis. There are several approaches to incorporating specific types of prior knowledge. The general themes are to either explicitly impose *a priori* structure upon the network based upon prior knowledge of the problem domain or to give the training algorithm a bias towards specific models, such as simpler networks. The approaches include structure specification, structure modification, and algorithm bias.

2.4.1 Constraining Architecture and Weights

In an approach which straddles structural and algorithmic bias, Joerding and Meador (1991) directly address encoding prior knowledge into neural networks. Their approach uses architectural constraints and weight constraints separately or together to incorporate prior knowledge including curvature, interpolation points, and relationships between output variables. Architectural constraints involve selecting a network with a specified structure and a known behavior, such as normalized outputs. Their work, however, has the drawback that they derive weight constraints from *necessary* conditions for satisfaction of monotonicity or convexity of an output with respect to *all* inputs, rather than the more stringent sufficient conditions and with respect to a specific subset of inputs.

2.4.2 Modular and Hierarchic Networks

In the class of structure specification techniques, hierarchical or modular neural networks have been proposed as a means of providing strong hints (biases) in the training process. Mavrovouniotis and Chang (1992) propose modeling a process using a hierarchical

network built from smaller subnets based upon prior knowledge of the underlying relationships between process variable. The network structure is based upon knowledge of the structure of the physical system, the task at hand, and the kinds of patterns that are important or most likely. They report improvements in training efficiency, prediction performance, and interpretability of the resultant model. However, unlike the approach proposed in our work, Mavrovouniotis and Chang do not explicitly include parametric model components nor enforce the satisfaction of process constraints imposed by nature and design considerations. As a result, the hierarchical network is still prone to producing inconsistent outputs, though less so than a pure black-box neural network.

Johansen and Foss (1992a, 1992b) and Johansen (1994) derived a modular approach for adaptive process control. They employed a parallel approach to generate different model behaviors in different regions of the process input space. In their network architecture, each parametric model is weighted according to a Gaussian response function centered in the region of applicability of that model. The research of Murray-Smith (1994) has strong similarities to this.

Jacobs, Jordan and Barto (1991) and Jacobs and Jordan (1991, 1992) propose a competitive learning approach to modular neural networks. The approach includes prior knowledge as probability distributions that weight the contributions of several "expert" networks. The probability distributions take the form of "gating" networks that may either be induced from the process data or specified *a priori*. This approach was developed in a purely nonparametric neural network framework, but could be enhanced to include parametric "expert" networks. This method is similar to the approach proposed by Johansen and Foss (1992b). However, Jacobs and Jordan do not restrict the gating or weighting networks to be Gaussian or to even have local support, and they give formal probabilistic interpretations to the gating network outputs. Adopting this probabilistic framework leads to a derivation of the posterior probability distribution of the model parameters. The maximum of this distribution gives the parameter estimates. Other

statistics from the distribution lead to confidence intervals on the parameters and the means to evaluate model generalization.

Su *et al.* (1992) also have incorporated prior knowledge of process relationships into a neural network architecture. Their approach combines a parametric model and a neural network in parallel. In this approach the model serves as an idealized estimate of the process or a best guess at a process model. The neural network, which is trained on the residual between the data and the parametric model, compensates for any uncertainties that arise from the inherent process complexity.

A series combination of a neural network and parametric model has also been tried. In a series approach, the neural network estimates intermediate variables to be used in the parametric model, which may be derived from equality constraints upon the variables. Jordan and Rumelhart (1992) refer to the training of such a model as "distal supervised learning" and present a neural network approach applied to dynamic control. Notably, their approach accounts for the inclusion of additional constraints into the objective function of the training algorithm. In a less elaborate approach, Psichogios and Ungar (1991) modeled a fermentation by applying the network to estimate the specific growth rate, which served as an input to the parametric model derived from the component mass balances.

Although the semi-parametric methods listed above enforce model consistency with process equality constraints, they do not guarantee accurate response, satisfaction of inequality constraints for all possible input combinations, nor reliable extrapolation. Although Jordan address some of these issues (1992), additional prior knowledge must be brought to bear upon the problem to achieve all of these properties.

2.4.3 Heuristics and Fuzzy Logic

Work has also been done on the explicit inclusion of qualitative heuristic knowledge into neural networks. These techniques include representing Boolean and fuzzy logic

computation in a neural network architecture. Research has been done to generate neural networks from if-then heuristics, as well as to interpret neural networks as if-then rules. These approaches are primarily used in classification applications rather than function approximation.

In a control and diagnosis application, Lin and Lee (1991) explore the inclusion of prior knowledge in the form of fuzzy set membership functions by combining unsupervised clustering to define the functions and supervised training to determine connection weights. In a computer vision application, Krishnapuram and Joonwhoan (1992) derive a scheme with a similar objective of using fuzzy set membership functions as node activations in an attempt to aggregate inexact and incomplete information from multiple sources. They experimented with a hierarchical network architecture that was based upon prior knowledge of the problem domain and automatically refined by aggregating subnets according to pre-defined connective operators. These approaches at integrating qualitative classification heuristics into neural networks are inflexible in the sense that the biases imposed by the choices of node type and architecture severely restrict the applicability of the methods. However, these techniques may find application in this project as a component of the broader methodology to incorporate all prior knowledge.

2.4.4 Self-Organizing Networks

Another trend in three-layer neural networks is to apply self-organizing techniques to determine the salient features of input space by data clustering and then to regress the final linear layer weights. In these methods the criteria for (1) including points into the same cluster, (2) splitting clusters, and (3) determining the dimensions of the region of influence of each cluster are strong inductive biases that usually originate from the modeler's prior knowledge of the general model synthesis problem. The most representative of these techniques are the radial basis function approaches that use either supervised or unsupervised clustering to determine the number and location of the hidden layer nodes as we have done (Kramer *et al.*, 1992). Also, Constrained Topological Mapping (CTM), a

variant of Kohonen nets, has been proposed for functional approximation by Cherkassky and Lari-Najafi (1991). CTM uses supervised clustering to determine node position and then linearly interpolates between nodes. These techniques are especially appealing due to their ability to be quickly trained. Early work in this project will attempt to broaden their usage to include constrained model synthesis.

Work is also being done in applying principal components or correlation metrics to determine the optimal hidden layer. These approaches are applicable to a variety of node activation functions. Sanger (1991) proposed gridding the space with basis functions that were known to be able to approximate any function. In the presence of a specific data set the optimal linear transformation of these bases into hidden layer nodes was performed. This approach requires the correlation matrix between the basis function weights to be specified prior. If the identity matrix is chosen, the method is equivalent to using the principal components of the basis functions as the hidden layer nodes. The approach suffers from the requirement of a large number of initial basis functions – far more than the actual number of nodes they will ultimately represent – resulting in lengthy calculations to evaluate the function during application. This destroys one of the major advantages of using neural networks in the first place. However, with certain special basis functions such as sines, the hidden nodes may be specified analytically in a much more compact functional form, giving reasonable application times.

2.4.5 Network Pruning and Occam's Razor

Another branch of research in training neural networks focuses upon imposing the strong inductive bias of "Occam's Razor." Smaller networks with fewer connections are preferred to larger more complex networks. This principle is enforced by three main approaches: (1) node-at-a-time construction, (2) network pruning (stripping or skeletization), and (3) simplicity ordering by objective function (learning metric). The first two are structure modification approaches while the latter imposes an algorithm bias.

Invoking Occam's razor avoids overfitting and improves the generalization capability of neural networks.

An example of network pruning is the StripNet algorithm of Bhat and McAvoy (1992). As is common in these approaches, StripNet uses a saliency criterion to determine which connections to trim and a complexity metric E_{com} as a penalty function to bias selection towards simpler networks. The complete objective function F_{obj} is:

$$F_{obj} = E_{ls} + \lambda E_{com}$$

where E_{ls} is the sum of the squared errors (used in least-squares regression), E_{com} is the complexity metric, and λ, $\{0 < \lambda < 1\}$, is a weighting parameter.

In all of these methods, selecting the three components (the saliency criterion, E_{com}, and λ) are the central issues. StripNet uses connection weight magnitude as its saliency criterion, the total sum of the squared weights as E_{com}, and cross-validation to select λ. Bhat and McAvoy (1992) cited results common to other pruning techniques: reduced training time and better generizability.

Correlation metrics similar to those above have been used in node-at-a-time training techniques. For example, the work of Fahlman and Lebiere (1990), who used a cascaded network with each new node connected to all previous nodes; and Sin and DeFigeiredo (1991), who presented an Optimal Interpolating (OI) network that has similarities to Sanger's approach. The OI network is constructed using exponential node activation functions. Preliminary tests on classification problems showed promise. But, our early results on a variant for functional approximation were not encouraging: the networks tended to be larger and less accurate than standard feedforward sigmoidal networks. However, OI quickly generates and trains networks that would serve as good initial guesses for more elaborate training algorithms. If the technique can be more suitably adapted to functional approximation it may prove to be an extremely efficient means of regressing neural network models.

2.4.6 Bayesian Learning Networks

Bayesian modeling is rapidly coming to the fore in neural network research. Buntine (1991), MacKay (1992a, 1992b, 1992c), and Neal (1995) are the primary advocates of this approach. Bayesian techniques suggest a Bayes optimal formulation for the objective function that includes prior knowledge in the form of a prior probability distribution on the network weights. Typically, priors help in inducing the smoothest or flattest models, though any manner of distribution may be posed as a prior in order to induce other behaviors. Use of prior distributions to enforce smoothness or flatness is referred to as regularization, and the priors themselves are called regularizers. MacKay (1992a) builds upon research by Skilling and Gull (1987) to show how to use the training data to choose among different candidate regularizers and to compare alternative models in a Bayesian interpolation framework. MacKay (1992a) also argues that Occam's razor is inherent in Bayesian interpolation. Subsequently, he applies these techniques to back-propagation neural networks (MacKay, 1992b, 1992c).

Levin *et al.* (1989) and Tishby *et al.* (1989) have also posed the neural network synthesis problem in a probabilistic framework. They illustrate the use of Bayesian techniques and maximum entropy to determine the generalization capabilities of the network and to estimate the minimal data set size necessary for adequate generalization.

2.5 Functional Analysis

Functional analysis is the branch of mathematics concerned with the definition, semantics, analysis and approximation of functions. The function approximation research in this field involves approximating a true function assumed to be a member of a specific function space and is important to the solution of inverse problems. The research most closely related to our research defines different norms and their resultant Hilbert spaces (e.g., Wahba, 1990; Johansen, 1994; Poggio and Girosi, 1990a and 1990b). Prior knowledge is incorporated into the function analytic approaches by defining an appropriate norm. The norm is based on any constraints and preferences to be imposed on the members of the

space. Preferences, such as smoothness of the function, help determine the form of the norm and serve as regularizers (penalty functions) in parameter estimation. However, it is often difficult to derive a suitable norm that captures the prior knowledge.

Poggio and Girosi (1990a and 1990b) present a function analytic justification for radial basis function networks (RBFNs) by deriving them as universal approximators to members of a normed space. Johansen (1995) applies similar ideas in performing system identification using nonparametric models. Wahba (1990) presents a function analytic derivation in Hilbert space to define a class of splines. From this basis, she extends the Generalized Cross Validation (GCV) criterion for estimating prediction error and the regularizing constant for parameter estimation of the splines.

2.6 Mathematical Programming

Most of the approaches above address the specification of the model, objective function, and constraints. However, the actual training of the model continues to be a computationally intensive process. The current work attempts to develop more efficient algorithms that explicitly take advantage of the different forms of prior knowledge. Exploration of semi-infinite programming and constraint propagation may give insight into more efficient algorithms for parameter estimation subject to simultaneous constraints on the parameters and the state variables.

2.6.1 Infinite Constraints and Semi-Infinite Programming

Parameter estimation can be posed as a mathematical programming problem. Traditionally, unconstrained nonlinear programming algorithms have been used in regression. However, Gallant and Golub (1984) demonstrate that the resulting models may not obey known process constraints, and that a constrained algorithm, with constraints enforced only at each data point, greatly increases the region of valid model predictions.

Though, imposing constraints at the data points improves generalizability, the model is still not guaranteed to be consistent with process knowledge everywhere in process input space. Enforcing constraints over a continuous domain of one or more of the process inputs transforms the parameter estimation problem into a semi-infinite programming (SIP) problem. In an SIP, the objective function is dependent upon a finite set of decision variables -- the model parameters; and the constraints depend upon not only the parameters but the infinite possible values of the input variables (Kramer *et al.* 1992). There is a substantial body of literature on SIP theory and algorithms (e.g.: Fiacco and Kortanek, 1983; Tanka *et al.* 1988). However, difficulty in maintaining computational tractability limits the utility of these methods to problems with low dimensionality. For practical computation, the constraints must involve only two or three infinite variables. To reduce computation, imposing constraints at the data points and solving the problem as a finitely constrained NLP will yield an estimate to the optimal parameters of the SIP. This estimate can serve as the initial guess for the full SIP.

2.6.2 Constraint Logic Programming

In theory, Constraint Logic Programming presents a means of transforming a semi-infinite programming problem into a finite programming problem by converting constraints on the dependent variables into constraints on the network weights. As such, the constraints may then be used explicitly in posing network training as a constrained optimization problem. In practice, constraint logic programming and constraint satisfaction systems are not yet capable of handling the number and complexity of nonlinear constraints typically encountered in process models. The possibility remains, though, that the theory behind these techniques may be useful in developing more efficient algorithms for semi-infinite programming problems.

Chapter 3

Function-Oriented Paradigm

In the function-oriented paradigm, the objective is to accurately approximate some true function. The focus rests on deriving an appropriate model structure and methods to estimate the model parameters. This chapter explores such an approach. The objective was to derive a process modeling methodology that combines prior knowledge and nonparametric models. The inclusion of prior knowledge is investigated as a means of improving the predictions when models are derived from sparse and noisy process data. This approach arises largely from intuition that adding prior knowledge will improve performance of a data-derived model. Our early successful empirical results motivated further exploration.

In the next section, we define the problem this research addressed. After that we present our approach, including an enumeration of the types of prior knowledge it handles. Moreover, we present a novel way of efficiently performing model validation. We illustrate our modeling methodology with case studies of vinyl acetate polymerization and fed-batch penicillin fermentation. The results show that prior knowledge enhances the generalization capabilities of a pure neural network model. Also, the new validation technique was tested in a series of Monte Carlo simulations and was shown to work well.

The implementation of the full modeling methodology was done in Matlab, the matrix and numerical analysis software by The MathWorks Inc. Appendix C contains the source code. Finally, Appendix A summarizes the notation of this chapter, which differs from that of the remaining chapters. Most significantly, this chapter uses bold-face type to designate vectors and matrices. In subsequent chapters, all variables are considered potentially to be vectors or matrices, and bold-face is not used. In any case, all variables and notation are defined at the point of first use in each chapter.

3.1 Rationale and Approach

The lack of a process-based internal structure is a liability for a nonparametric model when faced with sparse, noisy data. For example, if a nonparametric model is used to predict the composition of a reactive mixture, first principles dictate that the outputs are non-negative and that they sum to one if expressed as mole fractions. Due to the limited amount of training data and noise in the data, the model outputs may not conform to these process constraints. This inconsistency becomes more severe as the model makes predictions beyond the limits of the training data (i.e., extrapolates). For mole fractions, the problem can be corrected by applying a normalization transform upon the model outputs. In general, though, a more elaborate fixed form or parametric model must be applied.

Insufficient data hampers the accuracy of a nonparametric model because the model relies completely upon the data when inducing process behavior. However, qualitative knowledge of the function to be modeled may be helpful in overcoming data sparsity. For example, an engineer may know that a process variable increases monotonically in approaching an asymptotic limit. Both the monotonicity and the asymptote are prior knowledge that should be enforced upon the nonparametric model. However, sparse data may prevent a nonparametric model from capturing either of these behaviors.

To counter these problems we combine prior knowledge with the nonparametric model. Prior knowledge enters the hybrid model as a simple process model and first principle equations. The simple model controls the extrapolation of the hybrid in the regions of input space that lack training data. The first principle equations, such as mass and component balances, enforce equality constraints. The nonparametric model compensates for inaccuracy in the prior model. In addition, inequality constraints are imposed during parameter estimation.

3.1.1 Problem Specification

Taken together, the behavioral relations, their domains, and their precedence with respect to data constitute a process model specification. Process behavior is derived from whatever physical phenomena occur in the process. The domain specification describes where within the process variable space these phenomena occur. Both the process behavior and the domain are specified in the context of a set of process data. Also, each relation is assigned a precedence with regard to that dataset.

Therefore, the problem is specified as such:

*Input (**x**) and output (**y**) domain definitions.* The definitions specify the dimensionality and variable types of the input-output space: $\mathbf{x} \in D_x \subset \Re^n$, $\mathbf{y} \in D_y \subset \Re^m$. D_x and D_y may be hypercubes or regions bounded by arbitrary functions of **x** and **y**. The methodology should generate a model that is accurate within the full D_x and D_y domains. Therefore, the domains define the limits within which extrapolation should be reliable.

*Data model (*D*).* D provides *N* input-output pairs and assume normal i.i.d. noise ε_x and ε_y with unknown variances σ_x^2 and σ_y^2, respectively.

Prior knowledge. Prior knowledge takes the form of equality constraints $\mathbf{h}(\mathbf{x},\mathbf{y}(\mathbf{x};\mathbf{w})) = \mathbf{0}$, finite inequalities $\mathbf{g}(\mathbf{w}) \leq \mathbf{0}$ and infinite inequalities $\mathbf{g}_\infty(\mathbf{x},\mathbf{y}(\mathbf{x};\mathbf{w})) \leq \mathbf{0}$. Each constraint has a domain of applicability D_c and a precedence $p_c \in$ {greater, lower, default} with respect to the data. For quantitative relations, the functional form is entered explicitly. Quantitative equalities serve as the basis for parametric components in the hybrid model. For qualitative equalities, the relation is flagged as a candidate for a nonparametric submodel. Qualitative inequalities are not valid.

Candidate models ($\mathcal{M}$*).* $\mathcal{M}$ defines the family of nonparametric models to be used in combination with the parametric components. The families will

include radial basis function networks (RBFNs) and kernel estimators. These models take the following form:

$$f_N(\mathbf{x};\mathbf{w}) = \sum_{j=1}^{K} \mathbf{u}_j \, a_j(\mathbf{x};\mathbf{v}_j);$$

where $\mathbf{w} = \{\mathbf{u}_1\ \mathbf{u}_2 \ldots \mathbf{u}_K\ \mathbf{v}_1\ \mathbf{v}_2 \ldots \mathbf{v}_K\}$ is the set of parameters; and $a_j(.)$ are the kernel or basis functions. For example, for RBFNs

$$a_j(\mathbf{x}) = \exp\left(-(\mathbf{x}-\mathbf{m}_j)^T \mathbf{S}_j (\mathbf{x}-\mathbf{m}_j)\right).$$

The problem specification is the input to the model-building strategy. Parameter estimation is achieved by minimization of a suitable objective function. This minimization is performed in the presence of constraints that enforce consistency and generality. Equality constraints enter as subcomponents of the model structure that ultimately enter the objective function in the likelihood. Inequality constraints enter as prior probabilities on the model. The synthesis method achieves maximum generalization capability (minimum prediction error) by choosing the model structure that minimizes the estimated prediction mean square error (MSE).

3.1.2 Parametric Models

In the function-oriented paradigm, the equality and inequality constraints are at center stage. First principles in the form of parametric equality constraints impose known structure upon the process model. As in Kramer, *et al.* (1992), a parametric model may prove to be useful as a default model upon which the total model prediction is based. Additionally, a parametric model may be a governing process relation that depends upon unmeasured intermediate variables. These intermediates will depend upon process variables that may be computed directly as the outputs of a nonparametric model. This is illustrated in Figure 3.1, which portrays (a) a serial configuration of a nonparametric model with a parametric prior model, and (b) a parallel configuration with the nonparametric model serving as an additive correction to the parametric prior model.

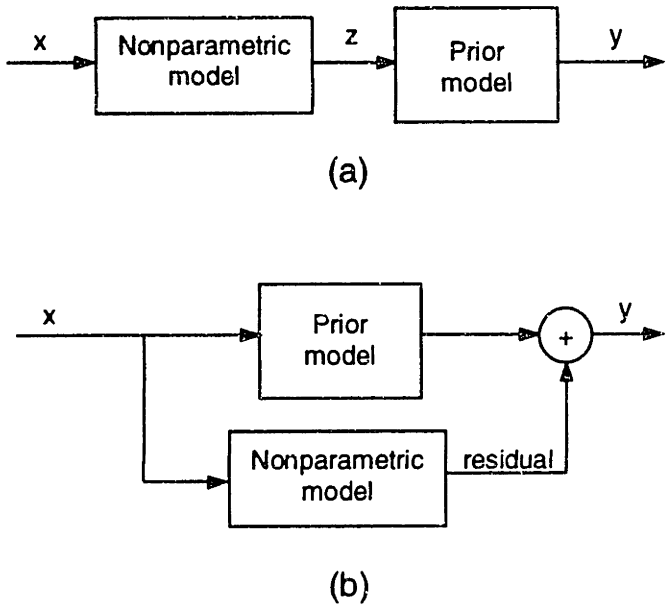


Figure 3.1: Hybrid models: (a) serial configuration; (b) parallel configuration.

3.1.3 Nonparametric Models

In the proposed methodology, the roles of parametric and nonparametric models are complementary. As a result of using a parametric prior model, the nonparametric component of the hybrid is generally smaller and easier to train than a purely nonparametric model. As in Figure 3.1, the nonparametric model may serve as an additive correction to the estimates of a default parametric model or as an integral component resolving the uncertainty inherent in an unmeasured intermediate variable. This leads to a hierarchic decomposition of complex models using nonparametric models as the plaster to fill in the gaps of missing knowledge and to integrate submodels. Section 3.3 presents the nonparametric models investigated in this research.

3.1.4 Prior Constraints

Inequality constraints may be applied to enforce certain types of consistency between the model estimates and the real process. Unlike equality constraints, which are parametric

models that resolve dependent variables in terms of model inputs, the inequality constraints must be accounted for by either restriction of the model structure or by inclusion in the parameter estimation techniques.

3.1.5 Data and Parameter Estimation

In the function-oriented paradigm, the data play a part in the evaluation of the objective function of the parameter estimation problem and in determining the regions of extrapolation, and thus the default behavior of the synthesized composite model. The parameter estimation technique should use the data in a way that takes advantage of the model structure. In neural network modeling this can be accomplished by considering the behavior of the node activation functions. For example, radial basis function neural networks, which are describe in detail in Section 3.3.1, have nodes with local support allowing fast clustering techniques to be used to identify features and determine the input weights. The output weights then are the results of least squares linear regression – assuming that the objective function and constraints remain quadratic in the weights, as is often the case. In general, though, constrained mathematical programming methods are necessary to estimate the parameters.

3.1.6 Model Validation

The next facet of the model synthesis methodology is to determine model reliability. This involves applying techniques for determining model validity, goodness of fit, generality, extrapolation detection, and model comparison. Our research addresses the problem of determining generalization capability by estimating prediction error while avoiding the heavy computational burden of resampling methods like leave-one-out cross-validation, discussed in Section 2.2.3.

Most systems are not known in sufficient detail for a modeler to propose a unique model structure. Typically, a modeler must choose from among many models capable of adequately describing the system. This research addressed the problem of selecting a

model that is nonlinear in the parameters. The criterion for selection was prediction with highest accuracy. Moreover, we investigated this problem in the context of moderate to small data sets and assumed that all of the available data are used in estimating the model parameters. This precludes the luxury of validating the model with data independent of the estimation step. Instead, we derived a more efficient method than resampling techniques, such as leave-one-out and S-fold cross validation, which are commonly used for this purpose (Efron, 1982). Described more fully below in Section 3.5, resampling methods estimate the model parameters multiple times over different partitions of the same data set. We present a cross validation method with best-case performance requiring only a single estimation of the parameters using the entire data set. Also, we investigated prediction criterion in the spirit of Craven's and Wahba's (1979) generalized cross validation (GCV) and Moody's (1992) general prediction error (GPE), which require an estimate of the effective number of parameters in the model.

3.2 Role Assignment of Prior Knowledge

Without going so far as to pursue a universal learning system, this research attempted to develop a system that is capable of combining several types of knowledge in the context of process modeling. Central to this project is the identification of the sources, types and appropriate representations of prior knowledge in a chemical process modeling framework. This section categorizes the types of prior knowledge.

Prior knowledge is knowledge about the process that exists prior to the synthesis of the model. It includes the intended purpose of the model, "hard" constraints imposed upon the process by first principles or design considerations, and preferences in model behavior. Usually this corresponds to process knowledge known prior to the presentation of the data. However, it may include knowledge from prior iterations in the iterative model synthesis procedure. In that case, the knowledge includes information about the performance of prior models.

The model purpose implies general properties of the model structure and performance criteria. For example, an on-line control application implies a dynamic model and possibly an adaptive formulation. On the other hand, a plant-wide optimization application would imply simpler, lumped models for individual process units. In the control application, accurate one-step ahead prediction may be the defining performance criterion. In optimization, accurate steady-state prediction may be most important.

In addition to the purpose of the model, equality and inequality constraints are also prior knowledge. These constraints are considered "hard" or "soft" depending upon their precedence with respect to the data. Hard constraints arise from mass and energy balances and physical restrictions imposed upon the plant by equipment limitations. They may be expressed in terms of equalities and inequalities among model inputs and outputs. These constraints take precedence over the process behavior implied by the data. Conversely, preferences in model behavior are "soft" constraints that influence the regression of the model, but have lower precedence than the data. These preferences may take the form of default relations, which should hold while extrapolating beyond the training data, or as qualitative behavior such as smoothness.

This research focuses on prior knowledge expressible as relations among the process input and output variables. A relation may be an equality or an inequality constraint. Each relation has a specific *domain of applicability* and a *precedence with respect to process data*. A relation may be categorized as quantitative or qualitative according to the degree to which the functional form of the relation can be specified. In this context, *quantitative relations* are explicit equality or inequality constraints with known functional form, while *qualitative relations* only specify that a relation exists between a subset of inputs or outputs. An example of a qualitative relation is the knowledge that several input variables always act together as a dimensionless group in their determination of an output, but how the group determines the output is unknown.

3.2.1 Domain of Applicability

The domain of applicability of a relation specifies the region of input-output space over which the relation applies. Inherent in any function approximation problem are the definitions of the input and output variables and their ranges, which define the domain of applicability of the model. However, function behavior in specific subregions may be known, such as asymptotic limits, zeros, extrema, inflection points, and default behavior. This knowledge may be expressed by partitioning variable space into subregions, each subject to specific relations. A domain may be expressed explicitly in terms of the input variables; implicitly in terms of the inputs through bounds on the output variables; or implicitly in terms of the distribution of the available data in input space.

3.2.2 Type of Relation

Generally, equality constraints appear as submodels of a composite model. Equalities with known functional form are parametric models that reduce degrees of freedom. If the equality has precedence greater than the data it is enforced either as an input preprocessor or an output model depending upon whether inputs or outputs are involved, respectively. If the data take precedence over the relation, the equality serves as a default model that provides a baseline estimate of the outputs. This estimate is refined by nonparametric submodels and controls extrapolation in the absence of data.

Inequality constraints serve as modifiers to the training objective function or appear as constraints during training. Inequalities only impart meaning when their functional form is known. If the inequality has a lower precedence than the data, the constraint represents merely a preference in model behavior. Such preferences can be injected into the estimation objective function as a penalty function in terms of the parameters. When the inequality is a hard constraint, such as non-negativity of a mole fraction, it must be strictly adhered to. Hard inequalities involving only inputs will partition the input space into domains of applicability. This serves as the basis for input data rectification or merely as a screening filter that prevents invalid inputs to the model. On the other hand, if the outputs

are involved in a hard inequality, the relation is an infinite constraint. As such, it may possibly be transformed into an output model. For example, if $g(x;\mathbf{w}) = -y(x;\mathbf{w}) \leq 0$ must hold, then the hybrid could compute intermediate variable $z(x;\mathbf{w}) = y^2$ and the output model would be simply $y(x;\mathbf{w}) = z^{1/2}$. Determination of other possible transformations of infinite constraints into output models will be investigated in this research.

3.2.3 Knowledge of Functional Form

Structural knowledge dictates whether the relation will appear (if known) as a parametric model or (if unknown) as a nonparametric model. In this way the flexibility of nonparametric models may be exploited wherever uncertainty exists about the form of any relation. A relation that can be expressed semiparametrically, as a partly known relation with an unknown contribution, provides the basis for structuring the hybrid model into parametric and nonparametric modules. Regardless of its precedence with respect to the data, an inequality constraint with unknown structure provides no additional useful information. In other words, knowing $g(\mathbf{x}) \leq 0$, *where g(.) is unknown*, provides no additional constraints upon the synthesis problem.

3.2.4 Precedence with respect to Data

Precedence also affects the role of a relation. If the relation has greater precedence than the data, it serves as a consistency constraint. If data have greater precedence, the relation is represented as a default model or a behavioral preference depending upon whether the relation is an equality or inequality, respectively. Note that an equality constraint with unknown functional form and precedence lower than that of the data's provides no useful information.

3.2.5 Variables Involved

Involvement of inputs or outputs also help determine the role of a relation. If a relation involves inputs only, it will serve as a pre-processing filter or an input space partitioning. If it involves outputs, the relation serves as an infinite constraint or a post-processor.

3.2.6 Role Assignment

Enumeration of the types of prior knowledge allows us to systematically address the role of each type in the model synthesis problem. The relations and their roles in model synthesis are summarized in Table 3.1. The role of the relation depends upon (1) whether the relation is an equality or inequality, (2) the extent to which the functional structure of the relation is known, (3) its precedence with respect to the data, and (4) whether it involves only inputs or includes outputs.

Table 3.1: Role of prior knowledge.

<table>
<tr><th>1. Relation Type</th><th>2. Functional Form</th><th>3. Precedence w.r.t. Data</th><th>4. Variables Involved</th><th colspan="2">Role</th></tr>
<tr><td rowspan="5">Equality</td><td rowspan="3">Known</td><td rowspan="2">Greater</td><td>Inputs only</td><td colspan="2">1. parametric preprocessor</td></tr>
<tr><td>Any outputs</td><td colspan="2">2. parametric output model</td></tr>
<tr><td>Lower</td><td>Any</td><td>3a. default model, if domain is region without data</td><td>3b. penalty function</td></tr>
<tr><td rowspan="2">Unknown</td><td>Greater</td><td>Any</td><td colspan="2">4. nonparametric submodel</td></tr>
<tr><td>Lower</td><td>Any</td><td colspan="2">provides no information</td></tr>
<tr><td rowspan="5">Inequality</td><td rowspan="3">Known</td><td rowspan="2">Greater</td><td>Inputs only</td><td colspan="2">5. input space partitioning</td></tr>
<tr><td>Any outputs</td><td>6a. infinite constraint</td><td>6b. parametric output model, if able to transform</td></tr>
<tr><td>Lower</td><td>Any</td><td colspan="2">7. penalty function</td></tr>
<tr><td rowspan="2">Unknown</td><td>Greater</td><td>Any</td><td colspan="2">provides no information</td></tr>
<tr><td>Lower</td><td>Any</td><td colspan="2">provides no information</td></tr>
</table>

Equality constraints with known functional form are parametric models that reduce degrees of freedom. An equality with precedence greater than the data is enforced either as an input preprocessor (*Role 1*) or an output model (*Role 2*) depending upon whether inputs or outputs are involved, respectively. If the equality has precedence lower than the data, the domain of applicability defines the role. In domains beyond the data, the equality serves as a default model that provides a baseline estimate of the outputs (*Role 3a*). This estimate is refined by nonparametric submodels and controls extrapolation in the absence of data. Within the range of the data, an equality with lower precedence than the data serves as a functional preference that is imposed by a penalty function during training (*Role 3b*). Equalities with unknown form and a greater precedence than the data are the minimum knowledge necessary to propose a model. As such, they are modeled using a purely nonparametric network (*Role 4*).

Inequality constraints serve as modifiers to the training objective function or appear as constraints during training. Inequalities only impart meaning when their functional form is known. When the inequality is a hard constraint, such as non-negativity of a mole fraction, it must be strictly adhered to. Hard inequalities involving only inputs will partition the input space into domains of applicability (*Role 5*). This serves as the basis for input data rectification or merely as a screening filter that prevents invalid inputs to the model. On the other hand, if the outputs are involved in a hard inequality, the relation is an infinite constraint (*Role 6a*). Whenever possible, infinite constraints should be imposed by applying a suitable post-processing transformation (*Role 6b*), as is done with link functions for generalized linear and generalized additive models (McCullagh and Nelder, 1989; Hastie and Tibshirani, 1990). For example, if the output variable y must be positive, then the constraint $g(x;\mathbf{w}) = -y(x;\mathbf{w}) \leq 0$ must hold, and the hybrid could compute intermediate variable $z(x;\mathbf{w}) = \ln(y)$. The output model, then, would simply be $y(x;\mathbf{w}) = e^z$. If the inequality has a lower precedence than the data, the constraint represents merely a preference in model behavior. Such preferences can be injected into the estimation objective function as a penalty function (*Role 7*).

In sum, equality constraints define the model structure (*Roles 1, 2, 3a, 4, & 5*), and inequalities define the parameter estimation problem (*Roles 6a & 7*). The exceptions are when an equality with known form and precedence lower than the data exists, in which case it modifies the estimation problem (*Role 3b*), and when an infinite inequality can be transformed into a parametric output model (*Role 6b*). Knowledge of functional form dictates whether relations appear (if known) as parametric or (if unknown) nonparametric models. In this way the flexibility of nonparametric models may be exploited wherever uncertainty exists about the form of any relation. A model expressible as a known relation with an unknown contribution provides the basis for decomposing it into parametric and nonparametric modules. Precedence is derived from whether the constraint is considered hard or soft. If the constraint is hard, it has higher precedence than the data and serves as a consistency constraint. If the constraint is soft, the data have higher precedence and a relation with known form is represented as a default model or a behavioral preference depending upon whether the relation is an equality or inequality, respectively. Constraints with precedence lower than the data and unknown functional form provide no useful information. Involvement of inputs or outputs also help determine the role of a relation. A relation involving inputs only serves as a pre-processing filter or an input space partitioning. The roles of relations involving outputs are determined primarily by their type, form and precedence.

3.2.7 Example

We illustrate the role of prior knowledge through an example model synthesis problem (Table 3.2). The objective is to determine an equation of state for a non-ideal ternary gas mixture with components A, B, and C. Using experimental data, we wish to determine the relationship between volume V, pressure P, temperature T, the total moles n, and composition (mole fractions) $\mathbf{X} = [X_A, X_B, X_C]^T$. Initially, the only knowledge that exists is that the output variable y = V is dependent upon inputs $\mathbf{x} = [P, T, N, X_A, X_B,$

$X_C]^T$. Table 3.2 shows the construction of the full hybrid as knowledge is introduced in each of six synthesis steps.

Step 1: At this point, only a pure nonparametric network can be posed. Additional knowledge comes from first principles.

Step 2: First principles dictate that a normality constraint on the mole fractions be imposed. This constraint eliminates the need to independently gather data on one of the three mole fractions. Alternatively, it may serve as the rationale behind a preprocessing normalization (rectification) of the input data for all three mole fractions.

Step 3: The non-negativity constraint on all inputs also arises from first principles. It is applied as a preprocessing transformation or as an input filter.

Step 4: First principles dictate that the output V must be positive. This is an infinite constraint because it must hold for all input combinations. As such, it can be transformed into an output model $f_{out}(.)$ or applied directly using semi-infinite programming. The output model will be $f_{out}(z) = e^z$, where intermediate variable $z(x;\mathbf{w}) = \ln(y)$ is learned by the nonparametric network.

Step 5: We suspect that the gas mixture behavior can be described by a known equation of state, but that significant differences exist between reality and the known model. Also, the mixture has near ideal gas behavior at low P, though we do not have data in this region. This knowledge is the basis for subdividing the hybrid model $f_{hyb}(\mathbf{x})$ into a known equation of state, which is the parametric component $z_{def}(x)$, and an unknown nonparametric

component $f_{unk}(x)$. The ideal gas law $z_{def}(x) = V_{def} = nRT/P$ will serve as the default model.

Step 6: The deviation from ideality at higher pressures must be compensated for, but the functional form is unknown, i.e. $f_{unk}(\mathbf{x})$. A nonparametric model $z_{net}(\mathbf{x};\mathbf{w})$, with parameters $\mathbf{w}$, will approximate this unknown function.

The resulting hybrid equation of state $y = f_{out}(z(\mathbf{x};\mathbf{w}))$ will maintain consistent predictions when extrapolating into the low-pressure regime, where training data were unavailable.

Table 3.2. Prior Knowledge in the Ternary Gas Mixture Example.

Step	Relation	Attributes		Role
1.	relation we wish to model: $V = f(P,T,n,X)$	Type: Form: Precedence: Variables:	equality unknown greater inputs & output	nonparametric network
2.	mole fractions sum to one: $\Sigma X_i = 1$	Type: Form: Precedence: Variables:	equality known greater inputs only	parametric preprocessor
3.	non-negative inputs: $\mathbf{x} \leq \mathbf{0}$	Type: Form: Precedence: Variables:	inequality known greater inputs only	parametric preprocessor
4.	non-negative output: $y(\mathbf{x}) \leq 0$, for all $\mathbf{x}$	Type: Form: Precedence: Variables:	infinite inequality known greater inputs only	parametric output model [e.g., $z(x;\mathbf{w}) = \ln(y)$ $f_{out}(z) = e^z$]
5.	default behavior at low P: $z_{def}(\mathbf{x}) = nRT/P$	Type: Form: Precedence: Variables:	equality known lower inputs & output	parametric default model
6.	deviation from ideality: $z(\mathbf{x};\mathbf{w}) = z_{def}(\mathbf{x}) + f_{unk}(\mathbf{x})$ $y = f_{out}(z(\mathbf{x};\mathbf{w}))$	Type: Form: Precedence: Variables:	equality unknown lower inputs & output	nonparametric subnetwork

3.3 Nonparametric Models

As described in Section 2.2.1, nonparametric models are flexible functions whose form is derived from the data they are calibrated (or trained) with. Therefore, they are a good choice for representing relations among variables for which we have no prior knowledge of the functional form. As typical examples of nonparametric models with universal function approximators, artificial neural networks will be used as the nonparametric components in the modeling methodology we propose. Below, two types of neural networks, radial basis functions and back-propagation networks, are presented.

3.3.1 Radial Basis Function Networks

The neural network in the hybrid architecture is responsible for learning the difference between the default model and the target data. Although the neural network is a nonparametric estimator capable of approximating this difference, we also require that the neural network give a negligible contribution to the model output for inputs far from the training data. This property results in the hybrid having the correct default behavior in regions without training data. The transition from the data-regressed behavior to the default behavior must occur smoothly. A radial basis function network (RBFN) has this desired property.

Figure 3.2 depicts the RBFN architecture. Mathematically, an RBFN with K nodes and m outputs is represented as follows:

$$\mathbf{f}(\mathbf{x};\mathbf{W}) = \sum_{j=1}^{K} \mathbf{w}_j \, a_j(\mathbf{x}) + \mathbf{w}_{K+1} = \mathbf{W}\mathbf{a}(\mathbf{x}) \qquad (3.1)$$

where $\mathbf{W}$ is the $(K+1)\times m$ matrix of second-layer weights; $\mathbf{w}_j$ is the j^{th} row of $\mathbf{W}$; and $a_j(\mathbf{x})$ is the activation function of the j^{th} node, $a_j(\mathbf{x}) = \exp\left(-\left(\mathbf{x}-\mathbf{m}_j\right)^T \mathbf{S}_j \left(\mathbf{x}-\mathbf{m}_j\right)\right)$. It is constructed by first using K-means clustering on the $\mathbf{x}$-data alone (Moody and Darken, 1989) to position the nodes at centers $\mathbf{m}_j$ in input-space ($\mathbf{x}$-space). Then the size or breadth of activation is determined to set the shape matrices $\mathbf{S}_j$. Details of the procedure

are given by Johnston and Kramer (1993). Locally-tuned elliptical units (i.e., non-diagonal shape matrices) account for differences in data density in different dimensions. After positioning and sizing the nodes, the second-layer weights, which appear linearly in the calculation of the network output, must be determined. These parameters are estimated by solving the appropriate optimization problem as dictated by the available prior knowledge.

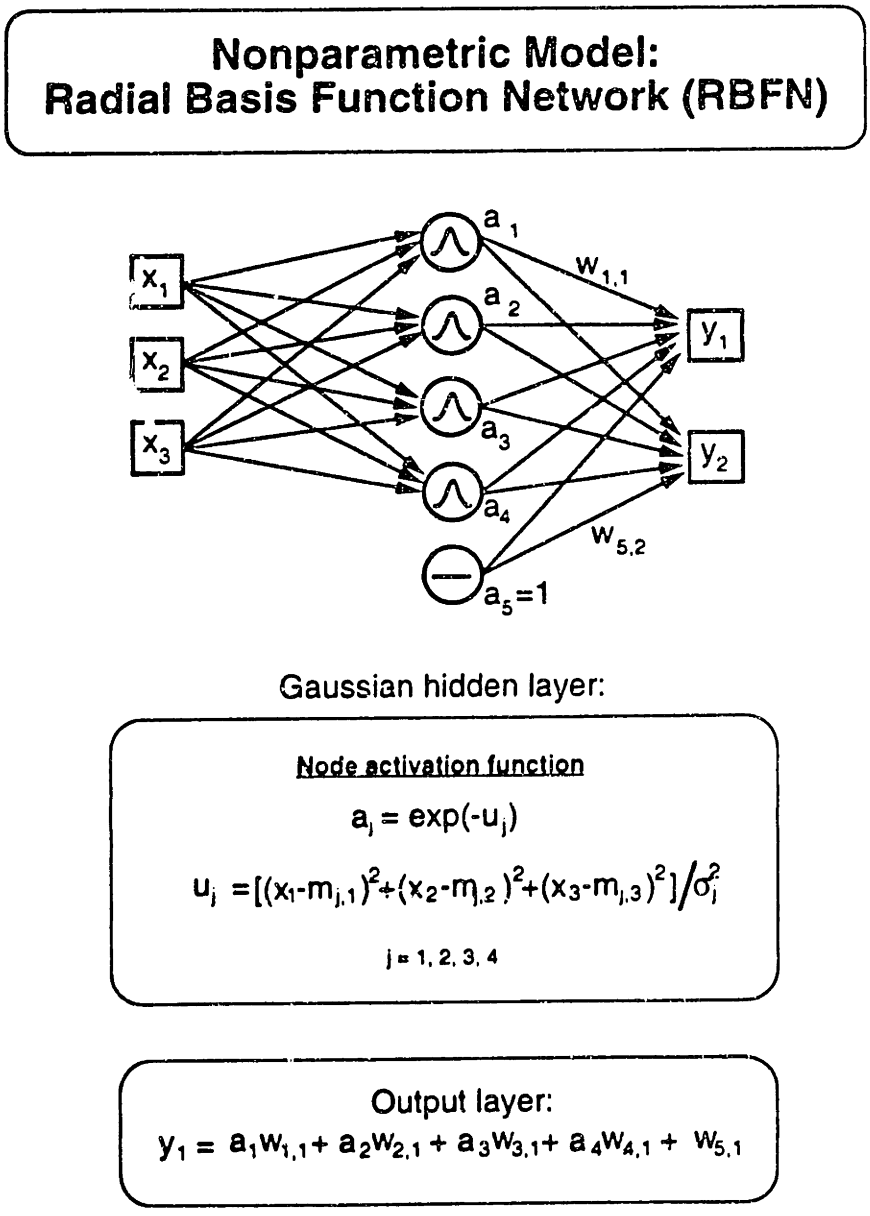


Figure 3.2: RBFN architecture

3.3.2 Back-Propagation Neural Networks

An alternative to a network with local support like the RBFN, is the back propagation neural network. A three-layer back-propagation network (3-BPN) is an artificial neural network with three layers: a linear input layer, a sigmoidal hidden layer, and a linear output layer. As shown in Figure 3.3, the input layer nodes are each connected to every one of the n inputs in **x**. Conversely, the nodes of the input and hidden layer are connected in pairs. Lastly, every node in the hidden layer has a connection to the single output node.

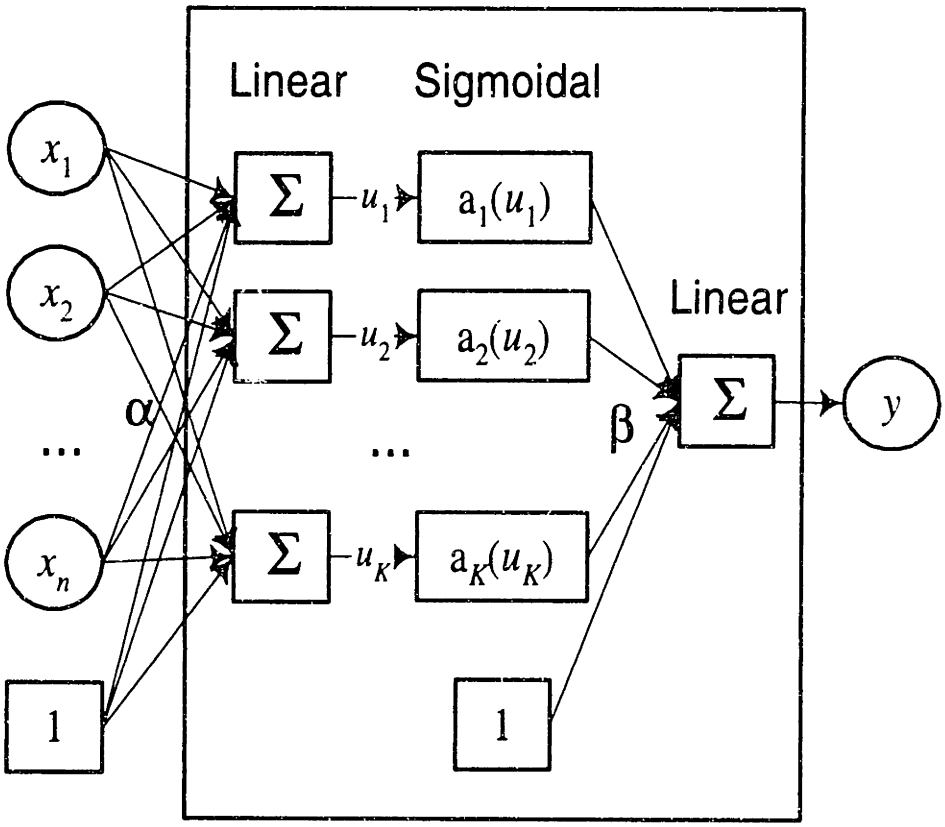


Figure 3.3: Architecture of a 3-layer back-propagation neural network.

The family of 3-BPNs is indexed by K, the number of node pairs in the input and hidden layers. Each of the K input nodes combines the n elements of **x** and a constant bias value into a weighted sum:

$$u_j = \sum_{i=1}^{n} \alpha_{ij} x_i + \alpha_{(n+1),j}, \quad j = 1,\ldots,K . \tag{3.2}$$

Thus. the network has K (n+1) input-layer weights, which we collect in the vector α. Each of the K hidden nodes accepts the value u_j from its corresponding node in the input layer and inputs it into a sigmoidal activation function $a_j(u_j)$:

$$a_j(u_j) = \frac{2}{\left(1+e^{-u_j}\right)} - 1, \quad j = 1,\ldots,K. \tag{3.3}$$

Lastly, the output layer is a single node that combines the hidden-layer activations into a weighted sum with a constant bias:

$$f(\mathbf{x};\mathbf{w}_K) = \sum_{j=1}^{K} \beta_j \, a_j\left(u_j(\mathbf{x})\right) + \beta_{K+1}. \tag{3.4}$$

The vector of model predictions $\mathbf{f}(\mathbf{X};\mathbf{w})$ for the 3-BPN is given in matrix notation by noting that the 3-BPN is a separable nonlinear model: it has parameters $\mathbf{w}$ that are separable into two disjoint sets α and β, and it is nonlinear in parameters α and linear in parameters β. The following equation, typical of a separable nonlinear model, results:

$$\mathbf{f}(\mathbf{X};\mathbf{w}) = \mathbf{A}(\mathbf{X};\alpha)\,\beta, \tag{3.5}$$

where $\mathbf{A}(\mathbf{X};\alpha)$ = the $N\times(K+1)$ matrix of node activations augmented with a column of ones; $\mathbf{w} = [\alpha^T\ \beta^T]^T$ is the vector of model parameters, such that α is the $[K(n+1)]\times 1$ vector of input weights, and β is the $(K+1)\times 1$ vector of output weights. The total number of parameters in the model is $p = K(n+1) + (K+1) = K\,(n+2)+1$.

3.4 Solution Methods

Prior knowledge not only dictates the model structure but also specifies the form of the optimization problem used in estimating the second-layer weights of the RBFN. The parameters of the default and output models are assumed to be known, and only the RBFN weights must be estimated. In general, the minimization problem for parameter estimation is an SIP problem:

$$\text{Minimize w.r.t. } \mathbf{w}: \quad F(\mathbf{w}) = \tfrac{1}{2}\sum_{i=1}^{N} \left\| \mathbf{y}_i - \mathbf{y}_{hyb}(\mathbf{x}_i;\mathbf{w}) \right\|^2 \tag{3.6}$$

$$\text{Subject to: } \mathbf{g}(\mathbf{w},\mathbf{x}) \le \mathbf{0};\ \ \mathbf{x} \in D_X \in \Re^n$$

where $\mathbf{w}$ are the RBFN weights; $\mathbf{y}_i$ is the m-dimensional vector of outputs for the i^{th} data point; $\mathbf{y}_{hyb} = \mathbf{f}_{hyb}(\mathbf{x}_i;\mathbf{w})$ is the hybrid model estimate of $\mathbf{y}$ given $\mathbf{x}_i$; $\mathbf{x}_i$ is the n-dimensional vector of inputs for the i^{th} data point; and $\mathbf{g}(\mathbf{w},\mathbf{x})$ are infinite inequality constraints.

Estimation becomes a constrained nonlinear programming problem if all inequalities are finite or the infinite inequalities can be transformed into finite constraints. Finite inequality constraints are independent of the inputs $\mathbf{x}$, and standard constrained optimization techniques are used to solve the associated Lagrangian, i.e. the penalized objective function:

$$\text{Minimize w.r.t. } \mathbf{w}, \boldsymbol{\lambda}: \; L(\mathbf{w}) = F(\mathbf{w}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{w}) \tag{3.7}$$

where $F(\mathbf{w})$ is the least-squares objective function, $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers, and $\mathbf{g}(\mathbf{w})$ are finite inequality constraints.

Parameter estimation for the hybrid model does not include infinite equality constraints because these constraints serve as the output model. When there are no infinite inequalities and the output model is nonlinear, the RBFN weights are estimated using conventional nonlinear programming (NLP) methods. However, if the output model $\mathbf{f}_{out}$ is linear in $\mathbf{x}$ and $\mathbf{y}$, the problem can be readily solved as a constrained linear regression. For example, mass balance constraints may dictate that $\mathbf{B}\,\mathbf{y} = \mathbf{C}\,\mathbf{x}$, where matrix $\mathbf{B}$ is $d \times m$ and $\mathbf{C}$ is $d \times n$. The m outputs may be partitioned into $m - d$ independent variables $\mathbf{y}_I$ and d dependent variables $\mathbf{y}_D$: $\mathbf{y} = [\mathbf{y}_I^T \mid \mathbf{y}_D^T]^T$. The $\mathbf{y}_D$ may be expressed in terms of $\mathbf{x}$ and $\mathbf{y}_I = \mathbf{z}(\mathbf{x}; \mathbf{w})$:

$$\mathbf{y}_D = \mathbf{B}_D^{-1} (\mathbf{C}\,\mathbf{x} - \mathbf{B}_I\,\mathbf{y}_I) \equiv \mathbf{Q}\,\mathbf{x} + \mathbf{P}\,\mathbf{z} \tag{3.8}$$

The $\mathbf{y}_I$ are estimated by summing the default model $\mathbf{z}_{def}$ and the RBFN $\mathbf{z}_{net} = \mathbf{W}\,\mathbf{a}(\mathbf{x})$, where $\mathbf{W}$ is the $(m-d) \times K$ matrix of output weights and $\mathbf{a}(\mathbf{x})$ is the vector of K node activations:

$$\mathbf{z}(\mathbf{x};\mathbf{w}) = \mathbf{z}_{def}(\mathbf{x}) + \mathbf{z}_{net}(\mathbf{x}; \mathbf{w}) = \mathbf{z}_{def}(\mathbf{x}) + \mathbf{W}\,\mathbf{a}(\mathbf{x}) \tag{3.9}$$

The resulting output model, $\mathbf{f}_{out}$, computes the outputs of the hybrid model:

$$\mathbf{y}_{hyb}(\mathbf{x}) = [\ \mathbf{y}_I{}^T \mid \mathbf{y}_D{}^T]^T_{hyb} = \mathbf{f}_{out}(\mathbf{z}(\mathbf{x};\mathbf{w}),\mathbf{x}) = [\ \mathbf{z}^T \mid (\mathbf{Q}\,\mathbf{x} + \mathbf{P}\,\mathbf{z})^T]^T \quad (3.10)$$

The output model is linear in $\mathbf{z}$ and therefore linear in the weights $\mathbf{W}$. Likewise, the parameter estimation objective function $F(\mathbf{W})$ (for a linear model with no inequality constraints) is quadratic in $\mathbf{z}$ and, thus, quadratic in $\mathbf{W}$. Therefore, a linear least-squares problem results that can be readily solved: Minimize, with respect to $\mathbf{W}$,

$$F(\mathbf{W}) = \tfrac{1}{2}\sum_{i=1}^{N}(\mathbf{y}_{Li} - \mathbf{z}_i)^T(\mathbf{y}_{Li} - \mathbf{z}_i) + \tfrac{1}{2}\sum_{i=1}^{N}(\mathbf{y}_{Di} - \mathbf{Q}\mathbf{x}_i - \mathbf{P}\mathbf{z}_i)^T(\mathbf{y}_{Di} - \mathbf{Q}\mathbf{x}_i - \mathbf{P}\mathbf{z}_i). \quad (3.11)$$

To summarize, if there are no inequality constraints the RBFN weights are estimated using a nonlinear programming (NLP) algorithm. If in addition the output model is linear, constrained linear regression may be used to determine the weights. When all inequalities present are finite, a constrained NLP algorithm is used; and, if any of the inequalities are infinite an SIP method is used.

3.5 Model Validation

Prior knowledge may dictate a specific structure, but when it does not, a modeler must choose from among many candidates the single model that is most suitable to the task at hand. Often suitability is measured in terms of prediction accuracy. We will assume that a set of candidate models $\mathcal{M}$ exists, from which we wish to select the best model. Thus, model selection becomes choosing the best model $\mathbf{f}_K(\mathbf{x};\mathbf{w}_K)$, where K indexes the set $\mathcal{M}$, and $\mathbf{w}_K$ is a p-element vector of model parameters, i.e., $\mathbf{w}_K \in \Re^p$. In general, the models in $\mathcal{M}$ may have completely unrelated forms, but, in nonparametric modeling, the members of $\mathcal{M}$ belong to the same family, each having similar form. In these cases, the space of possible models is infinite, e.g. the family of K-order polynomials with $K=1,\ldots,\infty$. So to make the problem tractable, the search is performed over a set of candidate models M, a finite subset of $\mathcal{M}$.

In order to assess the suitability of each model, the optimal parameters $\mathbf{w}_K^*$ are first estimated by minimizing a training objective $F(\mathbf{w}_K;f_K,D)$, which is a function of the available data D. In our work, $D \equiv \{\mathbf{x}_k, y_k: \mathbf{x}_k \in \Re^n, y_k \in \Re;\ k = 1,\ldots,N\}$, where $\mathbf{x}$ and y are realizations of the random variables X and Y with joint probability density function $p(X,Y)$. Also, we restrict the observed dependent variable y to be scalar for the sake of clarity, but note that the results are readily extended to vector-valued y. An added assumption is that y is generated by the true model with zero-mean additive noise ε; i. e., $y = f(\mathbf{x}) + \varepsilon$. So, the model $f_K(\mathbf{x};\mathbf{w}_K^*)$ will approximate the expectation of the conditional probability density function and the prediction for y_k, denoted $\hat{y}_k = f_K(\mathbf{x}_k;\mathbf{w}_K^*)$, is an estimate of the expected value for y given $\mathbf{x}_k$. More formally, the model selection problem is stated as follows: Select the model $f_K(\mathbf{x};\mathbf{w}_K) \in M \subset \mathcal{M}$ that, given its optimal parameters $\mathbf{w}_K^*$, minimizes prediction error criterion $J(f_K;\mathbf{w}_K)$. This paper focuses upon choosing $J(f_K;\mathbf{w}_K)$, an estimate of prediction accuracy.

There are many choices for a measure of prediction accuracy. Härdle (1991) lists several squared error measures and shows that they are all asymptotically equivalent to the Mean Integrated Squared Error (MISE):

$$\mathrm{MISE}(h) = \mathrm{E}[\mathrm{ISE}(h)] = \mathrm{E}\left[\int \left[f(\mathbf{x}) - f_h(\mathbf{x};\mathbf{w}_h^*)\right]^2 p(\mathbf{x}) d\mathbf{x}\right], \tag{3.12}$$

where "E" denotes expectation taken over all possible data sets D of size N and $p(\mathbf{x})$ is the marginal distribution of $\mathbf{x}$.

As Härdle (1991) points out, a convenient measure that asymptotically converges to $\mathrm{MISE}(K)$ is the Averaged Squared Error (ASE):

$$\mathrm{ASE}(h) = \frac{1}{N}\sum_{k=1}^{N}\left[f(\mathbf{x}_k) - f_h(\mathbf{x}_k;\mathbf{w}_h^*)\right]^2. \tag{3.13}$$

Note that Eqn. (3.13) involves the true model values $f(\mathbf{x}_k)$ (without noise). The [illegible] unknown, so ASE cannot be calculated exactly. However, an analysis of ASE provides the motivation for the objective function $F(\mathbf{w}_K;f_K,D)$ used in parameter estimation. More

importantly for our purposes, it also motivates the derivations of the major candidates for $J(f_K;\mathbf{w}_K)$, which include leave-one-out cross validation (CV) (Stone, 1974) and complexity-penalized measures such as GPE (Moody, 1992), FPE (Akaike, 1974), AIC (Akaike, 1970), and GCV (Craven and Wahba, 1974).

A natural estimate of ASE is the common least-squares objective function for parameter estimation. It is calculated by simply substituting the observed values y_k for $f(\mathbf{x}_k)$ and is expressed in terms of the residuals $c_k \equiv y_k - f_K^*(\mathbf{x};\mathbf{w}_K)$:

$$F(\mathbf{w}_h^*;f_h,D) = \frac{1}{N}\sum_{k=1}^{N}\left[y_k - f_h\left(\mathbf{x}_k;\mathbf{w}_h^*\right)\right]^2 \equiv \frac{1}{N}\sum_{k=1}^{N}c_k^2 . \qquad (3.14)$$

Härdle (1991) shows that this estimate of ASE is biased by a negative term, thus yielding overfit models if used as a prediction accuracy measure. However, approaches to eliminate that bias yield methods for computing $J(f_K;\mathbf{w}_K)$ (Härdle, 1991):

Approach 1: Leave-one-out estimation: In this approach we replace $f(\mathbf{x}_k)$ in Eqn. (3.13) with the leave-one-out estimate $\hat{y}_{[k]} = f_{K[k]}$, which is the model prediction of y_k when the parameters of f_K are estimated while holding out the k^{th} data point $(\mathbf{x}_k,y_k)$. This drives the expectation of the bias term to zero.

Approach 2: Complexity-penalized criteria: In this approach we introduce an additional term to Eqn. (3.13) that corrects for the bias.

3.5.1 Cross Validation

Approach 1 is standard leave-one-out cross validation CV (Stone, 1974). CV is the sample mean of the leave-one-out squared errors $c_{[k]}^2$ (for notational conciseness, we have suppressed the "K" denoting the dependence of $\hat{y}_{[k]}$ and $c_{[k]}$ on the model f_K):

$$CV(h) = \frac{1}{N}\sum_{k=1}^{N}\left(y_k - \hat{y}_{[k]}\right)^2 \equiv \frac{1}{N}\sum_{k=1}^{N}c_{[k]}^2 . \qquad (3.15)$$

3.5.2 Generalized Prediction Error Criteria

Approach 2 yields penalized criteria of the following form (Härdle, 1991):

$$G(h) = \frac{1}{N}\sum_{k=1}^{N}\left(y_k - \hat{y}_k\right)^2 \Phi(h). \tag{3.16}$$

Common complexity-penalized criteria arise from applying different assumptions in deriving $\Phi(K)$ (e.g., Zhang, 1993a; Eubank, 1988). The penalty function $\Phi(K)$ depends on the ratio of model degrees of freedom $d(K)$ and data degrees of freedom (sample size) N. For a linear model, the model degrees of freedom equals the number of parameters: $d(K) = p_K$, assumed less than N. Thus, $\Phi(K) = \Phi(p_K/N)$. The following complexity-penalized criteria have been shown to be unbiased estimators of ASE (Härdle, 1991):

Generalized cross validation (Craven and Wahba, 1974):

$$\Phi(K) = 1/(1-p_K/N)^2. \tag{3.17}$$

Final prediction error (FPE) (Akaike, 1974):

$$\Phi(K) = (1+p_K/N)/(1-p_K/N). \tag{3.18}$$

Akaike's information criterion (AIC) (Akaike, 1970):

$$\Phi(K) = \exp(2p_K/N). \tag{3.19}$$

Moreover, for all of the above criteria, $\Phi(K)$ approaches one as p_K/N approaches zero.

3.5.2.1 Linear Models

It is illustrative to examine leaving-one-out and complexity-penalized approaches in the context of models linear in the parameters. Insights gained in doing so lead to extensions that provide estimates of prediction accuracy for nonlinear models. Note that we shall refer to models linear in the parameters as "linear models," but these models are usually nonlinear in the independent variables. Moreover, we choose as the parameter estimation objective function $F(\mathbf{w}_K; f_K, D)$, the least-squares objective (Eqn. 3.14). As a result we arrive at the well-known expression of the optimal linear regression model (e.g., Buja, et al., 1989; Chatterjee and Hadi, 1988):

$$\hat{\mathbf{y}} \equiv f_K\left(\mathbf{X}; \mathbf{w}_K^*(\mathbf{y})\right) = \mathbf{X}\,\mathbf{w}_K^*(\mathbf{y}) = \mathbf{H}\,\mathbf{y} \tag{3.20}$$

where $\hat{\mathbf{y}}$ is the $N\times1$ column vector of predicted values; $\mathbf{X}$ is the $N\times p$ design matrix of input values with k^{th} row = $\mathbf{x}_k^T$ and rank($\mathbf{X}$)=p; $\mathbf{y}$ is the $N\times1$ vector of observed values; $\mathbf{w}_K^*(\mathbf{y})$ is the $p\times1$ vector of optimal parameter estimates given by $\mathbf{w}_K^*(\mathbf{y}) = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$; and $\mathbf{H}$ is the $N\times N$ hat (or influence) matrix given by $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$.

For a linear model, CV can be calculated without re-estimating the parameters. This is done by exploiting Eqn. (3.20), which gives the linear dependence of the predictions on each data point. The result is an explicit equation relating the leave-one-out residual $c_{[k]}$ to the complete-data residual c_k.

We begin by defining the value z_k to be the "zero-influence" point, which is the point to which y_k would have to be moved so that it contributes a zero residual to the least-squares objective function, i.e. just as if it had been left out of the regression:

$$\mathrm{F}(y_k = z_k) = \sum_{j\neq k}^{N} c_j^2 + c_k^2 \quad ; \tag{3.21}$$

where $c_k = z_k - \hat{y}_k\,(y_k{=}z_k) = 0$. Therefore, $\mathrm{F}(y_k{=}z_k)$ equals $\mathrm{F}_{[k]}$, the leave-one-out objective function. Consequently, the optimal parameter values of the two objective functions and the predictions at the k^{th} point are identical, so z_k is the leave-one-out prediction for y_k:

$$\hat{y}_k\,(z_k) = z_k = \hat{y}_{[k]}. \tag{3.22}$$

For a linear regression model each prediction can be written as a function of the data:

$$\hat{y}_k(y) = \sum_{j=1}^{N} h_{kj} y_j . \tag{3.23}$$

So, if y_k is the only one of the N points that is moved, and it is moved to z_k, the following equation holds:

$$\hat{y}_k(z_k) = \sum_{j=1}^{N} h_{kj} y_j + h_{kk}\left(z_k - y_k\right) = \hat{y}_k(y_k) + h_{kk}\left(z_k - y_k\right). \tag{3.24}$$

Subtracting the original data point y_k from both sides and substituting the equality $\hat{y}_k\,(z_k){=}z_k$ yields the following relation:

$$z_k - y_k = \hat{y}_k\,(y_k) - y_k + h_{kk}(z_k - y_k). \tag{3.25}$$

Substituting the definitions of the residuals $c_k \equiv y_k - \hat{y}_k\,(y_k)$ and $c_{[k]} \equiv y_k - z_k$ and rearranging gives the desired result:

$$c_{[k]} = c_k/(1-h_{kk}), \text{ and} \tag{3.26}$$

$$\mathrm{CV} = \frac{1}{N}\sum_{k=1}^{N}\frac{c_k^2}{\left(1-h_{kk}\right)^2}. \tag{3.27}$$

Equation (3.27) illustrates that the hat matrix **H**, through its diagonal elements h_{kk}, plays a central role in estimating prediction error given the residuals c_k computed from parameter estimation. This is one of the properties that make **H** a useful tool for model diagnosis in linear regression (Chatterjee and Hadi, 1988). **H** is also important in the complexity-penalized criteria of Equations (3.17), (3.18) and (3.19). For linear models (of full rank design matrix), trace(**H**) equals p_K, the number of parameters in the model; and therefore, the average value of the diagonal elements h_{kk} is given by $h_{\mathrm{avg}} = p_K/N$, which is the key term in the penalty function $\Phi(K)$. Additionally, Craven and Wahba (1974) derived GCV after noting that CV is not invariant to rotations: i.e., different models would be selected if the same data were used but were subjected to different rotational transformations. However, a rotationally invariant metric results if, for all k, h_{kk} is replaced in Eqn. (3.17) with the average value h_{avg}. Thus, the rotationally invariant metric has $\Phi(K) = 1/(1-p_K/N)^2$, which is the penalty function of GCV.

3.5.2.2 Nonlinear Models

Models nonlinear in the parameters, which we refer to as nonlinear models, are more difficult to analyze than linear models. Consequently, how to select the best nonlinear model remains an open issue (see, for example, Zhang, 1993a; Sclove, 1993; Moody, 1992; Wada and Kawato, 1992). Below are some estimates of prediction accuracy that are commonly applied to nonlinear models. As expected, they are extensions of techniques used in estimating the predictive capability of linear models and can also be classified with respect to the approach used to estimate ASE.

Employing Approach 1, cross validation methods estimate prediction error by leaving out one or more data points and computing residuals. However, for a nonlinear model, the techniques that achieve this must do so through repeated estimation of the parameters on different subsets of the same sample, thus the name "resampling methods." The most popular are leave-one-out cross validation (CV) (Stone, 1974), variants such as S-fold cross validation (SCV) (Zhang, 1993b), and repeated learning and testing (RLT) methods (Burman, 1989). The formula for CV applied to a nonlinear model is the same as Eqn. (3.15). But, for a nonlinear model, $c_{[k]}$ cannot be expressed explicitly in terms of c_k because the model predictions are related nonlinearly to the data points. Therefore, the $c_{[k]}$ are computed by brute force: leaving out the k^{th} observed value y_k of the N data points, retraining the model on the remaining N-1 data points, and computing the error between y_k and the model prediction for the left out point $\hat{y}_{[k]}$, i.e., the k^{th} leave-one-out residual $c_{[k]}$ (Stone, 1974). This process is repeated for each of the N data points.

An obvious drawback of applying CV to a nonlinear model is its requirement to train the model N additional times after parameter estimation. An alternative that reduces this computational burden is S-fold cross validation (SCV). Though it is more efficient, SCV suffers from greater variability than CV (Davison and Hall, 1992; Eubank, 1988). SCV partitions the data into $S<N$ randomly chosen sets of N_s points, where subscript $s=1,...,S$; $\sum_{s=1}^{S} N_s = N$; and, for all s, integer N_s is as close to N/S as possible. The model is retrained only S times, each time holding out one of the sets. The SCV metric is computed as the mean of the prediction squared errors:

$$\mathrm{SCV} = \frac{1}{S}\sum_{s=1}^{S}\sum_{k_s=1}^{N_s}\frac{\left(y_{k_s} - \hat{y}_{[k_s]}\right)^2}{N_s} \tag{3.28}$$

where y_{ks} = the k^{th} data point in the s^{th} set, and $\hat{y}_{[k_s]}$ = the prediction of y_{ks} by the model trained on the N-N_s other data points.

Complexity-penalized approaches corresponding to Approach 2 attempt to avoid the computational cost of cross validation. The complexity-penalized methods estimate the effective number of parameters (i.e., model degrees of freedom) without re-estimating the parameters. O'Sullivan and Wahba (1985) have extended GCV to nonlinear models by deriving analogues to h_{avg} for nonlinear models. The approaches linearize $f(\mathbf{x};\mathbf{w})$ about the optimal parameter $\mathbf{w}^*$:

$$f(\mathbf{x};\mathbf{w}) \cong f(\mathbf{x};\mathbf{w}^*) + \nabla f(\mathbf{x};\mathbf{w}^*)\,(\mathbf{w} - \mathbf{w}^*) \tag{3.29}$$

where $\nabla f(\mathbf{x};\mathbf{w}^*)$ is the gradient of $f(\mathbf{x};\mathbf{w})$ evaluated at $\mathbf{w}^*$.

Therefore, letting matrix $\mathbf{B}$ represent $\nabla f(\mathbf{X};\mathbf{w}^*)$, which is the Jacobian of the column vector of model predictions $\mathbf{f}$ given optimal parameters $\mathbf{w}^*$, the least-squares solution for $\mathbf{w}$ yields a relation analogous to Eqn. (3.20):

$$\mathbf{w}(\mathbf{y}) \cong \left(\mathbf{B}^T\mathbf{B}\right)^{-1}\mathbf{B}^T\left[\mathbf{y} - f\left(\mathbf{w}^*;\mathbf{y}\right) + \mathbf{B}\mathbf{w}^*\right], \tag{3.30}$$

$$\hat{\mathbf{y}} \cong \hat{\mathbf{H}}\mathbf{y} \tag{3.31}$$

$$\hat{\mathbf{H}} \equiv \mathbf{B}\left(\mathbf{B}^T\mathbf{B}\right)^{-1}\mathbf{B}^T. \tag{3.32}$$

The matrix $\hat{\mathbf{H}}$ is an effective hat matrix for nonlinear models. It is the nonlinear model analogue to the hat matrix of Eqn. (3.20), which proved valuable in relating the leave-one-out residuals to the full residuals in Eqn. (3.26). The values η_{kk}, which are the diagonal elements of $\hat{\mathbf{H}}$, are analogous to the h_{kk} of Eqn. (3.27). The sum of the η_{kk}, which equals trace($\hat{\mathbf{H}}$), is an estimate of the model degrees of freedom d(K), which Moody (1992) refers to as the effective number of parameters p_{eff}. Therefore, a GCV criterion for nonlinear models has complexity penalty function $\Phi(K)$ analogous to (3.17). It is as follows:

$$\Phi(K) = 1/(1\text{-trace}(\hat{\mathbf{H}})/N)^2 = 1/(1-\eta_{avg})^2. \tag{3.33}$$

Another complexity-penalized metric for nonlinear models is the general prediction error (GPE) criterion proposed by Moody (1992). Moody derived an alternative $\hat{\mathbf{H}}$ that benefits from the second-order derivatives of the model with respect to the parameters. He defined GPE to be a function of $p_{\text{eff}} = \text{trace}(\hat{\mathbf{H}})$ and the noise variance σ^2. The requirement of an estimate of σ^2 is a slight disadvantage of GPE. Here, for the purpose of illustrating the similarity of GPE with other complexity-penalized metrics, we use the estimate of σ^2 suggested by Wahba (1990), which is $\sigma^2 = F(\mathbf{w}_K; f_K, D)/(1-p_{\text{eff}}/N)$, and write $\Phi(K)$ for GPE as follows:

$$\Phi(K) = (1+p_{\text{eff}}/N)/(1-p_{\text{eff}}/N). \tag{3.34}$$

Thus, for this choice of σ^2, Moody's GPE can be viewed as a nonlinear analogue to Akaike's FPE in Eqn. (3.19).

3.5.3 Fast Cross Validation

In what follows, we present a method that efficiently solves the model selection problem for nonlinear models. The method is based upon a sensitivity analysis of the optimal parameters to perturbations in the data. The analysis yields a more accurate formula for the effective hat matrix $\hat{\mathbf{H}}$ than Eqn. (3.32). Using this alternative $\hat{\mathbf{H}}$, we derive a cross validation method with best-case performance requiring only a single parameter estimation step. The method also yields an estimate of p_{eff}. The estimated p_{eff} is then used to calculate GCV for nonlinear models and to estimate σ^2.

The next section presents the sensitivity analysis. We derive a method for estimating the cross validation criterion using the sensitivity analysis results. As an example, we demonstrate how to select the optimal number of nodes in the hidden layer of a single-output three-layer back-propagation neural network. Following that, we describe the design of a Monte Carlo simulation study performed to assess the performance of the new criteria. Finally, this section concludes with a summary of our approach.

3.5.3.1 Sensitivity Analysis

In this section, we apply the theory of sensitivity analysis for nonlinear programming to determine how the optimal parameters $\mathbf{w}^*$ change with respect to perturbations in the data $\delta\mathbf{y}$. That is, we must determine the function $\mathbf{w}(\delta\mathbf{y})$, where $\mathbf{w}(\mathbf{0}) = \mathbf{w}^*$. Fiacco (1983) proves the existence of the function $\mathbf{w}(\delta\mathbf{y})$ in deriving his Sensitivity Analysis Theorem for general optimization problems. We used Fiacco's Theorem for unconstrained minimization of $F(\mathbf{w},\delta\mathbf{y})$ to derive the method we call "fast cross validation." Below, we present that theorem.

Before the Theorem can be presented, Fiacco's definitions of specific terms are necessary. The unconstrained minimization problem is posed as follows: Minimize $F(\mathbf{w},\delta\mathbf{y})$ with respect to $\mathbf{w}$. The first-order Karush-Kuhn-Tucker (KKT) condition for an optimal solution of this problem is $\nabla F(\mathbf{w},\delta\mathbf{y}) = 0$. A *strict local minimizing point* (SLMP) of $F(\mathbf{w},\delta\mathbf{y})$, say $\mathbf{w}^*$, is a point that has a neighborhood such that there does not exist any other $\mathbf{w}$ where $F(\mathbf{w},0) \leq F(\mathbf{w}^*,0)$. And, the second-order sufficient condition for an SLMP is given as follows: *If* $F(\mathbf{w},\delta\mathbf{y})$ is twice continuously differentiable in a neighborhood of $\mathbf{w}^*$, *then* $\mathbf{w}^*$ is an SLMP if (1) the first-order KKT condition holds and (2) $\mathbf{M}$, the Hessian of F, is positive definite (i.e., $\mathbf{z}^T\mathbf{M}\mathbf{z} > 0$, $\forall \mathbf{z} \neq 0$).

Now, we present Fiacco's Sensitivity Analysis Theorem for the unconstrained minimization of $F(\mathbf{w},\delta\mathbf{y})$:

> *If* (1) $F(\mathbf{w},\delta\mathbf{y})$ is twice continuously differentiable in $\mathbf{w}$, (2) $\nabla F(\mathbf{w},\delta\mathbf{y})$ is once continuously differentiable in $\delta\mathbf{y}$, in a neighborhood of $(\mathbf{w}^*,0)$, and (3) the second-order sufficient conditions for an unconstrained SLMP of $F(\mathbf{w},0)$ hold at $\mathbf{w}^*$, *then*, for $\delta\mathbf{y}$ near 0, there exists a unique once continuously differentiable function $\mathbf{w}(\delta\mathbf{y})$ satisfying the second-order sufficient conditions for an unconstrained SLMP of $F(\mathbf{w},\delta\mathbf{y})$ such that $\mathbf{w}(0)=\mathbf{w}^*$ and hence, $\mathbf{w}(\delta\mathbf{y})$ is a locally unique SLMP of $F(\mathbf{w},\delta\mathbf{y})$.

This means that the optimal parameters are a function of the data perturbations $\delta\mathbf{y}$, and as long as the perturbations are small, the parameters will change in a way that maintains the objective function at a minimum corresponding to the new data point $\mathbf{y}+\delta\mathbf{y}$.

Sensitivity analysis determines what value the perturbation δy_k (the k^{th} element of $\delta\mathbf{y}$) must assume so that the resulting model prediction will equal the leave-one-out prediction. While analyzing each point y_k, all other points y_j are unchanged (i.e., $\delta y_j=0\ \forall\ j\neq k$). So, to estimate $c_{[k]}$, we must find Δy_k. The value Δy_k is the change in y_k that drives the parameters to the necessary values $\mathbf{w}_{[k]}$ causing the model to compute the leave-one-out prediction $\hat{y}_{[k]}$. In other words, we wish to find Δy_k such that

$$\mathbf{w}_{[k]} = \mathbf{w}^* + \int_0^{\Delta y_k} \nabla\mathbf{w}(u)du \equiv \mathbf{w}^* + \Delta\mathbf{w}(\Delta y_k), \tag{3.35}$$

$$\hat{y}_{[k]} = f(\mathbf{x}_k;\mathbf{w}_{[k]}), \tag{3.36}$$

where $\nabla\mathbf{w}(\delta\mathbf{y})$ is the Jacobian of $\mathbf{w}(\delta\mathbf{y})$ with respect to $\delta\mathbf{y}$.

In general, $\mathbf{w}(\delta\mathbf{y})$ is a nonlinear algebraic function that makes solving for Δy_k in Eqn. (3.35) a computationally expensive task. However, a first-order sensitivity analysis of the optimal parameters $\mathbf{w}^*$ provides us with an estimate of $\Delta\mathbf{w}(\Delta y_k)$ that reduces the computation needed to estimate Δy_k. Noting that $\mathbf{w}(\mathbf{0}) = \mathbf{w}^*$, we can express $\mathbf{w}(\delta\mathbf{y})$ as a first-order Taylor series expansion in $\delta\mathbf{y}$ about $\delta\mathbf{y}=\mathbf{0}$ (i.e., a first-order Maclaurin series):

$$\mathbf{w}(\delta\mathbf{y}) \equiv \mathbf{w}^* + \nabla\mathbf{w}(\mathbf{0})\ \delta\mathbf{y}. \tag{3.37}$$

Under the assumptions of Fiacco's Theorem, $\nabla\mathbf{w}(\delta\mathbf{y})$ can be expressed in terms of the derivatives of the parameter estimation objective function $F(\mathbf{w},\delta\mathbf{y})$. We denote the gradient of $F(\mathbf{w},\delta\mathbf{y})$ with respect to $\mathbf{w}$ as $\nabla F(\mathbf{w},\delta\mathbf{y})$ and the Hessian with respect to $\mathbf{w}$ as $\mathbf{M}=\nabla^2 F(\mathbf{w},\delta\mathbf{y})$. Then, $\nabla\mathbf{w}(\delta\mathbf{y})$ is given by the following equation:

$$\nabla\mathbf{w}(\delta\mathbf{y}) = \mathbf{M}^{-1}(\delta\mathbf{y})\ \mathbf{N}(\delta\mathbf{y}) \tag{3.38}$$

where, $\mathbf{M}^{-1}(\delta\mathbf{y})$ is the inverse of $\mathbf{M}$, the Hessian of $F(\mathbf{w},\delta\mathbf{y})$, and $\mathbf{N}(\delta\mathbf{y})$ is the negative Jacobian of $\nabla F(\mathbf{w},\delta\mathbf{y})$ with respect to $\delta\mathbf{y}$.

For the least-squares problem, the vector of residuals $\mathbf{c}$, the objective function of Eqn. (3.14), and the gradient of the objective function can be written in terms of the perturbation vector $\delta\mathbf{y}$:

$$\mathbf{c}(\delta\mathbf{y}) \equiv \mathbf{y} + \delta\mathbf{y} - \mathbf{f}\left(\mathbf{X}; \mathbf{w}(\delta\mathbf{y})\right), \tag{3.39}$$

$$F(\delta\mathbf{y}) = \frac{1}{N}\mathbf{c}^{\mathsf{T}}\mathbf{c}, \tag{3.40}$$

$$\nabla F(\delta\mathbf{y}) = \frac{-2}{N}\nabla\mathbf{f}^{\mathsf{T}}\mathbf{c}. \tag{3.41}$$

Then, the matrices $\mathbf{M}(\delta\mathbf{y})$ and $\mathbf{N}(\delta\mathbf{y})$ are expressed as follows:

$$\mathbf{M}(\delta\mathbf{y}) = \frac{2}{N}\left[\nabla\mathbf{f}^{\mathsf{T}}\nabla\mathbf{f} - \mathbf{D}\left(\nabla\mathbf{f}^{\mathsf{T}}\right)\mathbf{c}\right] \tag{3.42}$$

$$\mathbf{N}(\delta\mathbf{y}) = \frac{2}{N}\nabla\mathbf{f}^{\mathsf{T}}, \tag{3.43}$$

where $\nabla\mathbf{f}$ is the $N\times p$ Jacobian with respect to $\mathbf{w}$ of the $N\times 1$ model vector $\mathbf{f}$, and $\mathbf{D}(\nabla\mathbf{f}^{\mathsf{T}})$ is the Fréchet derivative of $\nabla\mathbf{f}^{\mathsf{T}}$ with respect to $\mathbf{w}$. The Fréchet derivative of the $p\times N$ matrix $\nabla\mathbf{f}^{\mathsf{T}}$ is a $p\times p\times N$ tensor, in other words, N matrix slabs, each with dimension $p\times p$. The k^{th} matrix slab is the $p\times p$ Hessian of $\mathbf{f}$ with respect to $\mathbf{w}$ evaluated at the k^{th} data point $\mathbf{x}_k$ (see, e.g., Golub and Pereyra, 1973). In Eqn. (3.42), the term $\mathbf{D}(\nabla\mathbf{f}^{\mathsf{T}})\mathbf{c}$ is calculated by matrix multiplication of each of the p $p\times N$ cross-sections of the tensor by the $N\times 1$ vector $\mathbf{c}$. The result is a $p\times p$ matrix.

We can relate the leave-one-out residual $c_{[k]}$ to the leave-one-out parameters $\mathbf{w}_{[k]}$ by defining $\Delta\mathbf{y}_k$ as a column vector with $c_{[k]}$ as its k^{th} element and zeros elsewhere and by perturbing the data point y_k to the zero-influence point z_k, i.e., letting $\delta y_k = c_{[k]}$:

$$\delta y_k = c_{[k]} \tag{3.44}$$

$$\mathbf{w}_{[k]} \cong \mathbf{w}^* + \mathbf{M}^{-1}(0)\mathbf{N}(0)\,\Delta\mathbf{y}_k\,, \tag{3.45}$$

$$\hat{y}_k(\Delta\mathbf{y}_k) \cong \mathrm{f}(\mathbf{x}_k;\, \mathbf{w}^* + \mathbf{M}^{-1}(0)\mathbf{N}(0)\,\Delta\mathbf{y}_k\,). \tag{3.46}$$

An analysis of the first-order effects of the data perturbation on the model prediction yields an estimate of $\hat{y}_k(\Delta\mathbf{y}_k)$. The following equation results from linearizing the model with respect to the data perturbation, in other words, expanding the model as a first-order Maclaurin series in terms of $\delta\mathbf{y}$, and examining the k^{th} row of the resulting equation:

$$\hat{y}_k(\Delta\mathbf{y}_k) \;\cong\; \hat{y}_k \;-\; \eta_{kk}\, c_{[k]} \tag{3.47}$$

where, now we redefine

$$\hat{\mathbf{H}} \equiv \nabla\mathbf{f}\,\mathbf{M}^{-1}(0)\mathbf{N}(0), \tag{3.48}$$

and η_{kk} = the k^{th} element on the diagonal of $\hat{\mathbf{H}}$.

This alternative definition for $\hat{\mathbf{H}}$ arises from the first-order sensitivity analysis of the parameters and model predictions with respect to the data perturbations. It is a better choice for the effective hat matrix of a nonlinear model than Eqn. (3.32) because Eqn. (3.48) retains second-order information from the model in the form of $\mathbf{M}$, the Hessian of the objective function (Celmins, 1982; Moody, 1992).

Just as in deriving Eqn. (3.27), introducing $z_k = \hat{y}_k(\Delta\mathbf{y}_k)$ into Eqn. (3.47) relates the leave-one-out residual to the full residual:

$$c_{[k]} = c_k/(1-\eta_{kk}). \tag{3.49}$$

The assumptions of the Sensitivity Analysis Theorem are the qualifying assumptions of the derivation of Eqn. (3.49). These assumptions require that δy_k must be small enough to remain in the neighborhood of δy_k=0 so that $\mathbf{w}(\delta y_k)$ remains the local minimizing point of $\mathrm{F}(\mathbf{w},\delta\mathbf{y})$. Applying Eqn. (3.49) as a single-step estimate of the leave-one-out residual $c_{[k]}$ is equivalent to stating that $c_{[k]}$ equals the perturbation δy_k. Unfortunately, $c_{[k]}$ is not

always small. It may be quite large, in which case the assumptions are violated. We can detect this by checking if the value estimated for $\hat{y}_k(\Delta \mathbf{y}_k)$ by Eqn. (3.46) – the actual model at the perturbed weights – and that by Eqn. (3.47) – the first-order data perturbation approximation to the model – differ by more than some small tolerance. Additionally, recognizing that Eqn. (3.45) corresponds to a single step of a Newton's continuation method for estimating the optimal parameters of the model, when a violation of the assumptions is detected, we can calculate Δy_k by iterating on Eqn. (3.45), each time taking δy_k small enough that Eqn. (3.46) and Eqn. (3.47) agree. The sum of the δy_k over all iterations is Δy_k, and convergence occurs when Δy_k equals $f_K(\mathbf{x}_k; \mathbf{w}(\Delta y_k))$-$y_k$, which is the estimate of $c_{[k]}$.

A point y_k that has large $c_{[k]}$ requires iteration. In such a case, estimation of $c_{[k]}$ is approximately as expensive as re-estimating the model parameters while holding out y_k, as is done in computing CV by leave-one-out cross validation. However, if the majority of the points have small $c_{[k]}$, in which case iteration is not necessary, then this method will represent a significant savings in computation over leave-one-out cross validation. Moreover, in the best-case scenario, where all of the points have small $c_{[k]}$, the approach does not require re-estimation of the parameters. This potential savings in computation is the motivation for our proposed method, which is described in the next section.

3.5.3.2 The Approach

The promise of this approach is that we can estimate the leave-one-out residuals $c_{[k]}$ and the effective number of parameters p_{eff} without a full re-estimation of the parameters as is necessary in ordinary cross validation (CV). The prediction error of a nonlinear model may be estimated by substituting Eqn. (3.49) into the formula for CV, which is given by Eqn. (3.15). The result is the following nonlinear analogue to the linear model CV of Eqn. (3.27):

$$\text{FCV} = \frac{1}{N}\sum_{k=1}^{N}\frac{c_k^2}{(1-\eta_{kk})^2}. \tag{3.50}$$

This criterion is referred to as Fast Cross Validation (FCV) because it provides a means of estimating CV for a nonlinear model without necessarily re-estimating the model parameters. In fact, re-estimation of the parameters is completely unnecessary in the best-case scenario, which corresponds to $c_{[k]}$ computed by Eqn. (3.49) being small enough such that Eqns. (3.46) and (3.47) agree for all $k = 1,..., N$. However, often some of the $c_{[k]}$ will be large enough that they must be estimated iteratively. Below, we present an algorithm that treats these cases.

An advantage of the FCV approach to model selection is that the derivatives necessary for estimating prediction accuracy can be computed during parameter estimation using a gradient-based minimization technique (Fiacco, 1983). In parameter estimation, the data are unperturbed, so $\delta\mathbf{y}$ equals zero. Moreover, the objective function gradient $\nabla F(0)$ from Eqn. (3.41) is the negative product of the matrix $\mathbf{N}(0)$ from Eqn. (3.43) and the residual vector $\mathbf{c}(0)$ from Eqn. (3.39). Thus, the parameter update formula for the i^{th} iteration of Newton's method is a direct parallel to Eqn. (3.45):

$$\mathbf{w}^{(i)} = \mathbf{w}^{(i-1)} + \mathbf{M}^{-1}\mathbf{N}\,\mathbf{c}, \tag{3.51}$$

where $\mathbf{M}$, $\mathbf{N}$, and $\mathbf{c}$ are evaluated at $\mathbf{w}^{(i-1)}$, the parameters of the previous iteration.

However, two implementation issues are important to consider. First, because FCV requires the Hessian of the objective function, a Newton-type method is a good choice for the parameter estimation routine but it requires a good approximation to the Hessian. Second, the algorithm must be able to detect those exceptional cases when single-step estimation of $c_{[k]}$ is insufficient and iterative estimation is required.

The first issue can be addressed by applying Newton's method for parameter estimation. This requires computing an estimate of the Hessian, which, for some problems may be a computationally expensive task. However, for a specific model structure, it may be possible to derive good approximations to the Hessian that can be computed efficiently, as

is true, of some neural network architectures, for example (Buntine and Weigend, 1991; Pearlmutter 1993).

The second implementation issue concerns handling exceptions requiring iterative estimation of $c_{[k]}$. There are two such exceptions for a given data point y_k: (*i*) when the single-step estimation of $c_{[k]}$ given by Eqn. (3.49) is too large for Eqns. (3.46) and (3.47) to agree; and (*ii*) when $\eta_{kk}<0$ or $\eta_{kk}\geq 1$. Both types of exceptions are easily detected: the first by checking if Eqns. (3.46) and (3.47) are within some tolerance of each other, and the second by direct comparison of η_{kk} with the bounds zero and one.

The limitations of a first-order sensitivity analysis and numerical approximations to the Hessian contribute to the occurrence of both exceptions. However, the second type of exception may also arise due to the fact that the η_{kk} are dependent on the data observations, which are random variables. Thus, the η_{kk} themselves are random variables that may not be strictly bounded by zero and one. See, for example, the analysis of h_{kk} of a linear model when the $\mathbf{x}_k$ have errors (Chaterjee, S. and Hadi, A. S., 1988). Moreover, a large variance σ^2 in the noise ε and the sparseness of the data, which is dependent on the probability density function $p(X,Y)$ and sample size N, contribute to large $c_{[k]}$ and η_{kk} values approaching one.

The FCV methodology is summarized by the flowchart in Figure 3.4. It accounts for the contingency of iteratively estimating $c_{[k]}$. In the event that either exception is detected for a point y_k, the method performs iterative estimation for that point. So any gains in speed over leave-one-out cross validation must be due to having to re-estimate the parameters for less than the total N data points.

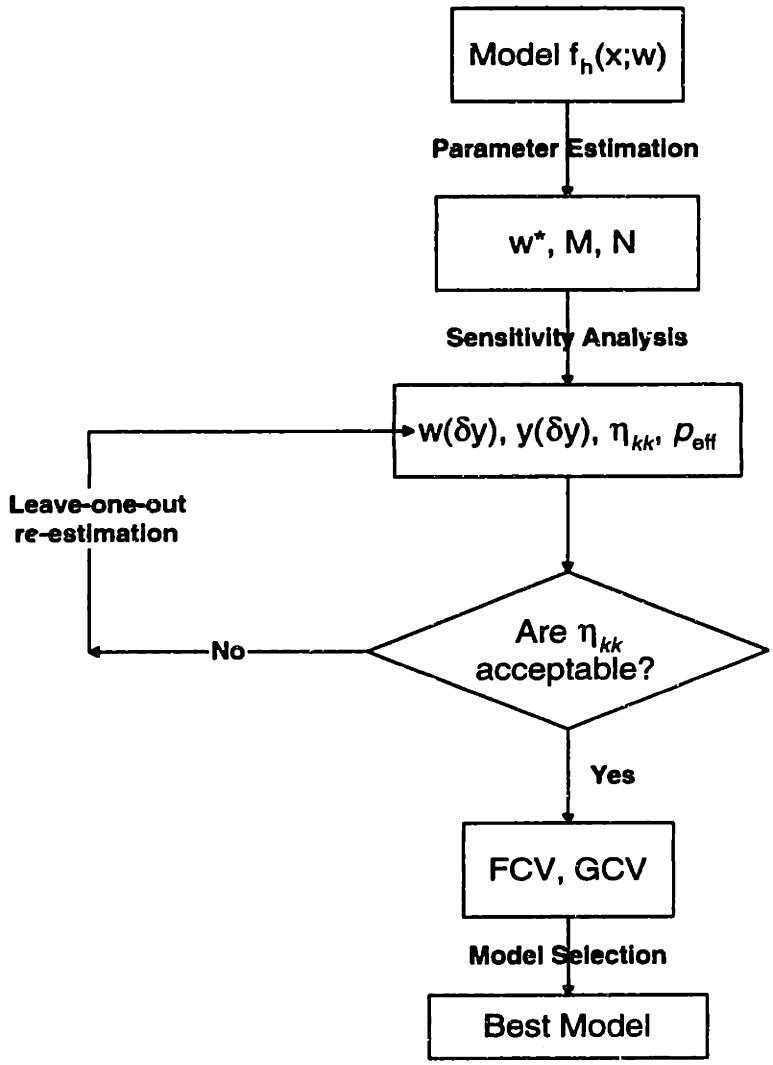


Figure 3.4: Flowchart of the FCV algorithm.

3.6 Implementation

The proposed model synthesis methodology was implemented on computer as a suite of Matlab functions, which appear in Appendix C as the "Function-Oriented Modeling Toolkit." The primary objective of this research was to develop a formalized methodology rather than a commercial-quality modeling environment, so the toolkit represents a loose collection of routines that must be coordinated by the user when applied. To assist in achieving this, the scripts that were used to execute the case studies of Section 3.7 are also included in Appendix C. Modelers can use these scripts as examples for structuring their own studies to take advantage of the routines.

3.7 Case Studies

We applied the function-oriented methodology described above in case studies of steady-state and dynamic processes. Also, a Monte Carlo simulation study of the Fast Cross Validation (FCV) method for estimating predictive capability of the models was performed. The next three sections describe the empirical studies and present their results.

3.7.1 Vinyl Acetate Polymerization

The application of the hybrid network is illustrated by modeling the polymerization of vinyl acetate in a continuous stirred tank reactor (CSTR). Conversion and molecular weight distribution in the CSTR is dependent upon the degree of mixing. In real polymerization systems, flow segregation (lack of perfect mixing at the molecular level) causes conversions and molecular weight distributions to deviate from those predicted for a perfectly mixed flow reactor (PMFR). However, as a first approximation, the segregated flow reactor may be modeled as a PMFR. The PMFR model for vinyl acetate is a set of nonlinear algebraic equations relating the mean residence time and composition to the product concentration and molecular weights (Hyun, *et al.*, 1976). At low monomer conversions, the PMFR model accurately predicts the number- and weight-average molecular weights of the polymer product; but, as conversion increases, the

number-average molecular weight is overestimated. In the hybrid model, the PMFR is used as the default model, and the RBFN is used to compensate for the discrepancies at high conversion.

A segregated flow model was chosen to represent the actual polymerization process (Hyun, *et al.*, 1976). It consists of nonlinear differential equations that must be solved numerically in order to estimate the moments of the polymer molecular weight distribution. The process was simulated assuming steady-state, isothermal operation of the reactor. Inputs to the process model were the mean residence time, x_1; the inlet solvent concentration, x_2; and the inlet initiator concentration, x_3. The measured outputs were the monomer conversion, y_1; the number-average molecular weight, y_2; the weight-average molecular weight, y_3; and the product concentration, y_4. The moles of polymer produced are related to the moles of monomer reacted and the average length of a polymer molecule. This relation supplies an equality constraint on the process outputs that must be satisfied for all future inputs: $h(\mathbf{y}) = y_2 y_4 - M_{mer} y_1 = 0$, where M_{mer} is the molecular weight of the vinyl acetate monomer. Therefore, $\mathbf{y}$ can be partitioned into $\mathbf{z} = \{y_1, y_2, y_3\}$ and $d = y_4$. The output function $\mathbf{y} = G(\mathbf{z})$ is expressed as: $G(\mathbf{z}) = \{z_1, z_2, z_3, M_{mer} z_1/z_2\}$. The default PMFR model is used to predict y_1, y_2, and y_3 only.

Training data for the example were generated using uniform sampling from the following ranges for x_1, x_2 and x_3: [1.6, 6.4], [3.5, 6.5] and [4.1, 4.9] $\times 10^{-4}$. Gaussian white noise, $N(0,(0.05)^2)$, was added to the outputs to simulate measurement errors. To illustrate the extrapolation properties of the hybrid model, the randomly generated training examples were restricted to inputs that yielded conversions between 0.25 and 0.45. A total of 60 observations were used for training. All were mean-centered and unit variance scaled, N(0,1). The hybrid network was trained by nonlinear optimization using the BFGS quasi-Newton method. The convergence of the optimization was accelerated by initializing the decision variables using least-squares regression of $\mathbf{z} - \mathbf{F}$ against $\mathbf{a}$.

For purposes of comparison, the same data were used to train a BPN and a RBFN. The architectures for all networks were optimized by selecting the minimum prediction error network as indicated by cross validation (CV). The conjugate gradient algorithm (Leonard and Kramer, 1990) was used to determine the weights of the BPN. The optimal BPN contained one hidden layer with 10 hidden sigmoidal nodes; while the optimal RBFN had 10 hidden radial units with overlap factor of 6 (used in setting the radial unit widths). The hybrid network architecture was selected by CV on the data set: {**x**, **z** - **F**}. The optimal architecture for the hybrid network was determined to be 10 hidden nodes with an overlap factor of 4.

Results comparing the prediction accuracy of the hybrid model, the BPN and the RBFN are depicted in Fig. 3.5. Test cases for this plot were generated by fixing the solvent and initiator concentrations at their mean values, and varying residence time. The ordinate shows the mean fractional error over all outputs, relative to the actual (noise-free) output from the segregated flow model. The hybrid model accurately estimates the true process values throughout the range of residence times. For conversions below 0.25, the lower limit of the training data, the hybrid model is still accurate because its output approaches that of the default model. The BPN and RBFN, on the other hand, predict accurately for points between conversions of 0.25 and 0.45, within the range of the training data, but deviate substantially from the actual outputs for conversions outside this range. This illustrates the unreliable extrapolation obtained using purely nonparametric models. For values of conversion above 0.45, the error in hybrid model begins to increase due to the poor accuracy of the default model in the region of high conversion.

Fig. 3.6 compares the degree of constraint satisfaction for the hybrid model, the BPN and the RBFN. The outputs are inconsistent if they do not satisfy the constraint $y_2 y_4/(M_{mer}\, y_1) = 1$, shown on the ordinate. The hybrid model satisfies the constraint exactly for all values of the inputs, while the BPN and the RBFN are unable to satisfy the constraint. This is because the constraint is not explicitly enforced during training and the training data, which contain substantial noise, do not accurately imply the constraint.

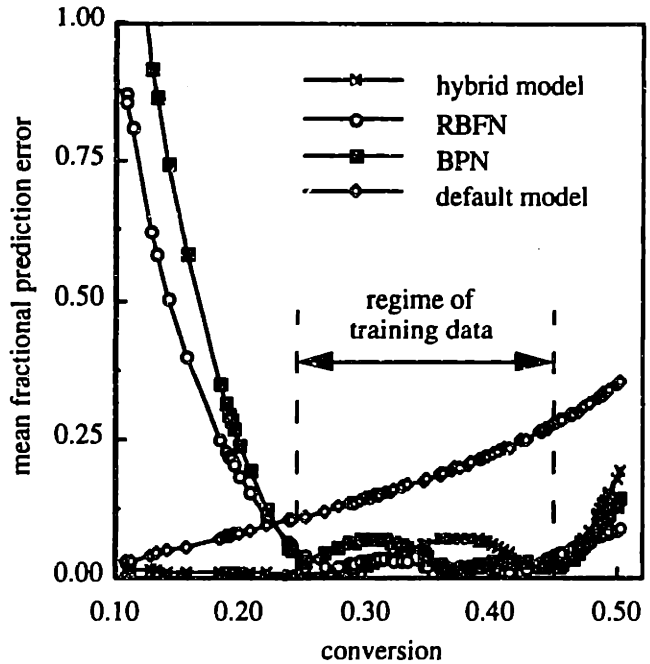


Figure 3.5: Prediction accuracy for hybrid model, radial basis function network, backpropagation network, and default model.

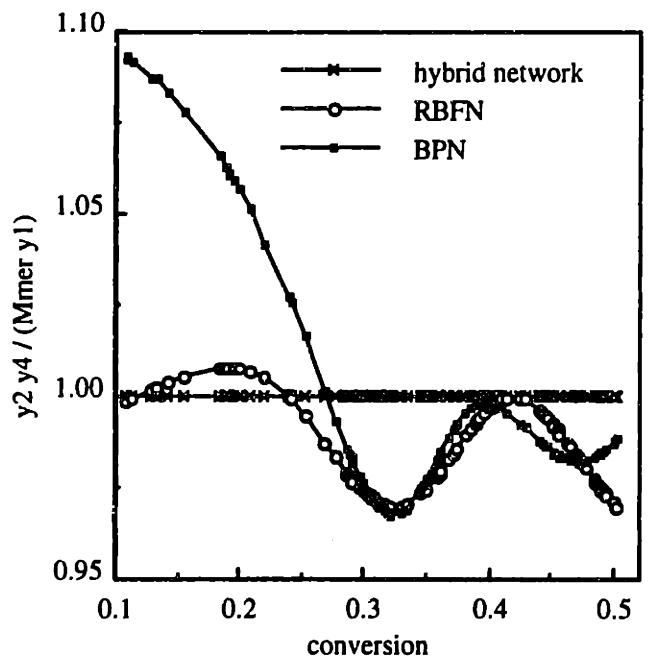


Figure 3.6: Constraint satisfaction for hybrid network, RBFN and BPN.

3.7.2 Fed-Batch Penicillin Fermentation

The second case studied was a fed-batch penicillin fermentation. This process is dynamic and illustrates how well the methodology produces accurate and general models suitable for process control applications. Thompson (1984) has demonstrated that the complex fermentation mechanism described by a nonlinear system of differential equations can be approximated by a system of nonlinear algebraic equations. This approximate algebraic system is ideal for use as a prior parametric model for the hybrid. This approach can be directly compared to applications of neural networks to fed-batch penicillin fermentation by Di Massimo *et al.* (1992) and Psichogios and Ungar (1991).

3.7.2.1 Process Description

The hybrid modeling methodology was applied in a case study of a fed-batch penicillin fermentation. In this dynamic biochemical process, a fermentor is charged with a growth medium containing the penicillin microorganism in low concentration. As the fermentation progresses, substrate (i.e., a sugar such as glucose) is fed into the fermentor in order to control the biomass growth rate at a high level. Once sufficient biomass has accumulated, the substrate feed rate is reduced to mere life-sustaining levels. At this point, the culture begins to produce the penicillin product in earnest. The fermentor analyzed in this study is characterized as follows:

- biomass concentration $X(t)$, penicillin concentration $P(t)$, and substrate concentration $S(t)$ define the process state at time t;
- specific growth rate μ, specific product formation rate q_p, and specific substrate consumption rate σ serve as time-varying parameters relating the state variables to their time-derivatives;
- substrate concentration in the feed S_f and the dilution rate D (defined as feed rate F divided by culture volume V) drive the process.

3.7.2.2 Mechanistic Model of True Process

Due to the complexity of a biological process, the true process exhibits phenomena that are not easily modeled or are unknown. For illustration purposes the phenomena characterizing the actual process are listed below but were assumed to be parametrically unmodeled:

- cell biomass growth rate is limited by substrate availability,
- growth rate decreases with biomass due to oxygen diffusion limitations,
- cell lysis becomes appreciable at high biomass concentration,
- maintenance energy of the culture (a component of the specific substrate consumption) is dependent upon biomass concentration, and
- penicillin product decays as the culture ages.

The balance equations on cell biomass, residual substrate in the fermentor broth, product and overall mass are the governing equations of the fermentation (Thompson, 1984). These equations were solved numerically to generate the simulated training and test data.

Cell biomass balance: $dX/dt = X(\mu - D - c_L)$

Substrate balance: $dS/dt = -\sigma X + (S_f - S)D$

Product balance: $dP/dt = q_p X - P(D + K)$

Overall mass balance: $dV/dt = F$.

The correlations and the parameters for the true model appear below. They are an adaptation of the Bajpai-Reuss model used by Thompson (1984):

Specific growth rate: $\mu = \mu_m S/(K_x X + 10)$

Specific cell lysis rate: $c_L = c_{Lm} X \exp(-S/100)/(K_L + X + 1)$

Specific substrate consumption rate: $\sigma = (\mu/Y_{x/s} + q_p/Y_{p/s} + m_x)$

Specific product formation rate: $q_p = 1.5 q_{pm} SX/[\, 4K_p + XS\,(1 + S/(3K_I))]$

Maintenance energy: $m_x = m_{xm} X/(X + 10)$.

The parameters had the following values: μ_m= 0.11 hr^{-1}; K_x= 0.3; c_{Lm}= 0.0084 hr^{-1}; K_L= 0.05; q_{pm}= 0.004 hr^{-1}; K_p= 0.0001 g/l; K_I= 1.0 g/l; K= 0.01 hr^{-1}; $Y_{x/s}$= 0.47; $Y_{p/s}$= 1.2; and m_{xm}= 0.029 hr^{-1}.

3.7.2.3 Default Model Based on Prior Knowledge

It is assumed that prior knowledge exists derived from either simple experiments on the current fermentor using the current strain of the penicillin microorganism, earlier experiments in lab-scale equipment, or possibly earlier experiments with a less productive strain. This prior knowledge is represented by a constant-specific growth rate model, which served as the default model (Thompson, 1984). The default model differs from the unknown actual process phenomena in several ways:

- specific growth rate is independent of biomass concentration;
- maximum specific growth rate is higher;
- cell lysis is negligible;
- greater inhibition of product formation by substrate;
- lower yield of product on substrate; and
- lower maintenance energy.

Taken together, these phenomena result in a default model that predicts higher biomass growth rate and lower product formation rate than the actual process. The equations of the default model are as follows:

Specific growth rate: $\mu = (1.2\mu_m)\, S/(5K_x + S)$

Specific substrate consumption rate: $\sigma = \mu/(1.1Y_{x/s}) + q_p/(0.9Y_{p/s}) + 0.9m_{xm}$

Specific product formation rate: $q_p = 0.9q_{pm}S/[\,1.1K_p + S\,(1 + S/(0.5K_I))]$.

3.7.2.4 Output Model Based on Mass Balances

The output model is derived by applying simplifying assumptions to the governing differential equations for the true penicillin fermentation model and solving them analytically. The cell biomass balance drops the cell lysis term, regarding it as negligible.

The other balance equations remain the same.

Cell biomass balance: $dX/dt = X(\mu - D)$

Substrate balance: $dS/dt = -\sigma X + (S_f - S)D$

Product balance: $dP/dt = q_pX - P(D - K)$.

The analytical solution is evaluated at one sampling time interval into the future for one-step-ahead prediction. The solution was obtained by applying the following assumptions:

(i) constant specific growth rate, μ;

(ii) specific product formation rate decays: $q_p(t) = q_p(t_0) + k(u)(t-t_0)$;

(iii) constant specific substrate consumption σ;

(iv) constant substrate feed concentration, S_f; and

(v) constant substrate feed rate, F.

Assumption (v) allows us to solve the overall mass balance, giving the fermentor working volume as a simple linear function of time. Substituting this into the definition of dilution rate (D=F/V) and solving the biomass balance gives the expression for the biomass concentration. The resulting formula for X, along with D, was substituted into the substrate and product balances to arrive at solutions for residual substrate concentration and product concentration, respectively.

Fermentor working volume: $V(t+\Delta t) = V(t) + F\Delta t$

Biomass concentration: $X(t+\Delta t) = X(t)\exp(\mu\Delta t)/(1 + D(t)\Delta t)$

Residual substrate concentration:

$$S(t+\Delta t) = S_f(t) - (S_f(t) - S(t))/(1 + D(t)\Delta t) - X(t+\Delta t)[1 - \exp(-\mu\Delta t)]\sigma/\mu$$

Product concentration:

$$P(t+\Delta t) = X(t+\Delta t)\{[P(t)/X(t) - q_p/(\mu+K)]\exp[-(\mu+K)\Delta t] + q_p/(\mu+K)\}.$$

3.7.2.5 Hybrid Penicillin Fermentation Model

Figure 3.7 depicts the 6-input/3-output hybrid model used in this study. Prior knowledge enters as the default model and the balance equations enforced as an output model. The hybrid accepts as input the three state variables at time t and the three exogenous variables: dilution rate D, feed substrate concentration S_f, and sampling interval Δt. It outputs predictions of the process state at time $t + \Delta t$. The default model consists of simplified specific rate correlations that take as input the current state and output the parameters $\mathbf{z} = [\mu, q_p, \sigma]^T$. These parameters are assumed to be constant over the sampling period and are used directly in the output model to compute the state at $t + \Delta t$.

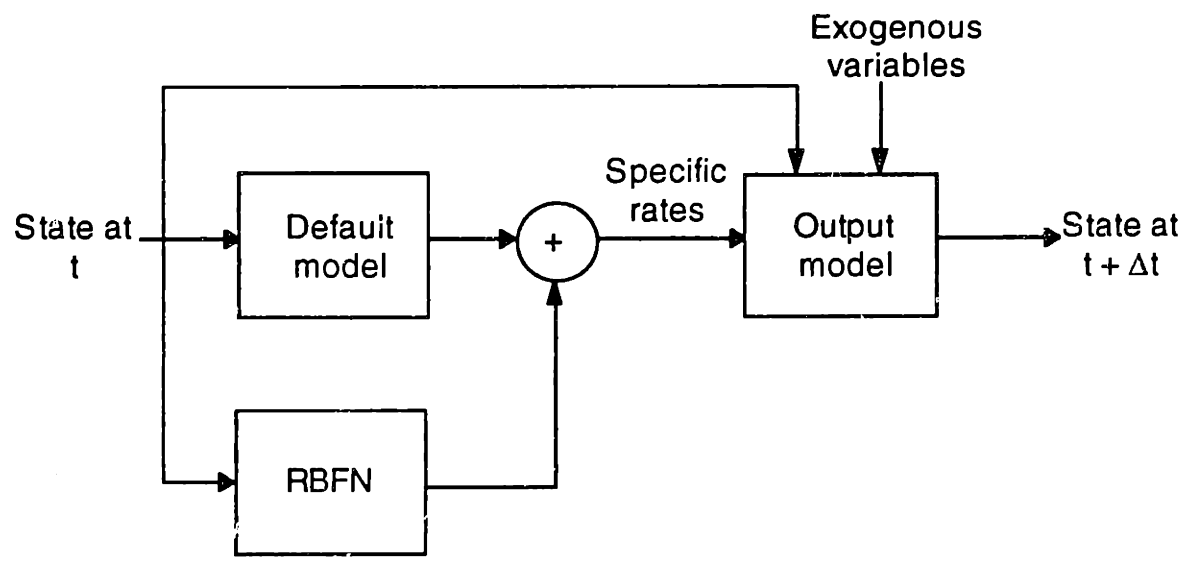


Figure 3.7: Hybrid network for fed-batch penicillin fermentation

The RBFN used in the hybrid model consisted of four nodes with an overlap factor of one (used in setting the radial unit widths). This architecture was chosen as the minimal network capable of accurately predicting the data. Maximum likelihood estimation (least-squares regression) yielded a model with good generalization properties. Therefore, the full hybrid network was trained by finding the network weights that minimized the least-squares objective function using the BFGS quasi-Newton method.

The process data used in training were simulated by a mechanistic model consisting of nonlinear differential balance equations and algebraic specific rate correlations. Gaussian white noise with a standard deviation of 5% was added to the simulation results to represent measurement error in each state variable. Four 200-hour fermentations sampled at 3-hour intervals and subjected to various feed schedules served as experimental data. Table 3.4 lists the operating conditions for these runs. The input/target pairs used as training data were built by pairing state variables at time t with the state at $t + \Delta t$. A total of 268 training points resulted. Prior to training, all data were mean-centered and unit variance scaled, N(0,1). Four additional 200-hour fermentations, shown in Table 3.5, were used to test the generalization and extrapolation capabilities of the model.

Table 3.4: Training data operating conditions

Run	Conditions
1	Standard run (F = 0.11 l/hr, S_f = 525 g/l)
2	Low feed substrate (S_f=480 g/l)
3	Variable feed concentration (S_f uniformly distributed from 475 to 575 g/l)
4	Ramped feed profile

Table 3.5: Test data operating conditions

Test	Conditions
1	Low feed rate (F = 0.02 l/hr)
2	Low feed substrate & high biomass (S_f=200 g/l, X=15gDCW/l)
3	High-variability feed concentration (S_f uniformly distributed from 420 to 630 g/l)
4	Long sample time (t = 6 hr)

For comparison purposes, the same data used to train the hybrid were used to train a conventional back-propagation network (BPN) and an RBFN. The architectures for both networks were optimized by selecting the minimum prediction error network as indicated by cross-validation, a procedure that involves a series of train/test subset partitions of the training data (Eubank, 1988; Härdle, 1990; Weiss and Kulikowski, 1991). The optimal BPN contained one hidden layer with 20 hidden sigmoidal nodes and was trained using the BFGS quasi-Newton method in least-squares minimization. The optimal RBFN had 45 hidden radial units with overlap factor of 6.

3.7.2.6 Results

Results of applying the hybrid model appear in Figures 3.8 and 3.9. Figure 3.8a plots biomass concentration versus time for the hybrid, the default, and the actual process; while, Figure 3.8b plots change in biomass during Δt. It is readily seen that the hybrid model compensates for the overestimation of biomass growth by the default model. Similarly, Figures 3.9a and 3.9b show that the hybrid compensates for the underestimation of product formation by the default.

Compared to the two black-box neural network approaches, the hybrid model is superior. Figures 3.10 and 3.11 compare the hybrid to the BPN and RBFN, respectively, in predicting product for a typical training run. The hybrid model has less variability and is more accurate than both of the black-box networks. Also, the hybrid always predicts values that are consistent with reality, unlike the other networks, which predict negative product concentration early in the fermentation.

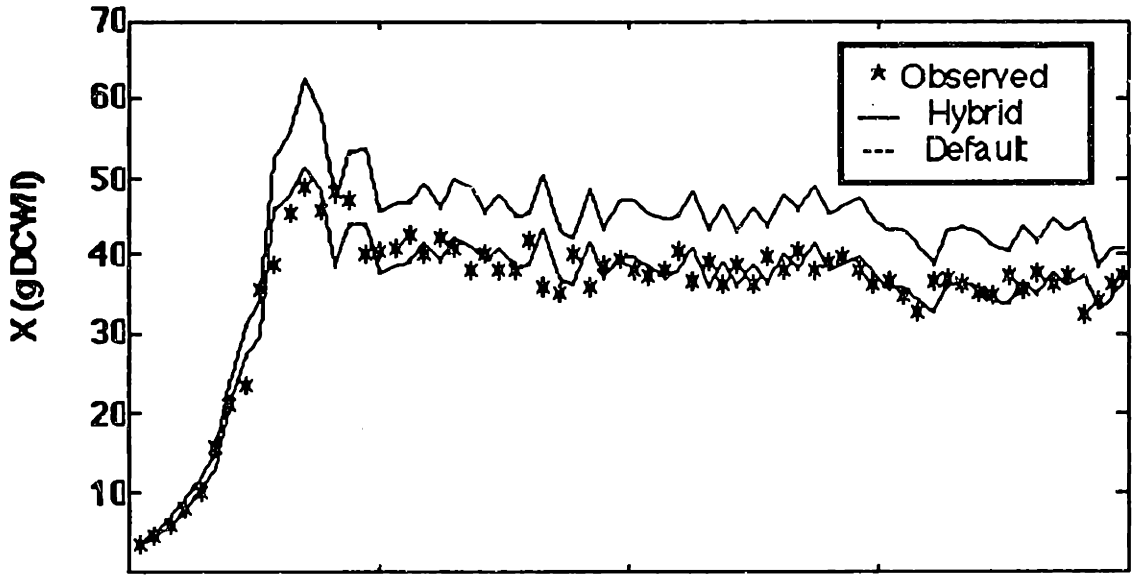


Figure 3.8: Biomass versus time: hybrid network, default model and actual process

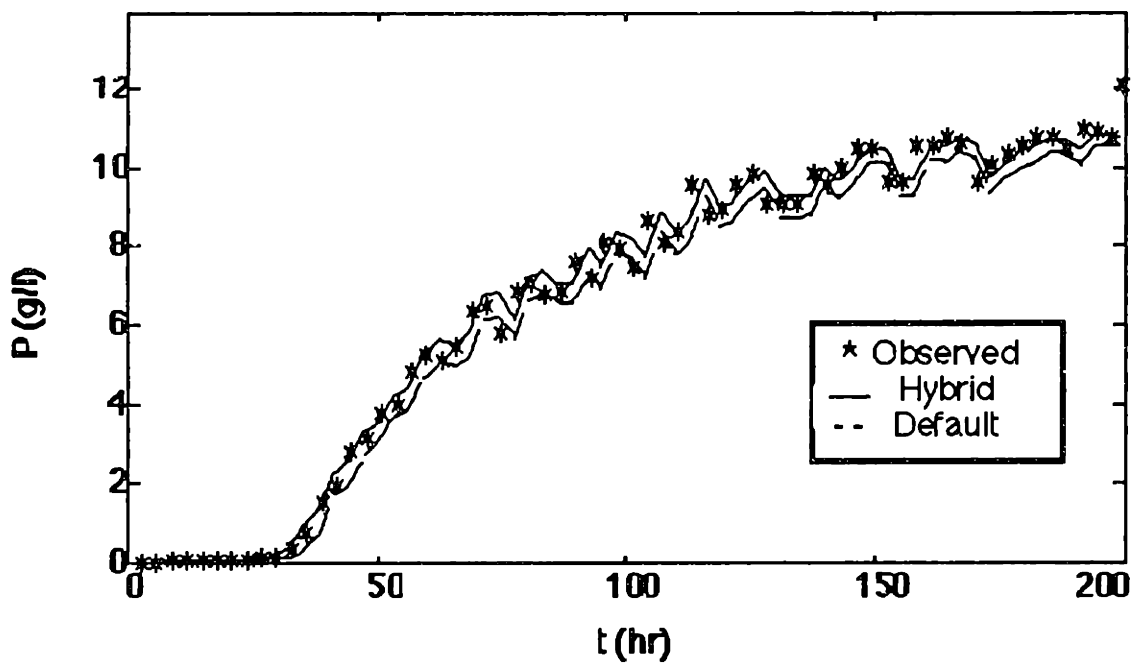


Figure 3.9: Product versus time: hybrid network, default model and actual process

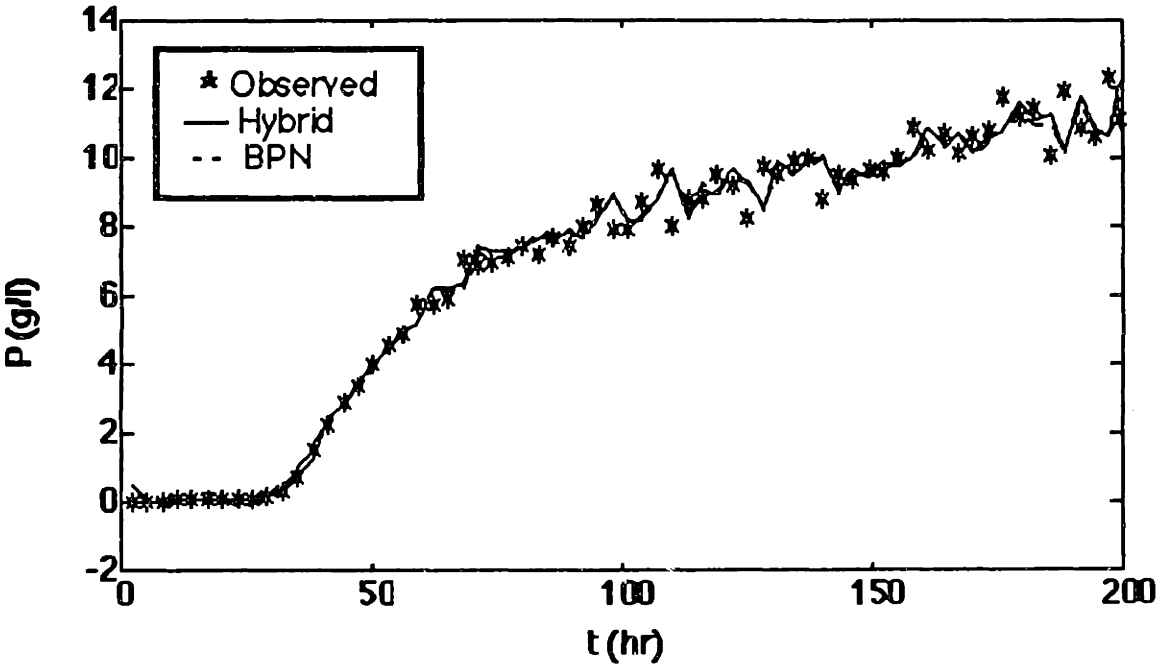


Figure 3.10: Product versus time. Pure 3-BPN.

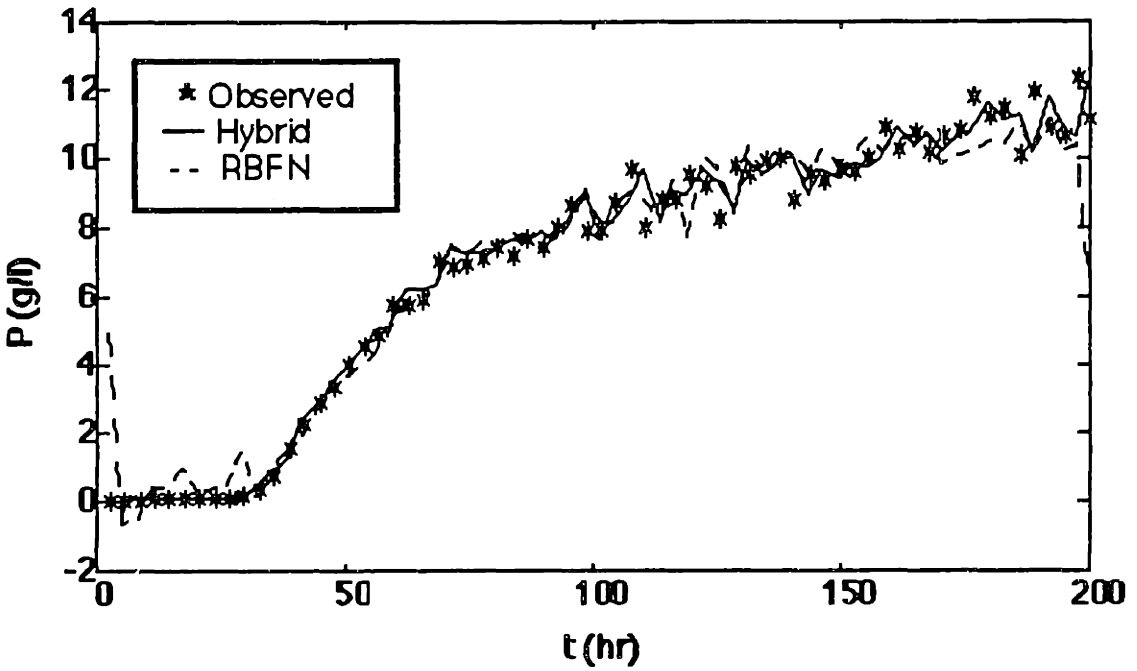


Figure 3.11: Product versus time. Pure RBFN

A more dramatic improvement over the black-box networks results when extrapolating. Figures 3.12a and 3.12b present the biomass and product predictions for the hybrid, BPN and RBFN for the low-substrate test fermentation. As expected, the RBFN cannot extrapolate reliably beyond its region of training data. The BPN performs better but also suffers when given inputs from outside of its training domain. The hybrid, on the other hand, maintains accurate and consistent predictions over a wider region of input space.

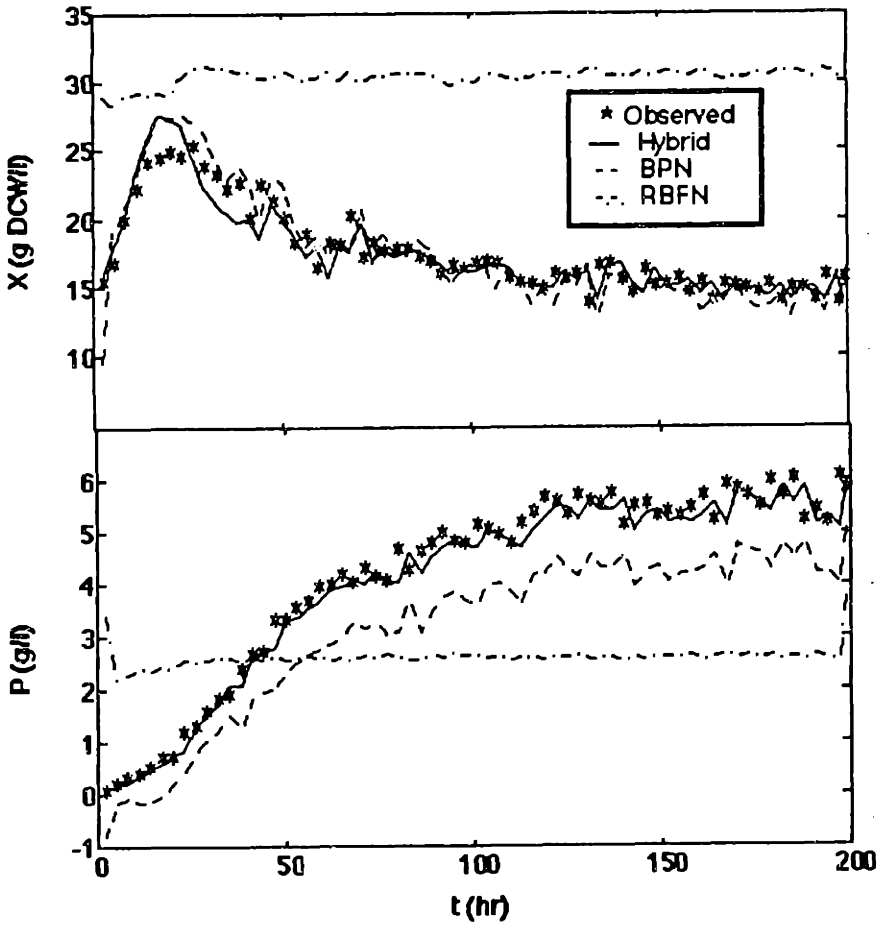


Figure 3.12: Extrapolation: (a) biomass and (b) product versus time.

The ability to learn the underlying relationship between the state variables and the specific rates is key to accurate extrapolation by the hybrid model. Figure 3.13 shows that the hybrid model was able to accurately learn the specific rate correlations. This results in improved extrapolation and greater insight into process behavior than is afforded by a conventional neural network approach.

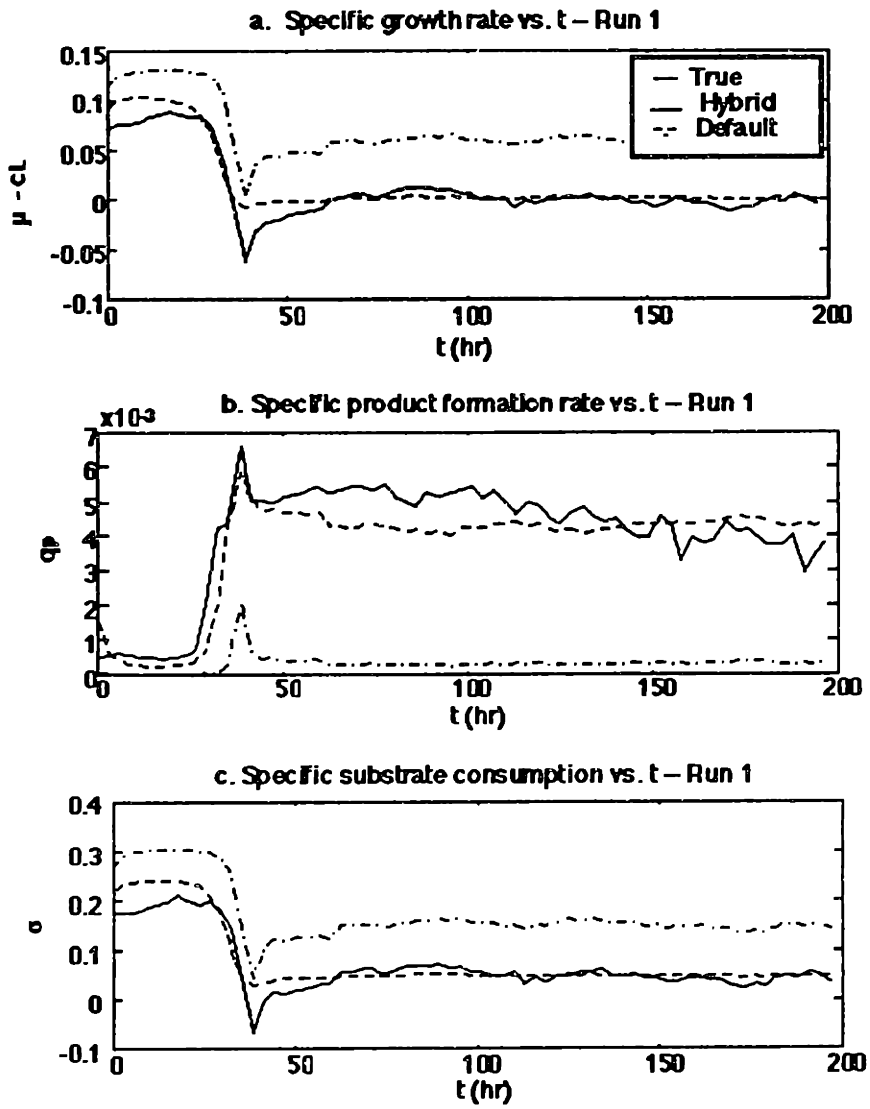


Figure 3.13: Specific rates versus time. Hybrid network.

3.7.2.7 Conclusions

This case study showed that combining prior knowledge of the process domain with a radial basis function network in a hybrid semi-parametric model provides accurate, consistent and reliable predictions. Prior knowledge improves performance by serving as a default estimate of process behavior in the absence of training data and as equality constraints that force the model to output predictions consistent with the physical process. Such an approach also yields a model that is more easily interpreted.

3.7.3 Fast Cross Validation Performance Study

To assess the usefulness of FCV in selecting models nonlinear in the parameters, we performed a Monte Carlo study. We also chose to investigate GCV as a complexity-penalized alternative to FCV because of the variety of applications in which it has proven useful (Craven and Wahba, 1979; Friedman, 1991; Gu and Wahba, 1991; Lu and Mathis, 1992; Lukas, 1993; O'Sullivan and Wahba, 1985; Schumaker and Utreras, 1990; Stein, 1990; Wahba, 1990). Also, we considered leave-one-out cross validation. However, computation of CV by leave-one-out parameter estimation was too expensive to be practical in assessing the 1540 models trained in the study. As a consequence, our comparison of FCV to CV was restricted to an assessment of the relative computation times of the two methods. This was done by estimating the ratio of FCV computation time to leave-one-out cross validation time, which is roughly the percentage of exceptional data points requiring iterative estimation of $c_{[k]}$ by FCV.

Given the opportunity to generate as much data as desired, we generated large test sets of size N_T and treated the predictive mean squared error (PMSE) as the true predictive capability of each model. PMSE is given by the following formula:

$$\mathrm{PMSE}(K) = \frac{1}{N_T} \sum_{i=1}^{N_T} \left[y_i - \mathrm{f}_K\left(\mathbf{x}_i; \mathbf{w}_K^*\right) \right]^2 \tag{3.52}$$

where $\mathbf{x}_i$ and y_i, $\forall\ i=1,\ldots,N_T$, are data sampled from the same joint probability density function $p(X,Y)$ as the training data D.

Armed with this estimate of true predictive capability, we sought answers to the following questions:

(1) Will a modeler *probably* select the model with best predictive capability if FCV is used for model selection? In other words, is the most probable model size selected by FCV the same as that selected using PMSE?

(2) How frequently would a modeler using FCV select the best model or one essentially as good as the best model? I.e., how frequently does FCV select the same model size as PMSE? And, if the selected models are different, how frequently is the difference in PMSE between the model selected by the new criterion statistically indistinguishable from the model with the best PMSE?

(3) Can a modeler use FCV to distinguish between underfit, well-fit, and overfit models? I.e., does FCV have sufficient precision to discriminate between underfit and well-fit models? Between overfit and well-fit models?

(4) Can a modeler use the criteria with confidence at small sample sizes? How robust is FCV to sample size? How does sample size N affect the answers to the above three questions?

Empirical answers to these questions can be found by generating samples of the distributions of each of the three criteria, PMSE, FCV, and GCV, and comparing the estimated statistics of their distributions. We present such an analysis of a single case study with a single family of models: three-layer back-propagation neural networks (3-BPNs). Below, we present the network architecture, identify the parameter vector $\mathbf{w}_K$, and derive the expressions for the derivatives needed to estimate Δy_k for a single-output 3-BPN. Then, we describe the data used in the study.

3.7.3.1 Data

In analyzing model selection for 3-BPNs, our case study used the following data model. We simulated the data by sampling input variables $\mathbf{x}$ distributed uniformly over the unit square $[0,1]\times[0,1]$. Given $\mathbf{x}$, we computed the dependent variable y by adding noise ε to the true nonlinear model $f(\mathbf{x})$:

$$y = f(\mathbf{x}) + \varepsilon = \frac{1}{2}\left[\sin\left(3\left(\pi x_1 - \frac{1}{2}\right)\right) + a\left(12\left(x_2 - \frac{1}{2}\right)\right)\right] + \varepsilon, \qquad (3.53)$$

where a(.) = the sigmoidal function defined by Eqn. (3.3) and $\varepsilon \sim N(0,\sigma^2)$ (normal, zero-mean, and independent identically distributed) with $\sigma^2 = 0.01$.

To assess the effects of sample size N, we computed the prediction criteria for moderate and small N. The values of N that constitute "moderate" and "small" sample sizes are primarily dependent on (*i*) the dimensionality n of the input space, (*ii*) the variance of the noise in the output variable, and (*iii*) the nonlinearity of the true model. For $f(\mathbf{x})$ and ε described above, values of N=100 for the moderate sample size and N=50 for the small size were found to be suitable. Therefore, in the first phase, 110 data sets of 2600 points ($\mathbf{x}$-y tuples) each were generated. Each set was then randomly partitioned into a training set of N=100 points and a test set of N_T=2500 points. Likewise, in the second phase, 110 sets of 2550 points were generated, and N=50 and N_T=2500 were used.

Before training, the data were standardized (i.e., made to have zero mean and unit variance). For both sample sizes N that we investigated, seven networks of sizes $K \in \{5, 10, 15, 20, 25, 30, 35\}$ were trained on each of the 110 N-point data sets -- for a total of 2(7)(110)=1540 network trainings. After each training, we calculated the PMSE, FCV, and GCV for the resulting network. The sample distributions of the statistics at each N were then analyzed and compared.

Parameter estimation for the 3-BPN was done using Newton's method given by Eqn. (3.51) to minimize the objective function of Eqn. (3.40). The final matrices $\mathbf{M}$ and $\mathbf{N}$ computed for Eqn. (3.51) were used to calculate $\hat{\mathbf{H}}$ in Eqn. (3.48). As shown in Eqns. (3.42) and (3.43), $\mathbf{M}$ and $\mathbf{N}$ are functions of the model Jacobian $\nabla\mathbf{f}$. The Jacobian is calculated by taking advantage of the fact that a 3-BPN is a separable nonlinear model. Golub and Pereyra (1973) show that the linear parameters $\boldsymbol{\beta}$ can be expressed as a function of the nonlinear parameters $\boldsymbol{\alpha}$ when the parameters are estimated using the least-squares objective function: $\boldsymbol{\beta} = \mathbf{Q}(\boldsymbol{\alpha})\mathbf{y}$, where $\mathbf{Q}(\boldsymbol{\alpha})$ is the $(K+1)\times N$ Moore-Penrose generalized inverse of $\mathbf{A}(\mathbf{X};\boldsymbol{\alpha})$. For full-rank $\mathbf{A}$, $\mathbf{Q}(\boldsymbol{\alpha})$ is $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$, the familiar linear regression result (i.e., Eqn. 3.20). Substituting $\mathbf{Q}(\boldsymbol{\alpha})\mathbf{y}$ for $\boldsymbol{\beta}$ in Eqn. (3.4) and, for

conciseness, suppressing the arguments, gives $\mathbf{f}=\mathbf{A}\beta=\mathbf{AQy}$. Thus, $\mathbf{w}$ has been reduced to α. Taking the derivative of $\mathbf{f}$ with respect to α yields $\nabla\mathbf{f}$ (Golub and Pereyra, 1973):

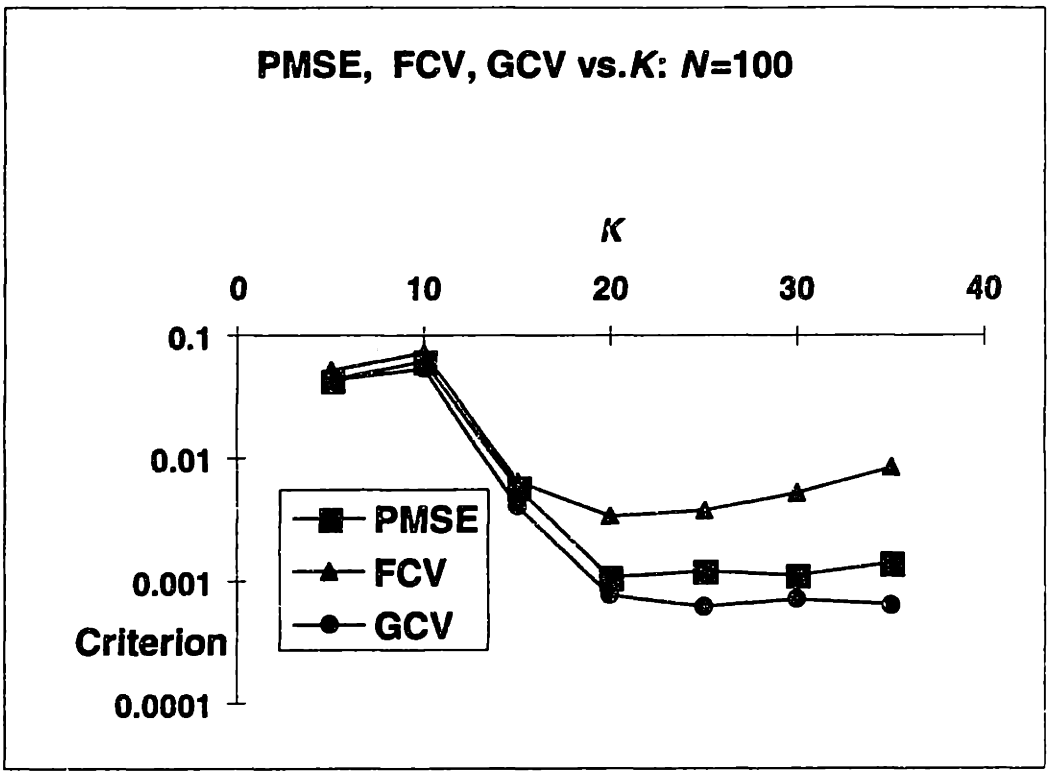
$$\nabla\mathbf{f} = [\mathbf{D}(\mathbf{A})\mathbf{Q} + \mathbf{A}\mathbf{D}(\mathbf{Q})]\mathbf{y}, \tag{3.54}$$

where the Fréchet derivatives $\mathbf{D}(\mathbf{A})$ and $\mathbf{D}(\mathbf{Q})$ are taken with respect to the $[K(n+1)]\times 1$ vector α. The bracketed term in (46) is an $N\times[K(n+1)]\times N$ tensor that, when multiplied by the $N\times 1$ vector $\mathbf{y}$, yields the $N\times[K(n+1)]$ Jacobian $\nabla\mathbf{f}$. Golub and Pereyra (1973) provide the formula for computing $\mathbf{D}(\mathbf{Q})$.

3.7.3.2 Results and Discussion

We begin with the first question: does the most probable model size K selected by FCV and GCV agree with that selected by PMSE? This is a matter of locating the modes of the distributions at each K for both N. The distributions of the three statistics PMSE, FCV, and GCV were analyzed from the 110-point samples for each K and both N. Ideally, a good predictive metric would be distributed chi-square because ε is normal. Failing that, the metric would at least have a unimodal distribution. As it turned out, all three criteria were distributed near log-normal, but with heavier tails and slight skewness towards the low ends. Thus the medians of the log-distributions are better estimates of mode than the log-means.

Plotting the medians of the log-distributions helps to answer the first question. Figures 3.14a and 3.14b present the medians of each metric at each K for N=100 and 50, respectively. For N=100 (Fig. 3.14a), the three criteria agree in selecting K=20 as the best model. This K corresponds to the bend in the PMSE curve. At K<20, the models are underfit; and, at K>20, the models have similar predictive capability but their greater complexity makes them less desirable. Probably, at this sample size for this problem, a modeler using FCV or GCV would select the best or nearly the best model from the candidate set. The results, however, are not as promising at the smaller sample size. Figure 3.14b shows that PMSE selects K=15 as the best model. In contrast, FCV did not distinguish between the underfit and overfit models, while GCV favored the overfit models.



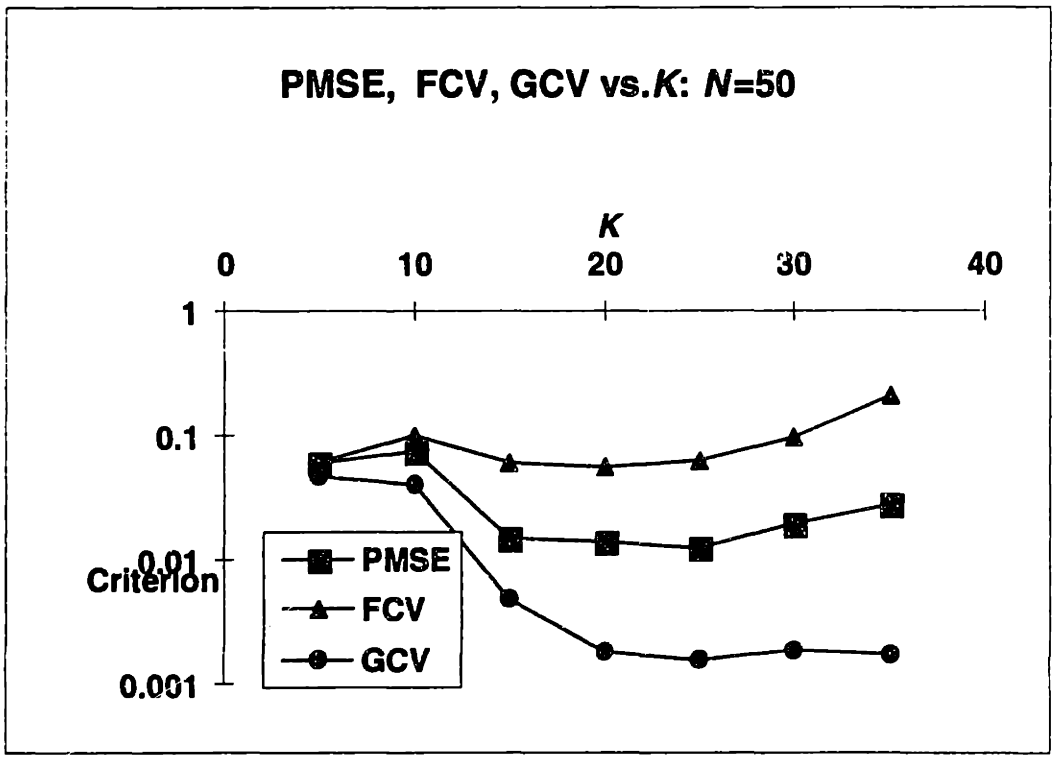
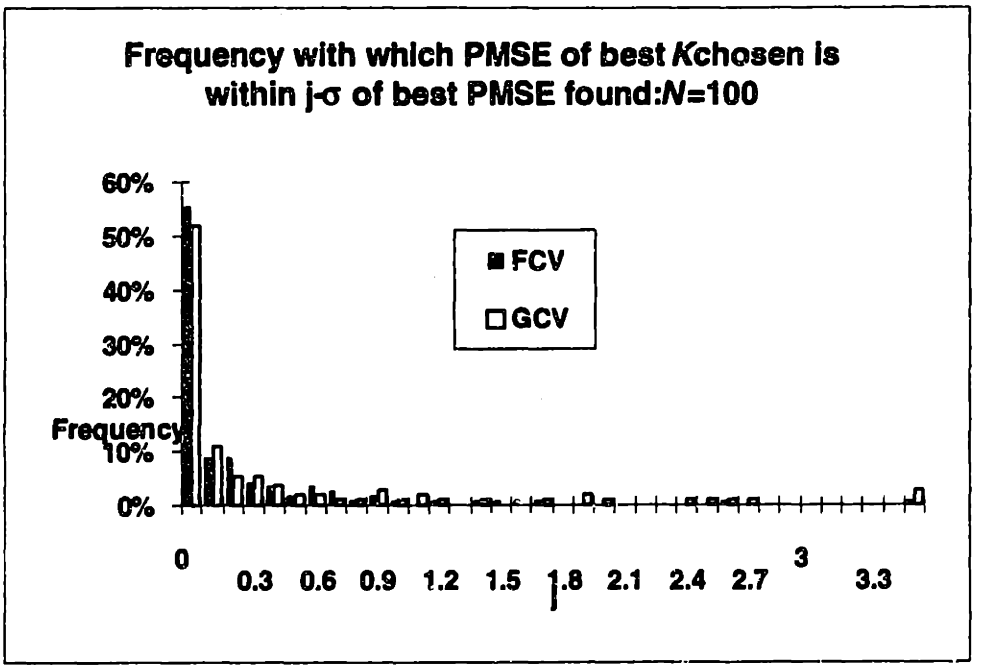


Figure 3.14: Medians of PMSE, FCV and GCV for (a) N=100 and (b) N=50

Addressing the second question, Figures 3.15a and 3.15b illustrate how frequently FCV and GCV select models that are indistinguishable from the model with the best PMSE for N=100 and N=50, respectively. FCV selects a model with a PMSE within one standard deviation of the best PMSE 93% of the time at N=100. However, GCV is less precise, selecting a model with PMSE within one standard deviation of the best 85% of the time. At N=50, FCV selects a model this close to the best only 86% of the time, while GCV drops to 70%.

The inability of FCV to distinguish between well-fit and underfit models addresses the precision of the criteria, which concerns the third question posed above. To answer that question, we estimated the dispersion (spread) of the distributions. The variances of the log-distributions were used to measure spread in the absence of any other measure for PMSE and FCV. Figures 3.16a and 3.16b contrast the discriminatory resolution of FCV for N=100 and N=50, respectively. The "x"s represent estimates of the 95% confidence limits on the estimates of location. It is readily seen that for N=50, FCV is unable to distinguish between all networks of size less than K=25. The result is that the smallest network would be selected, thus yielding an underfit model.

Further insight can be obtained by determining whether the model selected as best by each criterion is selected significantly more often than the other models. Figures 3.17a and 3.17b show how often each model size was chosen as best by each criterion. For this problem and at these sample sizes, even PMSE has considerable uncertainty in distinguishing the best model: the most frequently selected model is only selected 27% of the time for N=100 and only 23% of the time for N=50. However, model sizes with median PMSE within the 95% confidence limits of the best PMSE represent 69% of the models. FCV gives comparable results, tracking the selection behavior of PMSE quite well. On the other hand, at both values of N, GCV selects larger, overfit models far more frequently than PMSE and FCV.



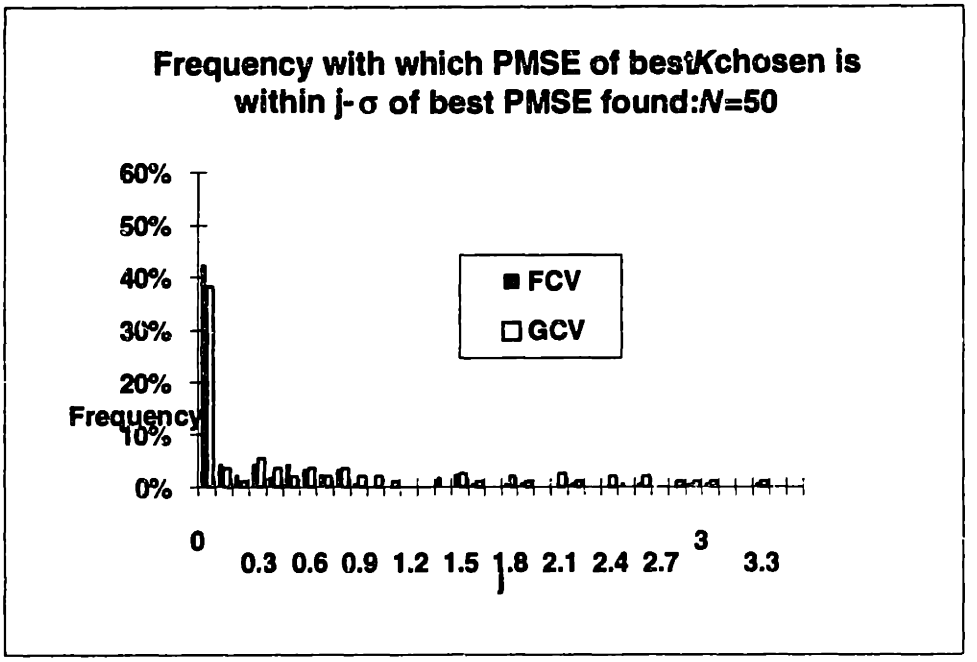
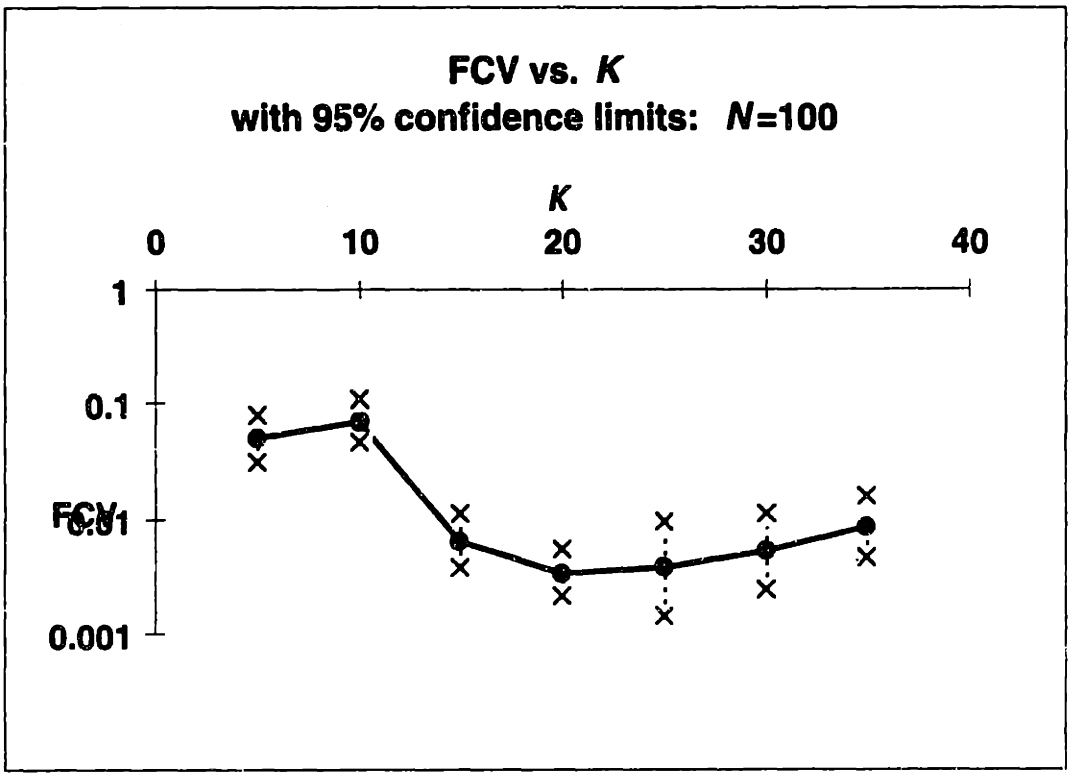


Figure 3.15: Frequency that FCV and GCV select models that are indistinguishable from the model with the best PMSE for (a) N=100 and (b) N=50



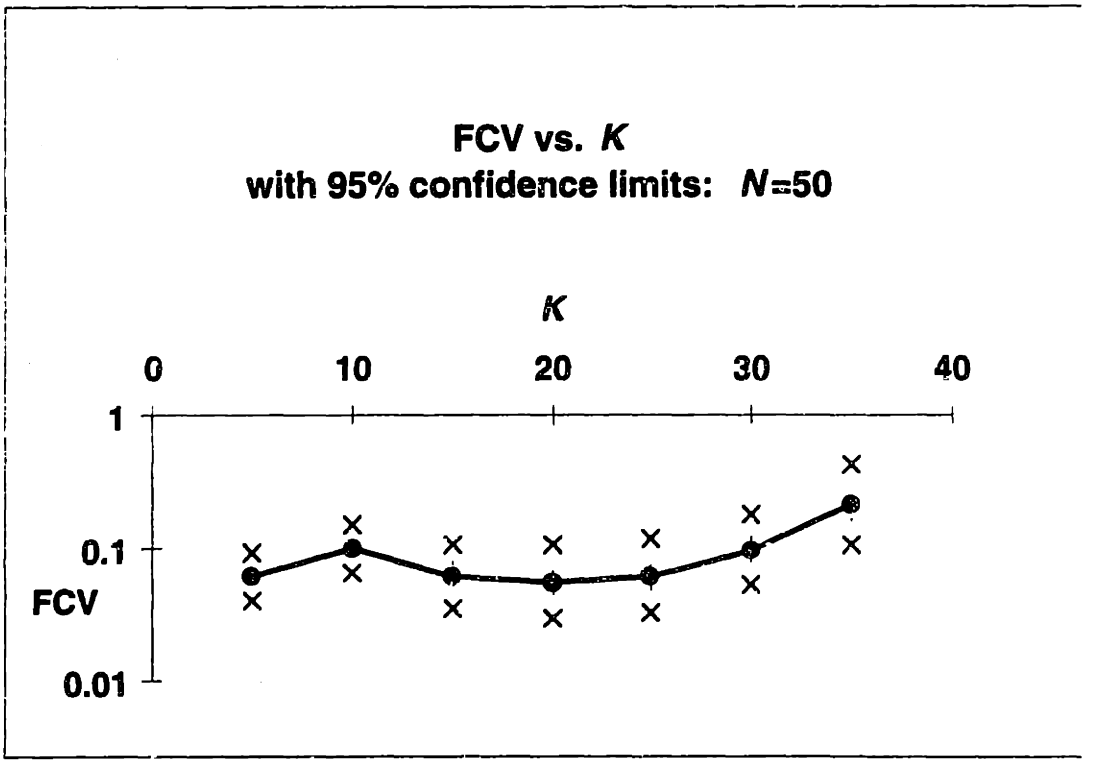
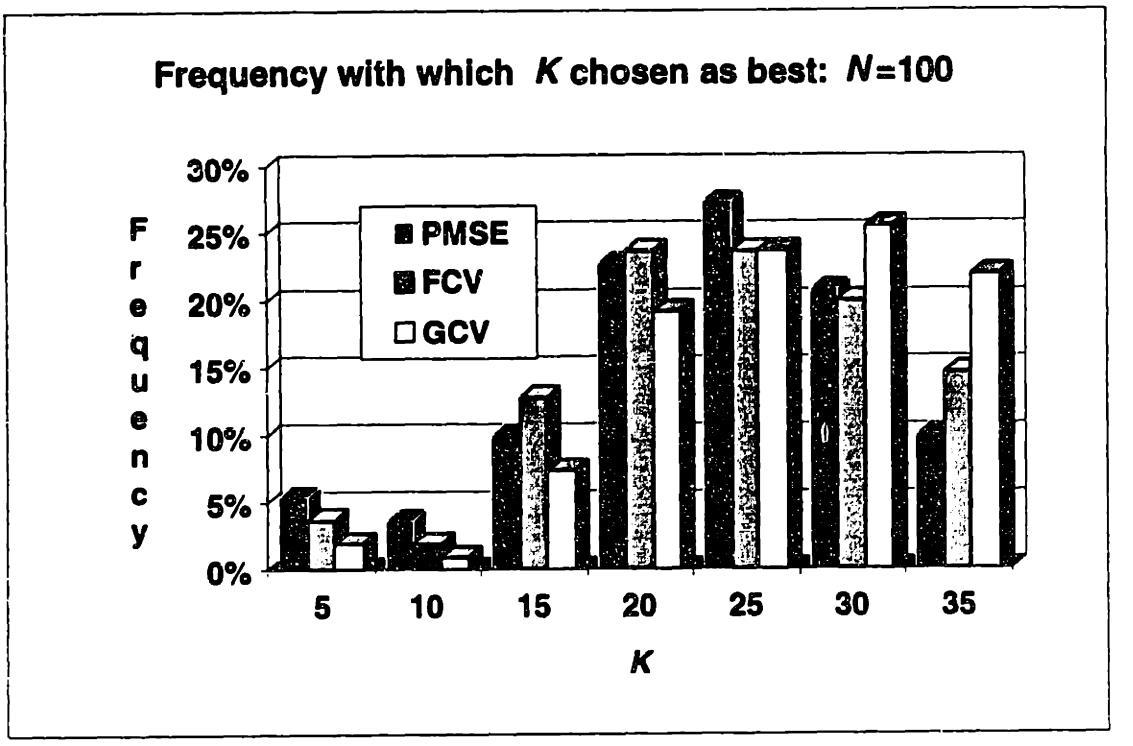


Figure 3.16: FCV with 95% confidence limits for (a) N=100 and (b) N=50

The low frequency at which even the most frequently selected model size is chosen can be explained by examining how the effective number of parameters p_{eff} changes with the number of actual parameters p, which is proportional to the number of hidden nodes K. Figures 3.18a and 3.18b show that beyond a certain point, the plot of p_{eff} vs. K begins to flatten, becoming insensitive to the number of nodes. This is due to the fact that the hidden nodes of the 3-BPN are non-orthogonal basis functions, which exhibit collinearity, i.e., $\text{rank}(\mathbf{A}(\mathbf{X};\alpha))<K+1$. Thus, learning capacity plateaus, and mean training squared error and p_{eff} become constant with increasing network size. So a model with, for example, 25 hidden nodes is indistinguishable in predictive capability from one with 20 nodes, and any criterion will distribute its choices for best model among a range of network sizes. Similar conclusions are drawn from the theoretical arguments of Plutowski, *et. al* (1994), who noted that even though CV asymptotically avoids overfitting, the stochastic nature of the problem causes the selection of networks that are more complex than the optimal architecture to occur with probability greater than zero.

The fourth question, concerning the robustness of the criteria to sample size, has been addressed above in the context of the other three questions. We conclude that, a sample size of N=50 may be insufficient for discriminating among the models' abilities to learn Eqn. (3.53) subject to the given probability distribution of the data.

Finally, we evaluated the computational efficiency of FCV. This problem required iterative estimation of $c_{[k]}$, on average, for only three of the 100 points in each sample. So, FCV required 3% of the computation time required by leave-one-out cross validation. However, for N=50, iterative estimation was required, on average, for 6% of the sample when evaluating models with small K, and as much as 25% for the more complex models. This is due to the sparseness of the data at such a small sample size and overfitting of the data by the larger models. Under these circumstances, more points have a large influence on the model fit and, hence, there are more leave-one-out residuals large enough to require iterative estimation.



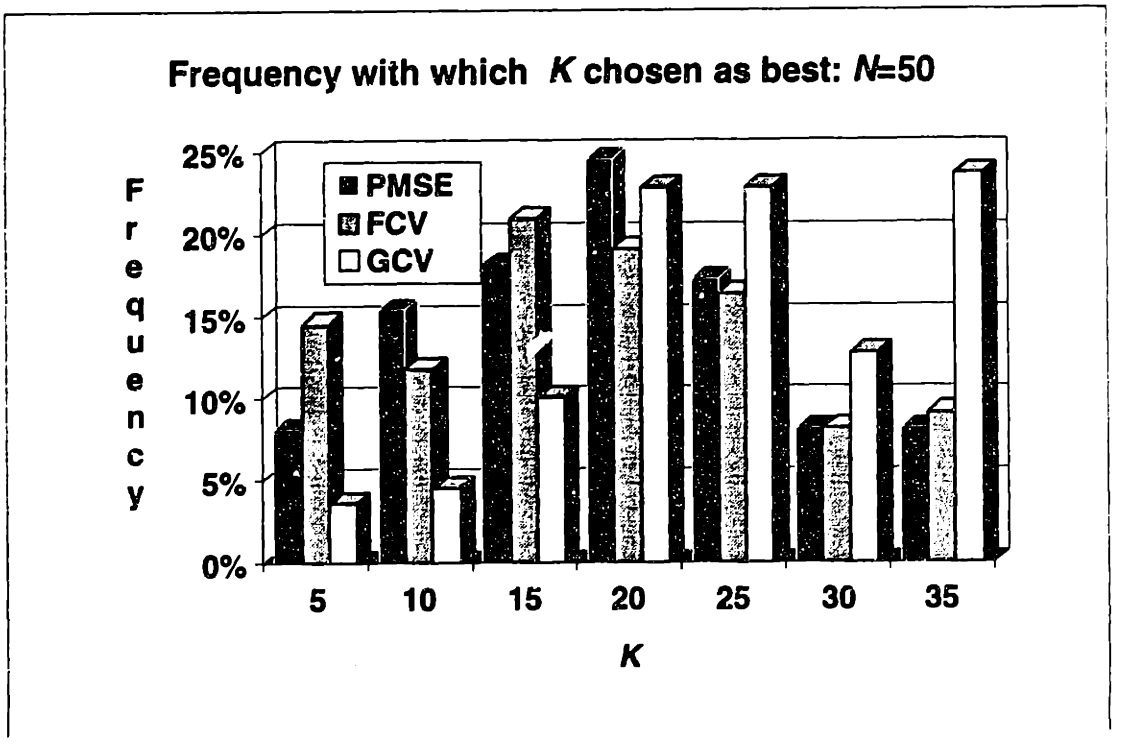


Figure 3.17: Frequency that each size was chosen as best for (a) N=100 and (b) N=50

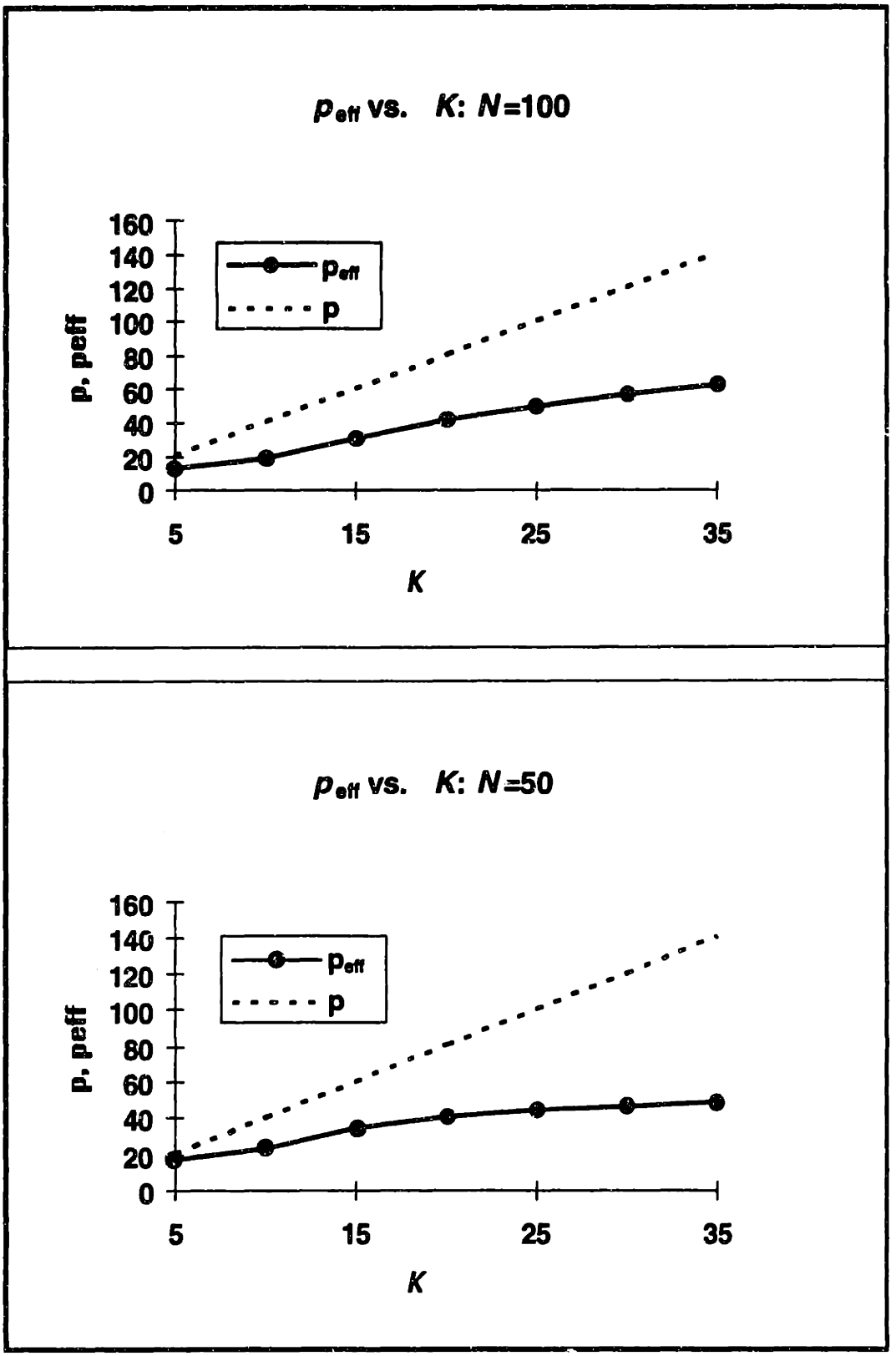


Figure 3.18: p_{eff} versus K for (a) N=100 and (b) N=50

3.7.3.3 Conclusions

FCV was derived from sensitivity analysis as a nonlinear analogue to the CV criterion for linear models. Monte Carlos simulation showed the method to be useful in selecting the optimal number of nodes in the hidden layer of three-layer back-propagation neural networks. FCV was effective at moderate sample size of N=100, but less so at sample size N=50. On the other hand, GCV based on the effective hat matrix derived from sensitivity analysis tended to select overfit models at both sample sizes, with the tendency being more pronounced at N=50. The lack of effectiveness at small sample sizes is expected considering that the favorable theoretical properties of such criteria only hold for asymptotic analysis.

The new method estimates an effective hat (or influence) matrix for nonlinear models. The hat matrix has proven useful in the analysis and diagnosis of a wide variety of nonparametric regression models called linear smoothers (Buja, *et al.* 1989; Chatterjee and Hadi, 1987; Eubank, 1984). In computing an effective hat matrix for a nonlinear model, some of the benefits of linear regression theory can be gained. An example of this is the estimate of the effective number of parameters of the model $p_{eff} =$ trace($\hat{\mathbf{H}}$). This value can be used in complexity-penalized criteria such as GCV and GPE to estimate the overfittedness of the model.

Other applications that have benefited from using GCV for linear models such as regularization in inverse problems, Bayesian estimation, and constrained parameter estimation may benefit from applying FCV for nonlinear models. Future work is necessary to determine the effectiveness of FCV in system identification for nonlinear dynamic models.

Chapter 4

Probabilistic Paradigm

Probability theory provides a rational means of (1) including prior knowledge into model synthesis, (2) combining several models into a single composite, (3) compensating for the inadequacies of the data that increase uncertainty, (4) determining the generalization capabilities of the regressed models, and (5) comparing alternative models. This chapter presents the probabilistic approach to developing a complete process modeling methodology.

4.1 Rationale and Approach

Models are used in systems engineering to estimate the states of a process. In this context, we broaden the term "state estimation" to include such applications as prediction of future states using a regression model; estimation of the true current or past state given corrupted measurements from the process, e.g.. data rectification; and diagnosis of process faults based on qualitative and quantitative observations. These state estimation applications, which apply knowledge in the form of models, are the motivation for model synthesis.

The state estimation problem may be posed in a decision theoretic framework that brings to bear all relevant knowledge, application-specific preferences, and available data to solve the problem. This framework recognizes the inherent uncertainty associated with process knowledge and the inherent stochastic nature of the physical phenomena and measurements of the process. In doing so, optimal state estimation is achieved by exploiting probability theory to uniformly manage both uncertainty and stochastic phenomena while applying both process knowledge and data.

4.2 Decision Theory

In process modeling, the primary decision problem is to optimally estimate the process state. This in turn requires solution of subproblems such as choosing the model structure and selecting model parameters. To resolve these problems requires that the decisionmaker use all available information to combat the shortcomings introduced by uncertainty. Considering that decision theory was derived for just such problems, this thesis investigates a decision theoretic framework for formulating the process modeling problem.

Before casting the problem in a decision theoretic framework, definitions of the quantities involved in the problem statement are necessary. In state estimation, we wish to estimate the variables S that characterize the state. To do so, we require some evidence E and an hypothesis H from which we base our deductions about the process state. The evidence E in this setting may be process measurements or prospective inputs to the process for which we wish predictions. The hypothesis H is our assumed knowledge base defining the context of our inferences. H is not merely the assumed model structure, as it was in the function-oriented approach of Chapter 3, rather it includes all relevant assumptions, relations, and constraints selected from knowledge of the process, engineering principles, the physical sciences, and heuristics drawn from empirical observations (common sense). H is but one hypothesis drawn from a possibly infinite set of candidate hypotheses $\mathcal{H}$. In general, there also exist data D, independent of E, that may be used to augment H.

For example, in simple regression the state estimation problem is to predict the dependent variable, say y, given specific values of independent variables, x. The prediction is based on a model, $y=f(x;\theta)$, parameterized by θ. The parameters θ are independent of x and relatively constant with respect to x and y. They are usually quantities summarizing some aspect of the relation $f(x)$ but they may also be process variables. At the time at which state estimation is performed, the parameters θ will have already been regressed on data D. These data are past x-y tuples independent of the values of x for which the prediction

will be made. Thus, for state estimation in the regression case, the state S we wish to estimate is the predicted value of y; the evidence E is the given value of x at which we wish to predict y; hypothesis H includes the model relation $y=f(x;\theta)$ and assumptions about the distribution of measurement errors; and data D are the past x-y tuples summarized by the parameters θ.

In what follows, decision and probability theory are presented with the necessary mathematical rigor to make clear their application. For more rigorous derivations and discussions of the philosophical aspects of these theories, see Bernardo and Smith (1994), Berger (1985), or Howson and Urbach (1988). Also, with respect to notation, probabilities will be denoted by an upper case "P" followed by the event or proposition in parentheses: thus, "$P(X>0|H)$" denotes the probability of the proposition "the variable X is greater than zero." Probability density functions (pdfs) will be represented as "$p(x)$": thus, $P(X>0|H)=\int_0^X p(x|H)dx$. Expectations will be denoted as "$E[x|H]$", where $E[f(x|H)]\equiv\int_{-\infty}^{\infty} f(x|H)\,p(x|H)dx$. And, conditional probabilities, pdfs and expectations are written respectively as follows: $P(Y>0|x=x^*,H)$, $p(y|x=x^*,H)$, and $E[y|x=x^*,H]$. When the conditioning information is listed without the equation to a specific value it denotes that the information is fixed at a value that is clear from the context. In general, the bounds of integration will not be explicitly written on integrals but instead will be understood to be multidimensional and over all valid values of the variables over which the integrand is integrated. In contrast to Chapter 3, bold-face type will not be used to denote vectors and matrices, because the dimensionality of quantities will be explicitly stated or be clear from the context and case – lower case for scalar and vectors; upper case for matrices. Subscripted braces on quantities will represent a subset of elements: for example, the quantity X represents the N quantities $\{x_N, x_{N-1}, ..., x_1\}$, while the quantity $X_{\{<i\}}$ represents the subset $\{x_{i-1}, x_{i-2}, ..., x_1\}$, $i\leq N$.

Note that by definition, all probabilities are conditioned on the knowledge context (here written as "H") in which the probability assignments and hence inferences are made. It is common practice in the literature to omit this implied conditioning when all probabilities concerned are derived within the same knowledge context. In this thesis, the conditioning is explicitly written to emphasize the prior knowledge that defines H and to stress the fact that an important step in process modeling is explicitly stating, validating and selecting among the subsets of constraints and assumptions that comprise a specific hypothesis H.

4.2.1 Decision Problems

A generic decision problem may be defined in terms of the decision to be made d(a), an action to be taken a, and a utility function U(d(a)) expressing the decisionmaker's preferences for one decision over another (Berger, 1985). The utility function U(d(a)) quantifies the value of a decision: so U(d(a)) increases with increasing preferability of d(a). The uncertainty associated with the information defining the actions and outcome of the decision is represented probabilistically by the pdf p(d(a)|I), which is conditioned upon all available information I.

For the generic state estimation problem, the decision d(a) is the selection of an estimate of the process state S. And, the pdf p(d(a)|I) is specified as p(S|E,D,H), where the evidence E, data D, and hypothesis H comprise all available information in the context of the decision. In this problem, rather than expressing our preferences in terms of an utility function, it is more natural, and provably equivalent, to express them in terms of a loss function L(S,S^*), where S^* is our chosen estimate of the true state S. Suitable loss functions are described below, but suffice it to say for now that L(S,S^*) decreases as S^* moves closer to the true state S.

The optimal coherent decision with respect to quantified utility U(.) is proven to be the decision d^* that maximizes the expected utility over the pdf p(.) or equivalently the decision that minimizes the expected loss over that pdf (Bernardo and Smith, 1994;

Berger, 1985). Therefore, the generic state estimation problem for loss function $L(S,S^*)$ can be formulated as such:

> **State estimation problem**: Given evidence E, estimate the optimal process state S under the assumptions imposed by an hypothesis H and in light of data D. Therefore, choose S^* that minimizes expected loss $L_E(S^*)$, where $L_E(S^*) \equiv E[L(S,S^*)|E,D,H] = \int L(S,S^*)p(S|E,D,H)dS$.

This formulation is general. It encompasses such varied cases as prediction with a regression model, rectification of corrupted data, and diagnosis of process faults. To make this abstract formulation more concrete, we present typical definitions of S, E, D and H in each of these areas. However, before proceeding, we define more notation. First, the state variables S are partitioned into two sets: $S=\{\tilde{x}, \tilde{y}\}$, where the "~" superscript denotes the true (noise-free) process state variables. We define $\tilde{x}$ to be those variables that are independent of the parameters θ, and define $\tilde{y}$ to be those that are dependent on $\tilde{x}$ and θ. The knowledge assumed in H dictates this partitioning and explicitly defines our assumptions about the distributions of $\tilde{x}$ and $\tilde{y}$. Of course, it is a fact of nature that what we assume in H about the true variables will not in reality coincide with the actual true variables in nature. After all, that is why we must resort to plausible reasoning rather than deductive reasoning.

We will see that the formulas we derive will also apply, after simplification, when either set of variables $\tilde{x}$ or $\tilde{y}$ is empty. The process measurements are denoted as unsuperscripted variables x and y. Then the errors between the measurements and true variables are defined as e_x and e_y, where $e_x \equiv x - \tilde{x}$ and $e_y \equiv y - \tilde{y}$. All lower case variables are assumed to be vector quantities: $x, \tilde{x}, e_x \in \Re^n$ and $y, \tilde{y}, e_y \in \Re^m$. Capitalized variables represent matrices. The data D then become $\{X,Y\}$, which are two matrices of the same row-dimension: $X \in \Re^N \times \Re^n$ and $Y \in \Re^N \times \Re^m$. Each row of the matrices represents a single measurement tuple $\{x_i, y_i\}$, $i=1,...,N$. Also the notation $X_{[i]}$ will be used to denote the

matrix X without the i^{th} row. Similarly, $X_{[1,2,\ldots,i]}$ will denote the matrix X without the first through i^{th} rows.

Regression: The entities *S*, *E*, *H*, and *D* for the state estimation problem in a regression context were described in the example above. The problem statement in this context becomes: Estimate the prediction *y* that minimizes expected loss with respect to $p(S|E,D,H)=p(\tilde{y}\,|\,\tilde{x},D,H)$, which is referred to as the "predictive distribution." Alternatively, when the independent variables are not known precisely but are corrupted in some sense, say by measurement noise, the estimated state *S* includes $\tilde{x}$ and $\tilde{y}$; and *E* represents the measured values *x*. In this case, known as error-in-variables regression, the prediction step is essentially the same as data rectification described below.

Data rectification: In this context, the goal is to estimate the true state *S* given measured states *E*, which are corrupted by noise, gross errors, or missing values. The estimation is subject to constraints dictated by our prior knowledge *H* and in light of past data *D*. For example, *S* are the true state variables $\tilde{x}$ and $\tilde{y}$; *E* are the measured values *x*; and *H* specifies probability density functions $p(x|\tilde{x},H)$, incorporating knowledge of the error structure, and $p(\tilde{x},\tilde{y}\,|D,H)$, incorporating constraints such as $f(\tilde{x},\tilde{y};\theta)=0$ and $g(\tilde{x},\tilde{y})\leq 0$ and correlation implicit in *D*. The problem statement becomes: Estimate the true variables $\tilde{x}$ and $\tilde{y}$ that minimize expected loss with respect to $p(\tilde{x},\tilde{y}\,|x,y,D,H)$.

Fault diagnosis: Here the goal is to estimate the most probable state of the process *S* associated with process faults that are either implicitly or explicitly represented by observations *E*. The problem may be solved using a process model *H* comprised of the marginal and conditional probability distributions necessary to derive $p(S|E,D,H)$, e.g. a Bayesian belief network, which again incorporates prior knowledge and information from past data *D*. In this context, *S*, *E* and *D* may include qualitative as well as quantitative variables. The problem statement is the generic one: Estimate *S* that minimizes expected loss with respect to $p(S|E,D,H)$.

4.2.2 Loss Functions

As mentioned above, sometimes in a decision problem it is easier to express the decisionmaker's objective in terms of a loss function $L(S,S^*)$ rather than a utility function. Thus, the optimal decision becomes minimize the expected loss rather than maximize the expected utility. Selection of a suitable loss function then becomes a key issue.

In estimation problems there are three loss functions that are most commonly used.

(1) **Quadratic loss**: $L(S,S^*) = \|S-S^*\|^2$.

Minimization of the expected quadratic loss minimizes the prediction mean square error (MSE) and thus is often appropriate in function approximation problems. The optimal estimate under this loss function is the expected value of S with respect to the predictive distribution: $S^* = \int S\,p(S|E,D,H)dS$.

(2) **Absolute loss**: $L(S,S^*) = |S-S^*|$.

Minimization of the expected absolute loss minimizes the prediction mean absolute error and thus sometimes finds use when the predictive distribution has heavy tails. The optimal estimate under this loss function is the median value of S with respect to the predictive distribution: S^*=median(p($S|E,D,H$)).

(3) **0-1 loss**: $L(S,S^*) = 0$ if $S=S^*$ else 1.

Minimization of the expected 0-1 loss is useful when the predictive distribution has little or no probability mass at its expectation or the median. It is often used in finding point estimates as in parameter estimation problems. The optimal estimate under this loss function is the mode of the predictive distribution: S^*= argmax(p($S|E,D,H$)), i.e. the value of S that maximizes p($S|E,D,H$).

Throughout this work, a quadratic loss function will be used when an estimate for a prediction from a regression model is sought. This is because of the minimum-prediction-MSE property of the resultant estimator. (In the statistics literature, this property is

referred to as "maximum efficiency.") This implies that the variance of predictions made with this estimator will be minimized.

When point estimates for model parameters are sought, a 0-1 loss function will be used. This is because parameter estimation is based on the posterior distribution of the parameters $p(\theta|D,H)$ which often for the models posed – especially the nonparametric models – will be multimodal and symmetric with little probability mass at the expected value of the parameters. In such cases, using a quadratic loss, which results in the expected value for the estimate of θ, would yield an extremely improbable parameter estimate. In contrast, using a 0-1 loss function in parameter estimation yields the most probable parameters.

4.2.3 Axioms of Probability Theory

The uncertainty associated with the decision problem faced by the modeler is represented by the probability density function $p(S|E,D,H)$. However, rarely in function approximation or state estimation problems is the available information (knowledge and data) in the form of this pdf. Usually infomation is available about more fundamental relationships between variables and about the uncertainty in the measurements and variables. For example, usually it iss reasonably hypothesized that the measurement errors are independent and identically sampled from a normal distribution. It is necessary to use such basic information to decompose $p(S|E,D,H)$ in terms of more basic pdfs. This is done using the axioms of probability theory, shown in Section 2.1 for discrete propositions and shown here for pdfs of continuous variables S, D and H:

Product Rule: $p(S,D|H) = p(S|D,H)p(D|H) = p(D|S,H)p(S|H)$

Sum Rule: $1 = \int p(S|H)dS$.

Marginalization and Bayes' Theorem for continuous variables are derivable from these axioms and are useful in probabilistic modeling:

Marginalization: $p(D|H) = \int p(S,D|H)dS$.

Bayes' Theorem: $$p(S|D,H)=\frac{p(D|S,H)p(S|H)}{\int p(D|S,H)p(S|H)dS}=\frac{p(D|S,H)p(S|H)}{p(D|H)}$$

Together, these allow us to perform the necessary decomposition of p(*S*|*E*,*D*,*H*) into more fundamental pdfs, presented in Section 4.4. Marginalization is good for integrating out "nuisance variables" or for introducing new variables for which more information is available. The product rule allows one to decompose a joint pdf into more basic components, e.g., the "likelihood" P(*D*|*S*,*H*) and the "prior" p(*S*|*H*) in Bayes' Theorem. This is useful because usually more information is known about the conditional probability distributions and priors than is known directly about the joint pdf. Probably the most important use of Bayes' Theorem is in updating the prior probability p(*S*|*H*) given additional information *D*, thus yielding a "posterior" probability p(*S*|*D*,*H*). In this way a modeler's beliefs (represented probabilistically) can be updated accordingly as data or any other relevant information become available.

4.3 State Estimation

Within the primary problem of state estimation are the subproblems of model synthesis, parameter estimation, interval estimation, and hypothesis selection. In each of the applications described above, the underlying probability density functions imposed by *H* may depend on the past data *D*. Usually it is too cumbersome to explicitly apply *D* in estimating *S* each time a new *E* arrives. Instead, it is usually possible and almost always desirable to summarize the dependence of *S* and *E* induced from *D* in terms of models $f(x;\theta(D))$ with parameters θ, which are dictated by the hypothesis *H*. So, $p(\theta|D,H)$ is important because it allows us to take *D*, which implicitly includes the relationship between past sets of inputs and states, and concisely summarize this relationship in terms of θ. The state estimation problem can then be restated to explicitly account for the parameters:

State estimation problem:

Given evidence E, estimate the most probable state S of the process under the assumptions imposed by H and in light of data D summarized as θ. Therefore, we wish to find S^* that minimizes

$$L_E(S^*) \equiv \int L(S,S^*)p(S|E,D,H)dS = \int L(S,S^*)p(S|E,\theta,H)p(\theta|D,H)d\theta dS .$$

The subproblems of model synthesis, parameter estimation, interval estimation and hypothesis selection are motivated by examining the above equation. The derivation of the functions for the pdfs themselves is the task of model synthesis. Given the pdfs, pragmatic considerations usually dictate that we avoid solving the integral above (though, as described in Chapter 6, techniques exist to do so). Most often, θ is treated as if it were known exactly rather than being integrated out. This calls for a point estimate of θ, and it motivates the subproblem of parameter estimation. Furthermore, we often wish to summarize the uncertainty in the estimate for θ in terms of an interval rather than the full pdf. This motivates interval estimation. Finally, many different hypotheses may be derived from the same state of knowledge about the process. The problem of selecting the best hypothesis thus becomes important. In subsequent sections, the subproblems of model synthesis, parameter estimation, and hypothesis selection are formulated in a decision theoretic framework. Though briefly mentioned, interval estimation is outside the scope of this research.

4.3.1 Model Synthesis

Model synthesis in the probabilistic modeling paradigm involves the specification of not only the model structure relating $\tilde{y}$ to $\tilde{x}$, i.e., $\tilde{y}=f(\tilde{x};\theta(D))$, but also all other pdfs representing the inequality constraints and uncertainty associated with our assumed prior knowledge and the data sampling model. Section 4.4 illustrates the decomposition of the central state estimation pdf $p(S|E,D,H)$ into more basic pdfs involving the model parameters θ and the measurement error and data sampling models. Later Chapter 5

shows how these basic pdfs are derived from our prior knowledge (functional relations and constraints) and assumptions about the process. Of particular note is the important role that finite mixture distributions play in the process modeling methodology proposed in this research. Defined in Section 5.4, finite mixture distributions are a versatile means of representing knowledge such as multiple process operating regimes and can also be used as nonparametric estimators of the data density distribution. The full model synthesis procedure is demonstrated by the case studies in Chapter 7. Once the model synthesis problem is resolved by representation as pdfs, the other subproblems may be solved.

4.3.2 Parameter Estimation

Parameter estimation poses another decision problem to be solved. As such, it requires a suitable loss function to express our preferences for a parameter. Section 4.2.2 gave the rationale for using a 0-1 loss function in such a situation. Therefore, the parameter estimation problem is stated as follows:

Parameter Estimation Problem:

> Given data D and subject to the assumptions imposed by hypothesis H, select the optimal θ under 0-1 loss. Thus, find θ that maximizes the posterior probability density function $p(\theta|D,H)$.

The maximization of the posterior distribution of θ is usually accomplished in practice using constrained or unconstrained nonlinear programming (optimization) methods. Chapter 6 illustrates how this is done and presents a more efficient means of solving the problem in special cases using expectation-maximization (EM). EM is an algorithm that exploits the special form of the problem resulting from using finite mixture distributions. We show how it is particularly effictive in solving process modeling problems.

4.3.3 Certainty of Point Estimates

State estimation and parameter estimation are called "point estimation" problems in statistics, and as such, it is often important to summarize the certainty or confidence in these estimates. Of course the distributions $p(S|E,D,H)$ and $p(\theta|D,H)$ summarize completely our certainty in S and θ given E, D and H. But, sometimes it is convenient to report the certainties in terms of intervals around the estimates within which the true values would fall a given percentage of the time. In other words, we would like to specify $\Delta\theta$ for a given probability of the true value of being within $\pm\Delta\theta$ of our estimate. These interval estimates are analogous, though not identical, to the "confidence intervals" of classical statistics. The determination of these regions for an arbitrary pdf can be quite complex, necessitating techniques such as Monte Carlo simulation. Suffice it to say that such "credible sets" or "highest probability density regions," as they are sometimes called, can be determined but are not the focus of this research.

4.3.4 Hypothesis Selection

Given the fact that we can never know for certain the true state of nature, we must rely on our empirical observations to validate any hypothesis H we pose about the process. We can never truly gauge how closely any given H is to coinciding exactly with nature, but by amassing enough empirical data D, we can become more certain in the H we select. Consequently, the pragmatic objective is to compose a hypothesis or assumed state of knowledge H that is close enough to reality to be useful in inferring things about the true state of nature. Given a single data set D, the best we can do is to select an individual H that is more plausible than any other alternative we pose. Naturally, this is done by using the probability of the hypothesis given the data $P(H|D)$. The hypothesis with the greatest $P(H|D)$ is the one that coincides most closely to our given set of empirical observations about nature.

Again, hypothesis selection is a decision problem, and the criterion for evaluating the decision is dictated by our preferences in the form of a suitable loss function. For this

decision, we choose a 0-1 loss function. Therefore, given the data D, we wish to choose the most probable hypothesis H_i: i.e., from a set of hypotheses $\mathcal{H}$, select the hypothesis $H_i \in \mathcal{H}$ that has greatest $P(H_i|D)$. So, to compare two hypotheses H_1 and H_2, we can assess their odds $O(H_1,H_2) \equiv P(H_1|D)/P(H_2|D)$. If $O(H_1,H_2)>1$, then H_1 is preferred to H_2; if $O(H_1,H_2)<1$, H_2 is preferred; and if $O(H_1,H_2)=1$, the hypotheses explain the data equally well and other attributes such as ease of application will dictate the preference for one hypothesis over the other.

By Bayes' Theorem, $P(H_i|D)=p(D|H_i)P(H_i)/p(D)$ and $O(H_1,H_2) = p(D|H_1)\,P(H_1) / [p(D|H_2)P(H_2)]$. If the prior probability of each hypothesis is equal to that of any other, then $P(H_1)=P(H_2)$; and thus, $O(H_1,H_2)=p(D|H_1)/p(D|H_2)$. This ratio is referred to as the "Bayes factor." Consequently, the key pdf in hypothesis selection is $p(D|H_i)$, the marginal probability distribution of the data given the hypothesis. So the hypothesis selection problem can be stated as follows:

Hypothesis selection problem:

> Given data D and a set of hypotheses $\mathcal{H}$, select the hypothesis $H_i \in \mathcal{H}$ that maximizes the marginal distribution of the data, $p(D|H_i)$.

One drawback to the Bayes' factor approach to hypothesis selection is that it is a relative criterion: hypotheses are compared to one another and in the context of a single data set. This is not a major drawback but it does limit the approach's applicability. Bayes' factors are inappropriate for situations such as comparing models intended for the same process but derived from different datasets or for comparing related models from different processes, e.g. to assess which models of separate process units of a single large process are the weakest so that future work might focus on strengthening them.

An alternative to this relative criterion is the absolute criterion of prediction error that was discussed in Section 3.5 for the function-oriented paradigm. In the probabilistic paradigm, the Mean Integrated Squared Error (MISE) can be defined as the quadratic loss $L(S,S^*) =$

$\|S-S^*\|^2$ averaged over all possible states, evidence and data sets. Therefore, MISE is given by the following formula:

$$\text{MISE}=\int\left\|S-S^*(E,D)\right\|^2 \text{p}(S|E,D,H)dS\,\text{p}(E|H)dE\,\text{p}(D|H)dD,$$

where the dependence of the estimate S^* on the evidence and data has been made explicit. Note that this formula for MISE differs slightly from that of Section 3.5: there is additional averaging over the assumed predictive distribution p(S|E,D,H). Recognizing that, under quadratic loss, S^* is the expectation of p(S|E,D,H), we see that MISE is actually the variance of p(S|E,D,H) averaged over all possible evidence and data. In other words, unlike the function-oriented version of MISE (Chapter 3), which involves the unknown true model, the probabilistic version is based on the variance of the known predictive distribution. The variance can be found in simple cases by analytically calculating it or by estimating it in more complicated cases by either taking a second-order approximation to p(S|E,D,H) around its mean – i.e. approximating p(S|E,D,H) as a normal (Gaussian) distribution – or by directly sampling p(S|E,D,H). Calculation of the variance and the additional marginalization over E and D can be done by sampling using the Markov Chain Monte Carlo (MCMC) methods presented in Chapter 6. However, MCMC methods are computationally intensive, and in practice we settle for approximating MISE using a Gaussian for p(S|E,D,H) or using the criteria given in Chapter 3.

Maximization of p(D|H) is a proven means of selecting one hypothesis over another. MacKay (1992b) presents simulations to illustrate that using the maximum of p(D|H) to choose prior distributions, hyperparameters (parameters of the prior distributions), and models is superior to evaluating mean prediction squared error on a test set. The maximum of the p(D|H) was more distinct, giving less ambiguous choices. These results are important because they represent a means of comparing the predictive capabilities of multiple models without using separate test data sets or requiring cross-validation, which have the disadvantage of requiring lots of data. Sibisi (1989) compared P(D|H) with generalized cross-validation (GCV) in selecting optimal hyperparameters. He found that plotting log[P(D|H)] as a function of the hyperparameter provided a distinct peak yielding

a clear choice for the constant. On the other hand, the prediction error estimated using GCV gave a broad flat peak with respect to the hyperparameter. The hyperparameter ranged over an order of magnitude across the peak. Therefore, the hypothesis selection using GCV was more ambiguous and error-prone.

Many different hypotheses can be derived from the same process knowledge. Ideally, we would like to determine which is the single optimal hypothesis that can be drawn from a given knowledge base. However for a typical modeling case, the problem of alternately imposing and relaxing all possible constraints and assumptions is combinatorial. Thus, it is infeasible to exhaustively search the hypothesis space (i.e., the problem of hypothesis selection is NP-complete). So, it is important to discover heuristics that aid in finding a good hypothesis, if not the optimal one. In Chapter 7, we discuss such heuristics based on the diagnostic information obtained by applying the probabilistic approach we present in this and following chapters.

4.4 Probability Density Functions for State Estimation

As a prelude to state estimation and its subproblems of model synthesis, parameter estimation, and hypothesis selection, this section presents the decompostion of $p(S|E,D,H)$ into its component pdfs. These more basic pdfs are more directly related to the types of knowledge available in process modeling. Later, in Chapter 5, we will see how our prior knowledge is actually transformed into the corresponding pdfs.

4.4.1 Predictive Distribution

The probability density function $p(S|E,D,H)$ is referred to as the predictive distribution in Bayesian regression, and we will use that terminology here. As shown above in the examples of Section 4.2.1, this pdf is readily expressed in terms of the parameters θ. The expression of $p(S|E,D,H)$ in terms θ is motivated by the fact that the basic component pdfs, which are necessary to include our full knowledge about dependence of $\tilde{y}$ on $\tilde{x}$ and

true states $\{\tilde{x}, \tilde{y}\}$ on the data $\{x,y\}$, will have parameters that are unknown. The decomposition is executed by applying the sum rule of probability theory:

$$p(S|E,D,H) = \int p(S|E,\theta,D,H)\,p(\theta|E,D,H)\,d\theta = \int p(S|E,\theta,H)\,p(\theta|D,H)\,d\theta \quad (4.1)$$

The equation of $p(S|E,\theta,D,H)$ to $p(S|E,\theta,H)$ is valid because we assume that the parameters θ summarize all information about the process state S that is contained in the data D. Also, $p(\theta|E,D,H)$ equals $p(\theta|D,H)$ because, as stated above, θ is independent of the evidence E. We call $p(S|E,\theta,H)$ the "evidential dependence" model because it captures the state variables' dependence on the evidence. Section 4.4.2 discusses this model in more detail. However, note that $p(S|E,\theta,H)$ is usually a simple function capturing the relation $\tilde{y} = f(\tilde{x};\theta)$, and that the focus of the estimation problem then is placed on $p(\theta|D,H)$. This latter pdf, which is well-known in Bayesian regression, is called the "posterior distribution of θ" because it represents our knowledge about θ after receipt of the data D, i.e., posterior to the accounting for information in D. The posterior distribution of θ and its components are described in Section 4.4.3.

4.4.2 Evidential Dependence Model

The pdf $p(S|E,\theta,H)$ is the function that relates state predictions to the available evidence, given the parameters. In other words, it is the predictive model. For regression in the probabilistic paradigm $p(S|E,\theta,H)$ accounts for uncertainty in the model structure itself, unlike in the function-oriented paradigm where a model $\tilde{y} = f(\tilde{x};\theta)$ is posed with assumed certainty. For example, if unmeasured intermediates z exist, then $p(S|E,\theta,H) = p(\tilde{y}|\tilde{x},\theta,H) = \int p(\tilde{y}|\tilde{x},z,\theta,H)\,p(z|\tilde{x},H)\,dz$. Also, either of these pdfs may be represented parametrically or nonparametrically, thus yielding semiparametric models as in the function-oriented approach. As we shall see through the examples, the probabilistic paradigm produces models with a more rational motivation than in the function-oriented paradigm.

4.4.3 Posterior, Likelihood, and Prior of the Parameters

The posterior distribution of θ, p(θ|*D*,*H*), is the probability density function of the parameters given the data (under hypothesis *H*). It is the key distribution in Bayesian parameter estimation. Rarely would we be able to specify this pdf without resorting to knowledge about the lower level relationships: first, between the data and the parameters; and second, between the parameters themselves prior to receiving the data. These more basic pdfs are injected into the problem by applying Bayes' Theorem:

$$p(\theta|D,H)=\frac{p(D|\theta,H)\,p(\theta|H)}{\int p(D|\theta,H)\,p(\theta|H)d\theta}=\frac{p(D|\theta,H)\,p(\theta|H)}{p(D|H)}. \qquad (4.2)$$

The pdfs in the numerator of Equation 4.2 are essential for solving the parameter estimation problem. Conversely, the pdf in the denominator is independent of θ and is not relevant to parameter estimation. We discuss its usefulness in the next section.

The first pdf in the numerator, p(*D*|θ,*H*) is known as the "likelihood of θ." It relates the data to the model parameters. However, what we usually know or can more directly specify is (1) how the true state variables are related to the parameters, (2) how each measurement is related to the true variables, via the assumed form of the measurement errors, and (3) how each measurement is then related to the other measurements. To relate the data *D*={*X*,*Y*} to the parameters, we must introduce the true variables $\tilde{X}$ and $\tilde{Y}$, corresponding to the measurements, and the error matrices, $\mathcal{E}_x \equiv X-\tilde{X}$ and $\mathcal{E}_y \equiv Y-\tilde{Y}$. The likelihood is then expressed in terms of more basic pdfs by injecting the true variables and errors using the sum rule (marginalization):

$$p(D|\theta,H)=\int p\left(X,Y|\tilde{X},\tilde{Y},\theta,H\right)p\left(\tilde{X},\tilde{Y}|\theta,H\right)d\tilde{X}d\tilde{Y}$$

$$p(D|\theta,H)=\int p\left(X,Y|\tilde{X},\tilde{Y},\mathcal{E}_x,\mathcal{E}_y,\theta,H\right)p\left(\mathcal{E}_x,\mathcal{E}_y|\tilde{X},\tilde{Y},\theta,H\right)d\mathcal{E}_x d\mathcal{E}_y\, p\left(\tilde{X},\tilde{Y}|\theta,H\right)d\tilde{X}d\tilde{Y}$$

where p(*X*,*Y*|$\tilde{X}$, $\tilde{Y}$ θ,*H*) is the "observation model" relating the data to themselves and the true values; p(*X*,*Y*|$\tilde{X}$, $\tilde{Y}$, $\mathcal{E}_x$,$\mathcal{E}_y$,θ,*H*) is the "measurement model" relating the data to themselves, the true values and the errors; p($\mathcal{E}_x$,$\mathcal{E}_y$|$\tilde{X}$, $\tilde{Y}$,θ,*H*) is the "error model" relating

the errors to themselves and the true values; and $p(\tilde{X}, \tilde{Y} | \theta, H)$ is the "state dependence model" relating the true state variables to themselves and the parameters. More is said about these models in Sections 4.4.5 and 4.4.6.

The other pdf in the numerator of Equation 4.2, $p(\theta|H)$ is known as the "prior distribution of θ" and must be specifed *a priori* by the modeler. As will be shown in Chapter 5, the prior must be derived from appeals to the modeler's fundamental knowledge, beliefs, preferences and assumptions about the specific application. This may be done objectively by applying principles such as invariance of the probabilities under transformation of the variables. Chapter 5 discusses these ideas and presents heuristics to assist in eliciting pdfs in typical process modeling applications.

4.4.4 Marginal pdf of the Data

The pdf $p(D|H)$ in the denominator of Equation 4.2 is the marginal distribution of the data. In parameter estimation, it only serves as a normalizing constant because it is independent of θ. However, $p(D|H)$ is central in solving the hypothesis selection problem as seen above in Section 4.3.4, where the Bayes' factors for model selection are expressed in terms of $p(D|H)$.

This pdf is computed by integrating the product in the numerator of Equation 4.2 over all parameter values:

$$p(D|H) = \int p(D|\theta, H)\, p(\theta|H)\, d\theta \,. \qquad (4.3)$$

Methods to compute $p(D|H)$ are shown in Chapter 6.

4.4.5 Observation, Measurement and Error Models

Section 4.4.3 showed that the likelihood of the parameters $p(D|\theta,H)$ can be expressed in terms of the observation model $p(X,Y|\tilde{X}, \tilde{Y}\, \theta,H)$, which in turn can be decomposed by the product and sum rules into the measurement model $p(X,Y|\tilde{X}, \tilde{Y}, \mathcal{E}_x, \mathcal{E}_y, \theta, H)$ and the error

model $p(\mathcal{E}_x,\mathcal{E}_y|\tilde{X},\tilde{Y},\theta,H)$. By definition, $\mathcal{E}_x \equiv X - \tilde{X}$; so X is conditionally independent of Y, $\tilde{Y}$, $\mathcal{E}_y$ and θ given $\tilde{X}$ and $\mathcal{E}_x$. In other words, $\tilde{X}$ and $\mathcal{E}_x$ provide all the information available about X. Analogously, Y is conditionally independent of X, $\tilde{X}$, $\mathcal{E}_x$ and θ given $\tilde{Y}$ and $\mathcal{E}_y$. Therefore, the measurement model becomes:

$$p\left(X,Y|\tilde{X},\tilde{Y},\mathcal{E}_x,\mathcal{E}_y,\theta,H\right) = p\left(Y|\tilde{Y},\mathcal{E}_y,H\right)p\left(X|\tilde{X},\mathcal{E}_x,H\right) = \delta\left(Y,\tilde{Y}+\mathcal{E}_y\right)\delta\left(X,\tilde{X}+\mathcal{E}_x\right) \quad (4.4)$$

where the delta pdf $\delta(u,v)$ denotes certainty, effectively substituting its second argument for its first upon integration, i.e. $\int f(u)\delta(u,v)du = f(v)$, and of course, as is true for all proper pdfs, $\int \delta(u,v)du = 1$.

The error model can also be simplified due to independence considerations. First, neither $\mathcal{E}_x$ nor $\mathcal{E}_y$ are dependent on θ, so

$$p(\mathcal{E}_x,\mathcal{E}_y|\tilde{X},\tilde{Y},\theta,H)=p(\mathcal{E}_x,\mathcal{E}_y|\tilde{X},\tilde{Y},H). \quad (4.5)$$

Also, in most cases it is convenient to assume that the errors are neither correlated with themselves nor the state variables, thus

$$p(\mathcal{E}_x,\mathcal{E}_y|\tilde{X},\tilde{Y},H)=p(\mathcal{E}_x|H)p(\mathcal{E}_y|H). \quad (4.6)$$

Note, though, that this is not always the case. The fact that correlated errors are naturally handled by this formalism is part of its power and flexibility.

4.4.6 State Dependence Model

The final component of the likelihood of the parameters is the state dependence model $p(\tilde{X},\tilde{Y}|\theta,H)$. This pdf describes the relationship between all of the true variables corresponding to the measured data matrix. Invoking the product rule factors this model into two more basic pdfs:

$$p(\tilde{X},\tilde{Y}|\theta,H)=p(\tilde{Y}|\tilde{X},\theta,H)p(\tilde{X}|\theta,H). \quad (4.7)$$

The first pdf on the right-hand side captures the dependence of the true state variables and the parameters. The second pdf describes the density distribution of the independent (input or exogenous) variables and is expressed more succinctly as $p(\tilde{X}|H)$, because $\tilde{X}$ and θ are independent of each other. In essence, $p(\tilde{X}|H)$ captures the process operating policy represented by the data.

Recursively applying the product rule allows us to express $p(\tilde{X},\tilde{Y}|\theta,H)$ in terms of the individual $\tilde{x}_i$ and $\tilde{y}_i$:

$$p(\tilde{X},\tilde{Y}|\theta,H)=\prod_{i=1}^{N} p\left(\tilde{y}_i|\tilde{x}_i,\tilde{Y}_{\{<i\}},\theta,H\right)p\left(\tilde{x}_i|\tilde{X}_{\{<i\}},H\right). \quad (4.8)$$

In the case that each observation is independent of the others (as is usually assumed in steady-state modeling problems), the state dependence model becomes a product of the N individual pdfs $p(\tilde{y}_i|\tilde{x}_i,\theta,H)$: $p(\tilde{X},\tilde{Y}|\theta,H)=\prod_{i=1}^{N} p\left(\tilde{y}_i|\tilde{x}_i,\theta,H\right)p\left(\tilde{x}_i|H\right)$. Also, assuming that the observations are identically distributed, we see that $p(\tilde{y}_i|\tilde{x}_i,\theta,H)=p(\tilde{y}|\tilde{x},\theta,H)$, $\forall i=1,2,...,N$. So, under these special circumstances, the evidential dependence model $p(\tilde{y}|\tilde{x},\theta,H)$ is at the core of the state dependence model $p(\tilde{X},\tilde{Y}|\theta,H)$.

4.4.7 Example: Ordinary Regression

This example applies the probabilistic formulation of the state estimation problem to the familiar setting of "ordinary" regression. By "ordinary," it is meant that (1) the true variables x are measured exactly; (2) errors e_y exist in the dependent variables y, i.e. $y=\tilde{y}+e_y$; (3) e_y are independent of each other, x and y; and (4) $e_y \sim N(0,\sigma^2)$, i.e. e_y are identically distributed as zero-mean normals with known variance σ^2.

In this context, the hypothesis H contains all of the above knowledge about the problem plus an assumed parametric model $\tilde{y}=f(\tilde{x};\theta)$ and prior distributions on $\tilde{x}$ and θ. For now we shall assume these priors to be uniform distributions over finite domains $\mathcal{D}_x$ and

$\mathcal{D}_\theta$, for x and θ, respectively (Chapter 5 discusses alternatives). The data $D=\{X,Y\}$ are again N tuples $\{x_i,y_i\}$, $i=1,2,...,N$. The evidence $E=x_p$ is the independent value at which we wish to estimate the state, i.e. make the prediction. So, our estimate $S^*=y_p$ will attempt to minimize the expectation of the quadratic loss $L(S,S^*)=(y_p-\tilde{y}(x_p))^2$ of the true state $S=\tilde{y}(x_p)$. The estimate that achieves this is $y_p=\int \tilde{y}\, p(\tilde{y}|\tilde{x}=x_p,X,Y,H)d\tilde{y}$, the expected value of the predictive distribution.

The predictive distribution is decomposed according to the steps presented above in this chapter. The component pdfs are listed below.

Evidential dependence model: H assumes certainty in the relation $\tilde{y}=f(\tilde{x};\theta)$. In other words it is an imposed equality constraint. So, the evidential dependence model is represented as a delta function imposing this constraint:

$$p(S|E,\theta,H)\equiv p(\tilde{y}|x=x_p,\theta,H)=\delta(\tilde{y},f(x_p,\theta)).$$

Prior distribution of θ: H assumes equiprobable values for the scalar parameter θ over the finite domain $\mathcal{D}_\theta$. So, the prior pdf for θ is a constant: $p(\theta|H)=c$, such that

$$\int_{\theta\in D_\theta} p(\theta|H)d\theta=1;\ \text{i.e.}\ \int_{\theta\in D_\theta} d\theta=\frac{1}{c}.$$

Measurement model: The x values are measured exactly, so $\mathcal{E}_x$ can be dropped from the expression. Also, the observations are independent, so this model is a series product of delta functions: $p(X,Y|\tilde{X},\tilde{Y},\mathcal{E}_x,\mathcal{E}_y,\theta,H)=\delta(Y,\tilde{Y}+\mathcal{E}_y)\delta(X,\tilde{X})=\prod_{i=1}^{N}\delta(y_i,\tilde{y}_i+e_{yi})\delta(x_i,\tilde{x}_i)$.

Error model: The error model involves only the errors in y and can be expressed as a product of the N identical normal distributions because the errors and observations are independent. The normal distribution with mean μ and variance σ^2 is expressed

functionally as $\phi(u;\mu,\sigma^2)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(u-\mu)^2}{2\sigma^2}\right)$. Therefore, the error model follows:

$$p(\mathcal{E}_y|\tilde{X},\tilde{Y},H)=p(\mathcal{E}_y|H)=\prod_{i=1}^{N}p(e_{yi}|H)=\prod_{i=1}^{N}\phi(e_{yi};0,\sigma^2)=(2\pi\sigma^2)^{-N/2}\exp\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{e_{yi}}{\sigma}\right)^2\right].$$

Observation model: This model is formed by integrating the product of the measurement and error models over all values for the errors. The result is a substitution of the data and true values into the error model pdf:

$$p(X,Y|\tilde{X},\tilde{Y},\theta,H)=\int(2\pi\sigma^2)^{-N/2}\exp\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{e_{yi}}{\sigma}\right)^2\right]\prod_{i=1}^{N}\delta(y_i,\tilde{y}_i+e_{yi})\delta(x_i,\tilde{x}_i)d\mathcal{E}_y$$

$$p(X,Y|\tilde{X},\tilde{Y},\theta,H)=(2\pi\sigma^2)^{-N/2}\exp\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i-\tilde{y}_i}{\sigma}\right)^2\right]\prod_{i=1}^{N}\delta(x_i,\tilde{x}_i)$$

State dependence model: H postulates independent and identical distribution of the observations along with assumed certainty in the relation $\tilde{y}=f(\tilde{x};\theta)$. This results in a series product of delta functions for the state dependence model. Also, because the $\tilde{x}$ are assumed uniform distributions over finite domain $\mathcal{D}_x$, a constant b serves as the pdf $p(\tilde{x}|H)=b$, same as was done for $p(\theta|H)$ above. So the state dependence model is as follows: $p(\tilde{X},\tilde{Y}|\theta,H)=\prod_{i=1}^{N}p(\tilde{y}_i|\tilde{x}_i,\theta,H)p(\tilde{x}_i|H)=b^N\prod_{i=1}^{N}\delta(\tilde{y}_i,f(\tilde{x}_i;\theta))$.

Likelihood of θ: The observation model and the state dependence model are combined to form the likelihood of θ:

$$p(D|\theta,H)=\int p(X,Y|\tilde{X},\tilde{Y},\theta,H)p(\tilde{X},\tilde{Y}|\theta,H)d\tilde{X}d\tilde{Y}$$

$$p(D|\theta,H)=(2\pi\sigma^2)^{-N/2}b^N\int\exp\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i-\tilde{y}_i}{\sigma}\right)^2\right]\prod_{i=1}^{N}\delta(x_i,\tilde{x}_i)\delta(\tilde{y}_i,f(\tilde{x}_i;\theta))d\tilde{X}d\tilde{Y}$$

$$p(D|\theta,H)=(2\pi\sigma^2b^2)^{-N/2}\exp\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i-f(x_i;\theta)}{\sigma}\right)^2\right].$$

Marginal distribution of *D*: p($D|H$) is computed by integrating the product of the likelihood and the prior of θ over all values of θ. Unfortunately, if the model $f(\tilde{x};\theta)$ is nonlinear in θ, this integration can be complex even for this simple regression case. Fortunately, in parameter estimation, p($D|H$) is not needed. When it is, the methods discussed in Chapter 6 should be applied.

In Bayesian estimation, p($D|H$) simply serves as a normalizing constant:

$$p(D|H)=c(2\pi\sigma^2 b^2)^{-N/2}/C, \text{ where } C \equiv \int \exp\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i - f(x_i;\theta)}{\sigma}\right)^2\right]d\theta .$$

Posterior distribution of θ: Finally, the component pdfs are combined in Bayes' Theorem to yield the posterior of θ, p($\theta|X,Y,H$):

$$p(\theta|X,Y,H) = \frac{p(X,Y|\theta,H)\,p(\theta|H)}{p(X,Y|H)} = C\exp\left[-\frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i - f(x_i;\theta)}{\sigma}\right)^2\right].$$

Parameter Estimation: One way to solve the state estimation problem is by settling for a point estimate θ^* of the parameters θ. Therefore, applying a 0-1 loss function in the parameter estimation problem, θ^* is found by maximizing the posterior p($\theta|X,Y,H$) or equivalently minimizing the negative of log[p($\theta|X,Y,H$)]:

$$-\log[p(\theta|X,Y,H)]=\frac{1}{2}\sum_{i=1}^{N}\left(\frac{y_i - f(x_i;\theta)}{\sigma}\right)^2 - \log(C) = \frac{1}{2\sigma^2}F(\theta) - \log(C),$$

where $F(\theta)=\sum_{i=1}^{N}\left[y_i - f(x_i;\theta)\right]^2$.

Thus, neglecting the constant terms of -log[p($\theta|X,Y,H$)], the parameter estimation problem becomes the ordinary least-squares problem traditionally solved in simple regression:

$$\text{Find } \theta=\theta^* \text{ that minimizes } F(\theta)=\sum_{i=1}^{N}\left[y_i - f(x_i;\theta)\right]^2 .$$

State Estimation: Finally, assuming certainty about the point estimate θ^*, the prediction is simply $y_p=f(x_p;\theta^*)$:

$$y_p = \int \tilde{y}\delta\left(\tilde{y}, f\left(x_p;\theta\right)\right)\delta\left(\theta,\theta^*\right)d\theta d\tilde{y} = \int \tilde{y}\delta\left(\tilde{y}, f\left(x_p;\theta^*\right)\right)d\tilde{y} = f\left(x_p;\theta^*\right).$$

4.4.8 Summary

Figure 4.1 summarizes the decomposition of p(*S*|*E*,*D*,*H*) in the decision theoretic problem formulation for state estimation. This decomposition is based on the axioms of probability theory and the focus of our state estimation decision problem. It represents the fact that the probabilistic paradigm for process modeling provides a formal, consistent, and information leak-proof methodology for combining our prior knowledge with data-derived information.

From the ordinary regression example presented in this chapter, it is clear that the generic formulation reduces to common techniques under certain specializing assumptions. The procedure to arrive at an ordinary least-squares problem was tedious, but if other assumptions had been applied such as the existence of outliers or a non-uniform prior on the parameters, different parameter estimation problems would have arisen thus justifying the methodical procedure. In defining this formal methodology, the probabilistic paradigm ensures coherence and dictates the ultimate model and objective function in the face of an extremely wide variety of case-specific properties of the data and process knowledge. This is a distinct advantage over many function-oriented approaches that start with the model $\tilde{y}=f(\tilde{x};\theta)$ and the ordinary least-squares objective function $F(\theta)$ and add extensions to accomodate outliers or functional preferences expressed through the parameters, such as smoothing penalty functions (i.e. regularization).

Chapter 5 will present examples of and guidelines for representing process knowledge in the form of the pdfs shown in Figure 4.1. In doing so, the natural roles of the types of relations and assumptions about the data will become evident. Moreover, these roles dictated by probability theory serve as the theoretical basis for the function-oriented

approach presented in Chapter 3. The connections to that approach are made explicit in Section 5.5. Consequently, the probabilistic formulation provides an explanation for the success of our empirically motivated methodology. For further verification of the practical usefulness of these ideas, Chapters 6 and 7 address the pragmatic issues of applying the probabilistic modeling methodology to real-world engineering problems.

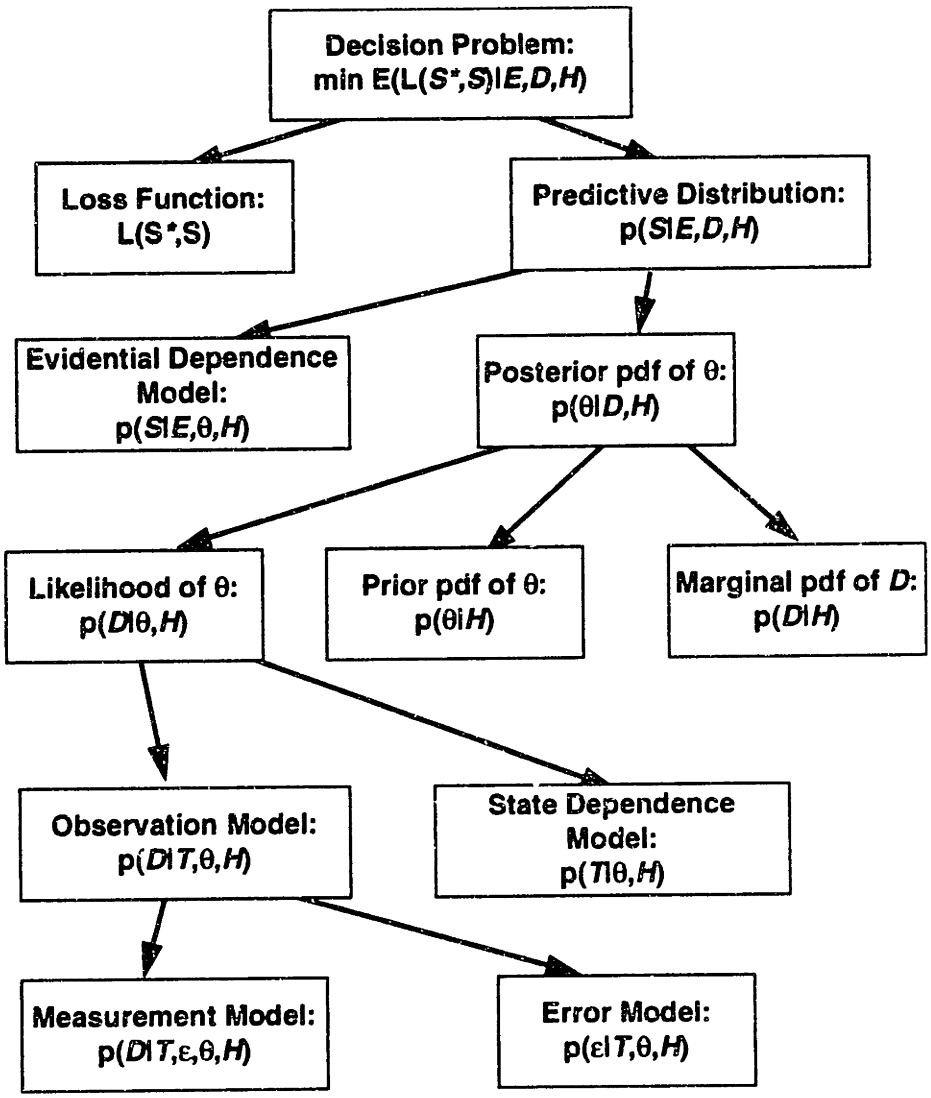


Figure 4.1: The probability density functions of state estimation.

Chapter 5

Knowledge Representation

The related problems of state estimation, model synthesis, parameter estimation, and hypothesis selection involve probability density functions (pdfs) $p(S|E,\theta,H)$, $p(D|\theta,H)$, $p(\theta|D,H)$, and $p(D|H)$. However, rarely will the process knowledge be in the form of these distributions, and often it is not obvious how to transform the known constraints and relations between process variables into pdfs. Fortunately, though, these pdfs are related to each other and to simpler pdfs relating subsets of the variables, data and knowledge. Explained in Chapter 4, the path leading from one such pdf to the target pdfs usually involves a combination of applying Bayes' Theorem, the Sum Rule, and the marginalization of distributions. Still often along this path, some component pdfs arise that are unknown or so complex that they must be treated by approximations driven by pragmatic considerations.

This chapter investigates the derivation of these pdfs. Moreover, we provide useful heuristics that allow an engineer to start from a given process knowledge base and proceed systematically through the pdfs to the resulting solutions of the synthesis, estimation and selection problems. This path from knowledge to pdfs and from pdfs to solutions is one regulated by probability theory but guided by pragmatic issues that necessarily require approximations to be imposed. As we chart this course in this and the remaining chapters, we duly note the approximating assumptions and, where possible, their ultimate impact on the final solutions. Along the way, examples of specific problems will be given in an attempt not only to clarify the points but to emphasize the close connection between the subproblems of model synthesis and the application of the model in state estimation.

This chapter also makes explicit the assumptions that must be imposed to traverse from the formal probablistic methodology to the empirically motivated function-oriented approach presented in Chapter 3. Traditionally, the imposition of implicit, often contradictory, assumptions and the application of *ad hoc* methods have splintered the closely related subproblems of model synthesis and state estimation. This is especially true of the many function-oriented approaches to process modeling. Hopefully, the treatment and examples given here will help to emphasize the uniform management of knowledge and data that is the reward of applying probability theory.

5.1 Probability Density Functions

The probability density functions in this work are considered "subjective" as opposed to the "objective" pdfs of classical frequentist statistics. The main difference is that objective pdfs for events are rationalized on a frequency of occurrence basis while subjective pdfs do not necessarily appeal to such sampling arguments to rationalize their assignment. That is not to say that subjective pdfs are necessarily arbitrary. Rather it means that they are applicable to any belief statements that the modeler makes about arbitrary propositions instead of being limited to only those propositions pertaining to the frequency of an event.

The assignment of pdfs to describe beliefs is an open issue. Here we consider some of the ways in which these assignments can be made and give examples in the domain of process modeling. The literature is rich with references addressing this topic. Some of note that ascribe to the subjectivist interpretation of probabilities are Berger (1985) and Bernardo and Smith (1994).

The probabilities we employ are subjective in the sense that they incorporate the personal knowledge and beliefs of the process modeler. Despite this fact, it is desirable that the probabilities still be as objective as possible: injecting what is known but not including any other bias. The probabilistic paradigm guards against the unjustified inclusion of an inductive bias in several ways. One is by forcing the modeler to specify prior knowledge

explicitly in the form of probability density functions for which the modeler must provide a rationale. In this way, other modelers can readily verify what information has been included along with the data in model synthesis. Another is by providing a means of quantitatively assessing the validity of the biases by allowing $p(D|H)$ to be calculated and compared with that of other candidate hypotheses. Ideally, only hypotheses that give high probability of the data will be adopted.

In this section, we address the former tact. We discuss a means of deriving pdfs that are defensible with respect to their objectivity aside from the knowledge injected by the modeler. Obviously, the modeler is responsible for justifying his or her own prior knowledge so introduced. The methods discussed here ensure that once the justifiable knowledge has been decided upon, its inclusion into model synthesis cannot be contaminated by any additional implicit biases.

5.1.1 Canonical Distributions

The canonical distributions of probability theory guarantee objectivity while injecting prior knowledge. These distributions belong to the exponential family of pdfs and arise naturally as the uniquely coherent pdfs that capture the available prior knowledge assuming that specific desirata are satisfied. Bernardo and Smith (1994) show that, separately applied, the desirata give rise to the exponential family of pdfs. The desirata considered are (1) invariance of probability assignment with respect to coordinate system transformations and (2) minimum information (maximum entropy) subject to constraints on the moments of functions. Starting from either of these criteria, a derivation of each member of the exponential family of pdfs is possible. The result is that each of these pdfs is the most noncommital pdf possible that still includes the specific knowledge, i. e. constraints, injected while maintaining the desired property.

Table 5.1 lists several of the continuous pdfs in the exponential family along with the requisite desirata and constraints from which each pdf was derived. The modeling of

continuous variables relies most heavily only on these pdfs, which include the uniform, exponential, normal, gamma and delta distributions. For many problems in process modeling, these distributions or a mixture of them are all that are needed.

Canonical distribution	**Formula of pdf**	**Expectation**	**Constraints/Desirata**
Exponential, $\mathrm{Exp}(u;\theta)$	$p(u\|\theta)=\exp(-u/\theta)/\theta$	$\theta=\mu$	Constraints: $u\geq 0$, $E[u\|H]=\mu\geq 0$. Invariance: $P(u\in D_1)=P(u\in D_2)$, D_1 same volume and distance from origin as D_2.
Truncated Exponential, $\mathrm{Texp}(u;\mu,a,b)$	$p(u\|\theta)=c\exp(-ku)$, where $c\int_a^b e^{-ku}du=1;\ c\int_a^b ue^{-ku}du=\mu$	μ	Constraints: $a\leq u\leq b$, $E[u\|H]=\mu\geq 0$.
Uniform, $U(u;\theta)$	$p(u\|\theta)=\theta\equiv 1\Big/\int_{u\in D_u} du$	centroid of D_u	Invariance: $P(u\in D_1)=P(u\in D_2)$, D_1 same volume as D_2.
Normal, $\phi(u;\mu,\sigma^2)$	$p(u\|\theta)=\dfrac{\exp\left[-\frac{1}{2}(u-\mu)^2/\sigma^2\right]}{\sigma\sqrt{2\pi}}$	μ	Constraints: $E[u\|H]=\mu$ and $E[u^2\|H]=\mu^2+\sigma^2$ Invariance: Spherical symmetry about μ
Gamma, $\mathrm{Ga}(u;\mu,\gamma)$	$p(u\|\theta)=\dfrac{\mu^\gamma}{\Gamma(\gamma)}u^{\gamma-1}e^{-\mu u}$	μ	Constraints: $E[u\|H]=\mu$, $E[\log u\|H]=\gamma$.
Delta, $\delta(u,\theta)$	$p(u\|\theta)=\delta(u,\theta)$, where $\int f(u)\delta(u,\theta)du\equiv f(\theta)$	θ	Constraints: $u=\theta$ Invariance: Certainty

Table 5.1: Canonical probability density functions

Rather than present the derivations of these pdfs, we merely list them and later give examples of their applicability in process modeling. Bernardo and Smith (1994) provide a complete description of and further references presenting the many ways in which the canonical distributions are motivated and derived. Furthermore, the Principle of Maximum Entropy, its use in deriving discrete and continuous probability distributions, and its broad applicability are fully discussed in Kapur and Kesavan (1992).

Finally, it is important to recognize that when the quantity u, whose pdf is being elicited, is actually assigned values by some sampling (repeated draw) process, all of the above distributions can be derived directly from sampling arguments. When justified, sampling arguments are an objective means of eliciting a pdf. In particular, when inducing a pdf for the distribution of many data-related quantities, such as measurement errors, it is appropriate to rely upon what one knows about the stochastic properties of the data-generation process to guide in the selection of a pdf. For example, appealing to the Central Limit Theorem to arrive at a normal distribution of measurement errors is appropriate unless there is other evidence or knowledge to indicate otherwise.

5.1.2 Hyperparameters

The canonical distributions presented above are all parameterized by quantities θ. The degree of uncertainty about the parameters θ is expressed at one level in terms of the pdf $p(\theta|H)$. It is represented functionally as $p(\theta|H)=h(\theta;\alpha)$, where α are the parameters of the function h(.). For example, if θ are normally distributed with mean μ and covariance matrix Σ, then $p(\theta|H)=\phi(\theta;\mu,\Sigma)$ and $\alpha=\{\mu,\Sigma\}$. However, $p(\theta|H)$ itself may have an additional level of uncertainty attached to it which we express in terms of pdfs on the parameters α appearing in $h(\theta;\alpha)$. These α are called hyperparameters, and their uncertainty is explicitly accounted for by introducing pdf $p(\alpha|H)$. Therefore, $p(\theta|H)=\int p(\theta|\alpha,H)p(\alpha|H)d\alpha$, where we have applied marginalization to integrate out the "nuisance" parameters α.

Alternatively, often in seeking point estimates for θ, for pragmatic reasons we wish to avoid performing the integration needed to account for the hyperparameters α and rather seek to fix α or obtain point estimates for them as well. To do this, we set $p(\alpha|H)=\delta(\alpha,\alpha^*)$. Each value α^* we hypothesize for α represents a new modeling

hypothesis H. Thus, selecting α becomes a hypothesis selection decision problem as described in Section 4.3.4.

Though we could inject another layer of uncertainty by introducing pdfs for the hyper-hyperparameters that parameterize $p(\alpha|H)$ (and continue *ad infinitum*), empirical knowledge indicates that inferences about θ and subsequently the state estimate S^* are generally insensitive to values of hyper-hyperparameters, etc. (Berger, 1985). So, for simplicity's sake, we will at most consider only the single additional level of hyperparameters described above.

5.2 Finite Mixture Distributions

When a quantity u follows a pdf that cannot be readily determined to be a single canonical distribution, it is useful to characterize the pdf as a combination of canonical distributions. This characterization can be done nonparameterically by allowing the data to determine the number of members to be combined. Alternatively, it may be done by recognizing the multiple modes of operation and explicitly specifying the set of distinct pdfs. In either case, once the component or kernel distributions have been specified, the resulting pdf is representable as a finite mixture distribution.

A finite mixture distribution $p(u|H)$ combines a set of K mutually exclusive and exhaustive models $\mathfrak{M}=\{M_1, M_2,...,M_K\}$, where each M_j defines a pdf $p(u|M_j,H)$. The resulting pdf is as follows:

$$p(u|H) = \sum_{j=1}^{K} p(u|M_j, H)P(M_j|H) = \sum_{j=1}^{K} q_j \, h_j(u;a_j) \tag{5.1}$$

where each $p(u|M_j,H)$ is represented by a parametric function $h_j(u;a_j)$, and $P(M_j|H)$ is the probability of the discrete proposition "the applicable model is M_j." The $P(M_j|H)$ are real-valued $q_j \in [0,1]$, $\forall\, j=1,2,...,K$ and $\sum_{j=1}^{K} q_j = 1$ (Titterington, *et al.*, 1985). Note that not only have hyperparameters a_j been introduced to parameterize the kernel distributions

$h_j(.)$, but also K, the number of kernels, and the q_j are hyperparameters. All of these hyperparameters should be estimated and validated by solving the hypothesis selection decision problem.

The representation in Equation 5.1 is rich enough to describe any arbitrary pdf as $K \to N \to \infty$. So it has great value in applications such as nonparametric density estimation, which describes the interdependence of variables based on data. Also, combining models that apply in different process operating regimes can be represented as finite mixture models with the mixing probabilities expressed as functions of $\tilde{x}$.

5.2.1 Nonparametric Density Estimation

When a suitable parametric function for $p(u|H)$ cannot be specified *a priori*, one recourse is to use data to derive a nonparametric estimator. In the probabilistic paradigm, nonparametric models take the form of finite mixture models with the component pdfs drawn from the same family of distributions. In this research, we use normal distributions to represent $h_j(u;a_j)$; and, assuming that local model M_j dictates $u \sim N(\mu_j,\Sigma_j)$ – i.e., u is locally distributed normally with mean μ_j and covariance matrix Σ_j – we write $p(u|M_j,H)=h_j(u;a_j) = \phi(u;\mu_j,\Sigma_j)$, where $a_j=\{\mu_j,\Sigma_j\}$, and $\phi(u;\mu_j,\Sigma_j) = \left[(2\pi)^n|\Sigma_j|\right]^{-1/2} \exp\left[-\frac{1}{2}(u-\mu_j)^T\Sigma_j^{-1}(u-\mu_j)\right]$. Thus, the nonparametric estimator of u is as follows:

$$p(u|H) = \sum_{j=1}^{K} q_j\, h_j(u;a_j) = \sum_{j=1}^{K} q_j \phi(u;\mu_j,\Sigma_j). \tag{5.2}$$

Chapter 6 describes the procedure for estimating hyperparameters $\alpha=\{q_j,\mu_j,\Sigma_j,K\}$ from data.

5.2.2 Regression

Another application of finite mixture distributions is regression. When regressing dependent variables $\tilde{y}$ on independent variables $\tilde{x}$, we require $p(\tilde{y}\,|\,\tilde{x},\theta,H)$. This pdf may be represented as a finite mixture distribution for either nonparametric or parametric estimators as follows:

$$\begin{aligned} p(\tilde{y}|\tilde{x},\theta,H) &= p(\tilde{x},\tilde{y}|\theta,H)/p(\tilde{x}|H) \\ &= \sum_{j=1}^{K} p(\tilde{y}|\tilde{x},\theta,M_j,H)\frac{p(\tilde{x}|M_j,H)P(M_j|H)}{p(\tilde{x}|H)}. \\ &= \sum_{j=1}^{K} p(\tilde{y}|\tilde{x},\theta,M_j,H)P(M_j|\tilde{x},H) \end{aligned} \tag{5.3}$$

Each $p(\tilde{y}\,|\,\tilde{x},\theta,M_j,H)$ captures the local dependence of $\tilde{y}$ on $\tilde{x}$ as defined by model M_j. Also, the probability of applying model M_j at $\tilde{x}$ is

$$P(M_j|\tilde{x},H) = p(\tilde{x}|M_j,H)P(M_j|H)/p(\tilde{x}|H).$$

Importantly, $P(M_j|\tilde{x},H)$ is a function of $\tilde{x}$ and is ***not*** equivalent to $P(\tilde{x}\in D_j|H)=\int_{\tilde{x}\in D_j} p(\tilde{x}|H)d\tilde{x}$, where D_j is the domain of applicability of model M_j.

($P(\tilde{x}\in D_j|H)$ is the probability of a value $\tilde{x}$ being picked from D_j if any $\tilde{x}$ were randomly chosen from the set of real numbers: it is not a function of $\tilde{x}$; it is a function of the boundaries of D_j, i.e., $P(\tilde{x}\in D_j|H)=g(D_j)$.)

5.2.2.1 Nonparametric Regression

In nonparametric regression, $P(M_j|\tilde{x},H)$ is derived from the nonparametric density estimation of $p(\tilde{x}|H)$, which sets $P(M_j|H)=q_j$ and $p(\tilde{x}|M_j,H)=h_j(\tilde{x};a_j)$ from Eqn. 5.1. Therefore,

$$P(M_j|\tilde{x},H) = \frac{q_j\, h_j(\tilde{x};a_j)}{\sum_{k=1}^{K} q_k\, h_k(\tilde{x};a_k)} \equiv K(\tilde{x};\alpha). \tag{5.4}$$

Also, the local pdfs of a nonparametric model are usually assumed to be delta functions

applying local models $f_j(\tilde{x};\theta_j)$: $p(\tilde{y}\,|\,\tilde{x},\theta,M_j,H) = \delta(\tilde{y};f_j(\tilde{x};\theta_j))$. Therefore, the nonparametric form for $p(\tilde{y}\,|\,\tilde{x},\theta,H)$ becomes the following:

$$p(\tilde{y}|\tilde{x},\theta,H) = \sum_{j=1}^{K} \delta\left(\tilde{y}, f_j\left(\tilde{x};\theta_j\right)\right) K\left(\tilde{x};\alpha_j\right). \tag{5.5}$$

Equation (5.5) represents a universal approximator capable of modeling an arbitrary functional relation between $\tilde{x}$ and $\tilde{y}$, yet it is easily synthesized due to the simple form of the $f_j(\tilde{x};\theta_j)$, which are usually constants or linear functions of $\tilde{x}$. The α_j and θ_j can be efficiently estimated separately by first applying density estimation to the x-data to find the α_j and then regressing the θ_j. Sections 6.1 and 6.2 present the solution methods to achieve this.

Of note is the relation of the above finite mixture model representation with the usual form of a nonparametric model when the $f_j(\tilde{x};\theta_j)=\theta_j$, simple constants. The predictive model assuming certainty in parameters yields predictions y_p of the following form:

$$\begin{aligned} y_p &= \int \tilde{y}\, p\left(\tilde{y}|\tilde{x}=x_p,\theta,H\right)\delta\left(\theta,\theta^*\right)d\theta d\tilde{y} = \int \tilde{y} \sum_{j=1}^{K} \delta\left(\tilde{y},\theta_j\right) K\left(x_p;\alpha_j\right)\delta\left(\theta,\theta^*\right)d\theta d\tilde{y} \\ y_p &= \sum_{j=1}^{K} \theta_j^* K\left(x_p;\alpha_j\right) \end{aligned} \tag{5.6}$$

In the case that the $h_j(\tilde{x};a_j)$ are normal distributions $\phi(\tilde{x};\mu_j,\Sigma_j)$, the kernel functions $K(\tilde{x};\alpha_j)$ are the much studied Nadaraya-Watson kernels (Eubank, 1988). Traditionally, a kernel is placed at each data point (i.e., $K=N$ and $q_j=1/N$), and each kernel is spherical, thus characterized by the single "kernel bandwidth parameter" a_j. Also, the $f_j(\tilde{x};\theta_j)=\theta_j$ are simple constants, so only the N θ_j and the N a_j must be estimated. The key to the success of this formulation is in finding suitable values for a_j, which determines the smoothness of the estimator. The literature is extensive on such estimators (e.g., Fan and Marron, 1992; Hall, *et al.*, 1991; Van Es, 1992). Chapter 6 presents a robust and efficient solution to this regression problem when the $h_j(\tilde{x};a_j)$ are normal distributions.

5.2.2.2 Semiparametric Regression

If the data from which a nonparametric model is derived do not satisfactorily cover the domain of $\tilde{x}$, then the components of the nonparametric model do not represent an exhaustive set of models. In such a case, the nonparametric model M_{Np} must be augmented by a parametric model M_P to provide an exhaustive set. The result is a semiparametric finite mixture distribution of the following form:

$$\begin{aligned} p(\tilde{y}|\tilde{x},\theta,H) = p(\tilde{y}|\tilde{x},\theta,M_{Np},H)P(M_{Np}|\tilde{x},H) + \\ p(\tilde{y}|\tilde{x},\theta,M_P,H)\left[1-P(M_{Np}|\tilde{x},H)\right] \end{aligned} \tag{5.7}$$

$P(M_{Np}|\tilde{x},H)$ is derived from the components of the nonparametric density estimator, except now $p(\tilde{x}|H)$ has an additional term representing the regime where the nonparametric estimator is deemed inapplicable:

$$P(M_{Np}|\tilde{x},H) = \frac{p(\tilde{x}|M_{Np},H)P(M_{Np}|H)}{p(\tilde{x}|H)}, \tag{5.8}$$

where the nonparametric density estimator is $p(\tilde{x}|M_{Np},H) = \sum_{j=1}^{K} q_j\, h_j(\tilde{x};a_j)$; and $P(M_{Np}|H)$ is the prior probability that the nonparametric model is applicable.

The full $p(\tilde{x}|H)$ has the following form:

$$\begin{aligned} p(\tilde{x}|H) &= p(\tilde{x},M_{Np}|H) + p(\tilde{x},M_P|H) \\ &= p(\tilde{x}|M_{Np},H)P(M_{Np}|H) + p(\tilde{x}|M_P,H)\left[1-P(M_{Np}|H)\right] \end{aligned} \tag{5.9}$$

where $p(\tilde{x}|M_P,H)$, is the density under the assumption that the parametric model applies (i.e., that the nonparametric model is inapplicable).

Prior knowledge dictates $P(M_{Np}|H)$ and $p(\tilde{x}|M_P,H)$. These values are dependent on the specifics of a particular problem. In Section 5.3.2 we discuss this issue further in the context of eliciting the prior pdf $p(\tilde{x}|H)$.

5.3 Elicitation of Probability Density Functions

In Chapter 4, we showed how the predictive distribution $p(S|E,D,H)$ can be decomposed into more basic pdfs of which we generally have more information. This top-down decomposition was summarized in Figure 4.1. In this section, we show how a modeler would go about deriving these basic pdfs and building $p(S|E,D,H)$ from the bottom up.

The most basic pdfs must be formulated from the modeler's prior knowledge – they are the mechanism through which prior knowledge is injected into model synthesis. The most basic pdfs are (1) the evidential dependence model, (2) the prior pdf of the independent variables, (3) the error model, and (4) the prior pdf of the parameters. The knowledge base represented by hypothesis H dictates the forms of these pdfs and how they are combined to form the observation, measurement and state dependence models. From there, the axioms of probability theory dictate the posterior pdf of θ, essential for parameter estimation; the marginal distribution of the data, essential for hypothesis selection; and ultimately the predictive distribution, essential for state estimation.

5.3.1 Evidential Dependence Model

The evidential dependence model $p(S|E,\theta,H)$ captures knowledge about the relationship between the state and the evidence. In the regression case, this pdf is written $p(\tilde{y}\,|\,\tilde{x},\theta,H)$. This knowledge may include the parametric functional relation $f(\tilde{x};\theta)$ between $\tilde{y}$ and $\tilde{x}$. This relation is represented in accord with the degree of certainty ascribed to it:

(1) Certainty $\rightarrow f(\tilde{x};\theta)$ as an equality constraint: $p(\tilde{y}\,|\,\tilde{x},\theta,H)=\delta(\tilde{y},f(\tilde{x};\theta))$.

(2) Unbiased uncertainty $\rightarrow f(\tilde{x};\theta)$ as expected value of $\tilde{y}$ about which the distribution is symmetric: $p(\tilde{y}\,|\,\tilde{x},\theta,H)=\phi(\tilde{y};f(\tilde{x};\theta),\Omega)$, where the normal distribution is dictated by symmetry about the constraint on the expectation, and covariance matrix Ω is an hyperparameter.

(3) Uncertainty with nonparametric bias → "guide" function f($\tilde{x}$;θ) and bias b:

$$p(\tilde{y}|\tilde{x},\theta,H)=\int p(\tilde{y}|\tilde{x},b,\theta,H)p(b|\tilde{x},\theta,H)db;$$

$$p(\tilde{y}|\tilde{x},b,\theta,H)=\delta(\tilde{y},f(\tilde{x};\theta)+b);\text{ and }p(b|\tilde{x},\theta,H)=\sum_{j=1}^{K}\delta(b,\theta_j)K(\tilde{x};\alpha_j).$$

Therefore,

$$p(\tilde{y}|\tilde{x},\theta,H)=\int\delta(\tilde{y},f(\tilde{x};\theta)+b)\sum_{j=1}^{K}\delta(b,\theta_j)K(\tilde{x};\alpha_j)db$$

$$=\sum_{j=1}^{K}\delta(\tilde{y},f(\tilde{x};\theta)+\theta_j)K(\tilde{x};\alpha_j),$$

where $K(\tilde{x};\alpha_j)\equiv q_j\,h_j(\tilde{x};\alpha_j)\Big/\sum_{j=1}^{K}q_j\,h_j(\tilde{x};\alpha_j)$, as in Eqn. (5.4).

The use of a prior model serving as a guide function is helpful if the bulk of the behavior inherent in the dependence of $\tilde{y}$ on $\tilde{x}$ can be specified *a priori* as f(.), thus, easing the task of capturing the remaining bias *b* (i.e., model mismatch). The nonparameteric bias model in this case is usually considerably smaller than that necessary for the following case.

(4) General uncertainty → f($\tilde{x}$;θ) represented by nonparametric pdf:

$$p(\tilde{y}|\tilde{x},\theta,H)=\sum_{j=1}^{K}\phi(\tilde{y};f_j(\tilde{x};\theta_j),\Omega)K(\tilde{x};\alpha_j).$$

(5) Multiple operating regimes → f($\tilde{x}$;θ) represented by a finite mixture distribution: $p(\tilde{y}|\tilde{x},\theta,H)=\sum_{j=1}^{K}p(\tilde{y}|\tilde{x},\theta,M_j,H)P(M_j|\tilde{x},H)$ where each local model p($\tilde{y}$ | $\tilde{x}$,θ,M_j,*H*) can assume any one of the forms listed above or even represent a hierarchy of models by being yet another mixture. The next section explains how to derive P(M_j| $\tilde{x}$,*H*) for semiparametric models. For parametric models,

$P(M_j|\tilde{x},H)$ must be specified *a priori*. Also, this model form can include knowledge of asymptotic behavior: $P(M_j|\tilde{x},H)$ characterizes the bounds of the asymptotic regime and $p(\tilde{y}|\tilde{x},\theta,M_j,H)$ characterizes the model behavior at the asymptote. Finally, discontinuous functions can be represented as a finite mixture distribution with discontinuous $P(M_j|\tilde{x},H)$.

(6) Unmeasured intermediate $\rightarrow$ marginalization is used to integrate out the intermediate v: $p(\tilde{y}|\tilde{x},\theta,H)=\int p(\tilde{y}|\tilde{x},v,\theta,H)p(v|\tilde{x},\theta,H)dv$, where $p(v|\tilde{x},\theta,H)$ assumes one of the forms described above. This is another way in which any number of models – parameteric or nonparametric – may be nested or connected in series.

(7) Bounds $\rightarrow$ bounding constraints can be represented as canonical distributions on $\tilde{y}=f(\tilde{x};\theta)$ if they correspond to the bounds specified and include the moment constraints required by a canonical pdf. Simple one-sided bounds such as $\tilde{y}>a$ (without a constraint on the expected value) are represented using the "indicator function" $I(\tilde{y}>a)$, which equals one when the relation $\tilde{y}>a$ is satisfied and zero otherwise. The constraint pdf $I(\tilde{y}>a)$ serves as a multiplicative factor in the evidential dependence model, and a may be a function or a hyperparameter. With respect to the parameter estimation problem, constraints bounding $\tilde{y}$ are infinite inequality constraints.

5.3.2 Prior pdf of the Independent Variables

As shown in Chapter 4, under the assumption of independent identically distributed (iid) observations, the state dependence model $p(\tilde{X},\tilde{Y}|\theta,H)$ factors into a series product of the

basic pdfs evaluated at each observation: $p(\tilde{X}, \tilde{Y} | \theta, H) = \prod_{i=1}^{N} p(\tilde{y}_i | \tilde{x}_i, \theta, H) p(\tilde{x}_i | H)$. The first pdf is the evidential dependence model $p(\tilde{y} | \tilde{x}, \theta, H)$ discussed above. The other pdf is the prior distribution of the independent variables $p(\tilde{x} | H)$.

This factorization is motivated by the common case that an evidential dependence model can be directly derived and that the prior on $\tilde{x}$ is considered uniform over some domain D_x, i.e. $p(\tilde{x} | H) = U(\tilde{x}; D_x)$. However, if the assumption of a uniform prior is not valid, either a suitable parametric form must be specified or the prior must be estimated nonparametrically. In the former case, such things as equality constraints among the $\tilde{x}$, $f(\tilde{x}) = 0$, and bounding inequalities, $g(\tilde{x}) > 0$, must be incorporated into $p(\tilde{x} | H)$. Note that equality constraints, if linear are readily applied to reduce the dimension of the $\tilde{x}$-space; while if nonlinear, they are applied as delta pdfs. As mentioned in Chapter 3, infinite inequality constraints on $\tilde{x}$ serve as filters or aids to data rectification of the independent variable, as demonstrated in Section 6.3.3.

In the nonparametric case, the density estimation discussed in Section 5.2.2 is applied to approximate $p(\tilde{x} | H)$ from the data. In the case of the semiparametric estimator of Eqn. 5.9, we must also specify $P(M_{Np} | H)$ (the prior probability that the nonparametric model is applicable) and $p(\tilde{x} | M_P, H)$ (the probability density of $\tilde{x}$ when the nonparametric model is inapplicable), which are related by Eqn. 5.8 to $P(M_{Np} | \tilde{x}, H)$ (the probability that the nonparametric model is applicable given $\tilde{x}$). Important considerations in assigning these values follow.

$P(M_{Np} | H)$ can be set by assessing one's confidence in the nonparametric model based on the adequacy of the data sampling. For example, an estimate of how frequently the nonparametric model must extrapolate beyond the data could be used to determine $1 - P(M_{Np} | H)$ and thus set $P(M_{Np} | H)$. Also, the Principle of Maximum Entropy could be applied by assuming a constraint on the expectation of a suitable function. For example, maximizing the entropy of the assignment of $P(M_{Np} | H)$ subject to a constraint on the mean

of the distribution of $\tilde{x}$ could be performed: Constrain E[$\tilde{x}$ |H] to equal a specific μ; where E[$\tilde{x}$ |H]=P(M_{Np}|H)p($\tilde{x}$ |M_{Np},H) + [1-P(M_{Np}|H)]p($\tilde{x}$ |M_P,H); (however, this particular constraint requires that p($\tilde{x}$ |M_P,H) be specified first).

The pdf p($\tilde{x}$ |M_P,H) is reasonably determined by invariance arguments or the Principle of Maximum Entropy. Both approaches yield a uniform distribution p($\tilde{x}$ |M_P,H)=c under the assumption that only bounds on $\tilde{x}$ are known. Here we illustrate the form of p($\tilde{x}$ |M_P,H) and P(M_{Np}|$\tilde{x}$,H) in three reasonable situations: (1) all $\tilde{x}$ are equally probable when M_{Np} is deemed inapplicable; (2) p($\tilde{x}$ |M_P,H) is negatively correlated with p($\tilde{x}$ |M_{Np},H); and (3) all $\tilde{x}$ are equally probable over the full $\tilde{x}$-domain. In each case we assume that P(M_{Np}|H) has been specified.

In the first case, p($\tilde{x}$ |M_P,H)=c, a constant. This is usually an appropriate assumption to impose. It means that when the nonparametric model is deemed inapplicable, all inputs are equally probable. (Note that if, for example, the inputs are related due to constraints, this uniform pdf would be specified in the space of the independent $\tilde{x}$ variables and the other inputs become dependent variables, i.e., additional $\tilde{y}$ variables.) In general, we would expect c to be much smaller than the maximum of p($\tilde{x}$ |M_{Np},H) because if the data are at all representative of common process operation, the probability mass outside of the data should be small. By manipulation of Eqns. 5.8 and 5.9, the probability that the nonparametric model is applicable at a given $\tilde{x}$ is expressed by the following equation:

$$P\left(M_{Np}|\tilde{x},H\right)=\frac{p\left(\tilde{x}|M_{Np},H\right)}{\beta+p\left(\tilde{x}|M_{Np},H\right)}=\frac{\sum_{j=1}^{K}q_j\,h_j\left(\tilde{x};\alpha_j\right)}{\beta+\sum_{j=1}^{K}q_j\,h_j\left(\tilde{x};\alpha_j\right)}; \qquad (5.10)$$

$$\text{where } \beta\equiv c\frac{\left[1-P\left(M_{Np}|H\right)\right]}{P\left(M_{Np}|H\right)}.$$

A value for c can be explicitly specified or implicitly derived by specifying β, which regulates the sensitivity of P(M_{Np}|$\tilde{x}$,H) to the nonparametric density p($\tilde{x}$ |M_{Np},H). At high

density relative to β, $P(M_{Np}|\tilde{x},H)$ is nearly constant and close to one. Conversely, at low density relative to β, $P(M_{Np}|\tilde{x},H)$ is approximately proportional to the nonparametric density and close to zero. Thus, β serves as a threshold value which, if exceeded by the nonparametric density, forces $P(M_{Np}|\tilde{x},H)$ towards a constant value near one.

In the second case, $p(\tilde{x}|M_P,H)=c[1-p(\tilde{x}|M_{Np},H)/p_{max}]$, where p_{max} is the maximum value of $p(\tilde{x}|M_{Np},H)$, and c is the same as in the above case. Combining Eqns. 5.8 and 5.9 yields the following equation for $P(M_{Np}|\tilde{x},H)$, where β is the same as above:

$$P(M_{Np}|\tilde{x},H)=\frac{p(\tilde{x}|M_{Np},H)}{\beta(1-\rho_{Np}(\tilde{x}))+p(\tilde{x}|M_{Np},H)}; \qquad (5.11)$$

$$\text{where } \rho_{Np}(\tilde{x})\equiv p(\tilde{x}|M_{Np},H)/p_{max}.$$

For the same $P(M_{Np}|H)$ and c, Eqn. 5.11 gives a larger $P(M_{Np}|\tilde{x},H)$ than Eqn. 5.10 because the scaled nonparametric density $\rho_{Np}(\tilde{x})$ is less than or equal to one for all $\tilde{x}$. Also, unlike Eqn. 5.10, Eqn. 5.11 forces $P(M_{Np}|\tilde{x},H)=1$ at $p(\tilde{x}|M_{Np},H)=p_{max}$. However, it is readily seen that for the common case with $P(M_{Np}|H)$ near one, β is approximately zero, and Eqns. 5.10 and 5.11 yield essentially the same value for $P(M_{Np}|\tilde{x},H)$.

In the third case, $p(\tilde{x}|H)=c\geq P(M_{Np}|H)p_{max}$. Eqn. 5.8 gives the following expression for $P(M_{Np}|\tilde{x},H)$:

$$P(M_{Np}|\tilde{x},H)=\frac{P(M_{Np}|H)}{c}p(\tilde{x}|M_{Np},H). \qquad (5.12)$$

This function for $P(M_{Np}|\tilde{x},H)$ is simply proportional to the nonparametric density. In general, for multi-modal nonparametric density, Eqn. 5.12 yields a much more variable $P(M_{Np}|\tilde{x},H)$ than the thresholded functions of Eqns. 5.10 and 5.11. Thus, the other two formulas for $P(M_{Np}|\tilde{x},H)$ are usually preferred.

Figure 5.1 illustrates these observations for a 2-component mixture of normal distributions. $P(M_{Np}|H)$ was set at 0.8 and c equals $0.1p_{max}$, which are conservative values corresponding to a modeling scenario with adequate data. The probability that the

nonparametric model is applicable $P(M_{Np}|\tilde{x},H)$ is denoted "P1" for the first case (Eqn. 5.10) and "P2" for the second (Eqn. 5.11). The nonparametric density $p(\tilde{x}|M_{Np},H)$ is also shown and is labeled "px." Note that for the third case $P(M_{Np}|\tilde{x},H)$ would overlay $p(\tilde{x}|M_{Np},H)$ in this plot for $c=p_{max}$ or would be proportionately smaller for c less than p_{max}. The values chosen for $P(M_{Np}|H)$ and c correspond to having a reasonably high confidence in the coverage of the data and adequacy of the nonparametric model. The $P(M_{Np}|\tilde{x},H)$ from Eqns. 5.10 and 5.11 are essentially the same.

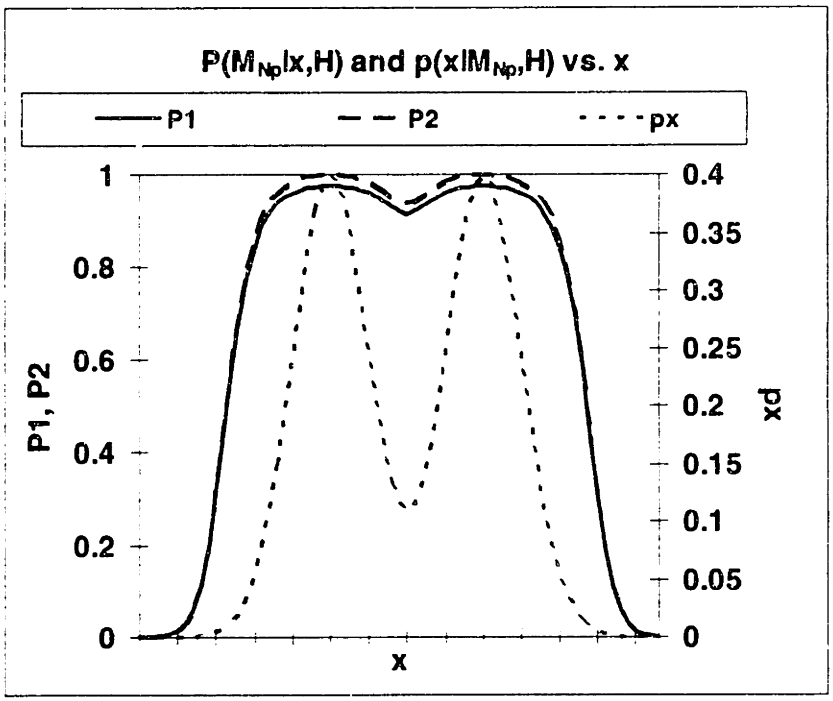


Figure 5.1: Probability of applying a data-rich nonparametric model

In Figure 5.2, the probabilities are depicted for $P(M_{Np}|H)$ equals 0.6 and c equals $0.5p_{max}$, which correspond to a modeling scenario with poorer data coverage yielding a relatively inadequate nonparametric model. The $P(M_{Np}|\tilde{x},H)$ of the first case, which assumes a uniform $p(\tilde{x}|M_P,H)$, is now much lower at all values of $\tilde{x}$. $P(M_{Np}|\tilde{x},H)$ of the second case, which assumes low $p(\tilde{x}|M_{Np},H)$ in regions of high nonparametric density, reaches one at p_{max} and has nearly the same variability as $p(\tilde{x}|M_{Np},H)$. This variability is usually not desirable because it causes high variance in the semiparametric model predictions.

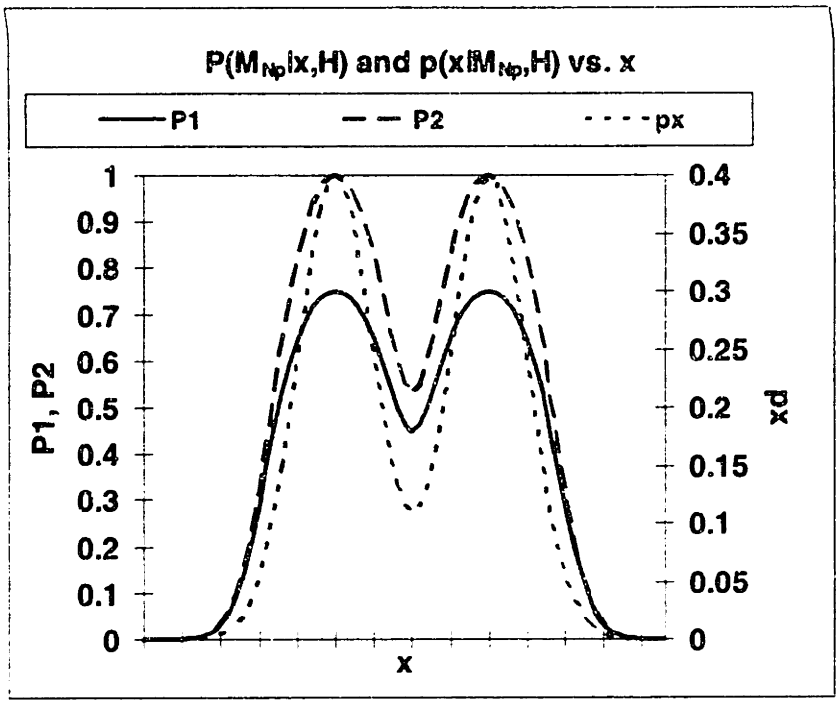


Figure 5.2 Probability of applying a data-poor nonparametric model

Regardless of the value chosen for $P(M_{Np}|H)$ and the function chosen for $p(\tilde{x}|M_P,H)$, the final arbiter on determining their validity is $p(D|H)$ as explained in the discussion of hypothesis selection in Section 4.3. This reiterates the theme that in the probabilistic paradigm all assumptions that are introduced must be explicitly represented and quantifiably accounted for in our confidence in the resultant model.

5.3.3 Error Model

The error model $p(\mathcal{E}|H)$ is another basic pdf. Traditionally it is used to capture the total uncertainty inherent in the model rather than merely the true measurement error. However, as seen above, the uncertainty in the model structure or doubt about its capabilities can be explicitedly specified in various ways. This is a direct result of using subjective probabilities rather than requiring the process to be stochastic or combined with stochastic measuring processes to justify injecting pdfs. Using pdfs to capture uncertainty

specifically associated with the dependence relationship between the state and evidence, separately from $p(\mathcal{E}|H)$, frees up the error model to better capture measurement error.

Usually, the errors are either uncorrelated or have short-term dependence determinable by time series analysis. Therefore $p(\mathcal{E}|H)$ can be written as a product of the $p(e_i|H)$ for each observation. The individual error models then assume the forms of the uncertainty representations shown in the previous section and presented here in terms of e.

(1) Certainty $\rightarrow e=g(.)$: $p(e|,H)=\delta(e,g(.))$, where $g(.)$ is known and may be a function of $\tilde{x}$, $\tilde{y}$, both or an arbitrary measurement bias. However, exact measurement, i.e. $g(.)=0$, is the most common type of certainty and usually is assumed of the independent variables $\tilde{x}$.

(2) Biased with symmetric uncertainty $\rightarrow$ errors distributed symmetrically about a known expected value: $p(e|H)=\phi(e;b,\sigma^2)$, where again the normal distribution is dictated by symmetry about the constraint on the expectation $E[e|H]=b$, and σ^2 is the noise variance, which is an hyperparameter.

(3) Biased with asymmetric uncertainty $\rightarrow$ errors distributed asymmetrically about mean b and having geometric mean γ, which induces skewness: $p(e|H)=Ga(e;b,\gamma)$. The hyperparameter γ can be estimated by hypothesis selection or chosen to give the desired amount of skewness then validated by hypothesis selection methods.

(4) Multiple error modes $\rightarrow p(e|H)$ represented by a finite mixture distribution: $p(e|H)=\sum_{j=1}^{K} p(e|F_j,H)P(F_j|H)$ where each local model $p(e|F_j,H)$ is one of the above parametric pdfs representing an

error or fault mode F_j, including normal sensor noise. Other modes include such things as stochastic outliers and deterministic biases.

Note, a generic nonparametric form of the error model is rarely applicable because its derivation implies the existence of known errors from which the nonparametric density function could be estimated.

5.3.4 Prior pdf of the Parameters

The prior pdf $p(\theta|H)$ for the parameters θ appears in the numerator of the posterior pdf of θ. The modeler's knowledge, preferences and assumptions about the process dictate this pdf. This may be done objectively by applying principles such as invariance of the probabilities under transformation of the variables. Recognizing the type of parameter, i.e. whether it is a location or a scale parameter, helps one to decide what type of invariance is the most appropriate. Location parameters such as the expected value of distributions are usually assumed to have a mean and/or to be bounded in some range. Depending on the type of bounds, a suitable canonical pdf can be used. Scale parameters such as the covariance matrix in the normal distributions can be assumed to yield invariant probability assignments under different scalings of the variables. Therefore, special priors such as Jeffreys' prior, below, are derived (Mardia, *et al.*, 1979; Berger, 1995).

Here we present a common prior distribution for the parameters of a multivariate normal distribution $\phi(x;\mu,\Sigma)$ (see Appendix B). The parameter set is $\theta=\{\mu,\Sigma\}$. We examine this cases because of the importance of normal distributions in deriving tractable solutions for probabilistic state estimation. This brief discussion of $p(\theta|H)$ is based on the derivation in Mardia, *et al.*, (1979) and Bernardo and Smith (1994), which also present further references.

The example prior we consider is a "non-informative" prior and is derived from invariance or Maximum Entropy considerations. By assuming that (1) μ and Σ are independent –

$p(\mu,\Sigma|H)=p(\mu|H)p(\Sigma|H)$ -- and (2) that we wish the probability assignments to be invariant upon (a) the location of μ and (b) the scaling of x, the following prior results:

$$p(\mu,\Sigma|H)=|\Sigma|^{-(n+1)/2}.$$

The resulting posterior $p(\mu|D,H)$ is a multivariate-t distribution and the optimal parameters θ are the same as the maximum likelihood estimates.

5.4 Representing Knowledge in the Motivating Examples

To demonstrate the representation of prior knowledge in process modeling as pdfs, examples of several problems faced by process modelers are listed below. These example problems are the same as the important motivating examples presented in Section 1.3. The representation of knowledge and solution of the state estimation problem in the context of most of these issues are presented in the case studies of Chapter 7.

5.4.1 Unmeasured Phenomena

Unmeasured variables are incorporated into this framework quite naturally. As shown in case (6) of Section 5.3.1, the evidential dependence model may involve unknown variables v. These intermediate variables are usually represented by a nonparametric pdf relating them to the inputs $\tilde{x}$, per Equation 5.3 with "v" replacing "$\tilde{y}$". This is done because the functional form of the relationship between v and $\tilde{x}$ cannot be specified *a priori*. The intermediates are then integrated out of the model, as shown in Section 4.4.2. This in essence averages the probability density of $\tilde{y}$ over the possible values of v and accounts for the uncertainty in v. Alternatively, the state vector S may be augmented with v, and the unmeasured intermediates can be estimated along with the outputs $\tilde{y}$.

5.4.2 Multiple Models: Parametric and Nonparametric

Section 5.2.2 explained the important role that finite mixture distributions play in combining multiple models, and Section 5.3.1 demonstrated how this information shapes

the evidential dependence model $p(\tilde{y} \mid \tilde{x}, \theta, H)$. Each local model $p(\tilde{y} \mid \tilde{x}, M_j, H)$ in case (5) of Section 5.3.1 can be either parametric – if its form is specified by prior knowledge – or nonparametric – if its form is not known *a priori*, and we wish to induce it from data. Any arbitrary semiparametric model may be formed by the nesting of any combination of parametric and nonparametric models as subcomponents.

5.4.3 Extrapolation: Role of a Default Model

In order to hedge against extrapolation, default knowledge is included in the evidential dependence model as demonstrated by the semiparametric regression of Section 5.2.2.2. The finite mixture combines a default model with any number of other models whose combined applicability does not fully cover process operating conditions. The probability of applying the default model is then given by the following equation:

$$P(M_{\text{Def}} \mid \tilde{x}, H) = 1 - \sum_j P(M_j \mid \tilde{x}, H),$$

where the summation is over all other models. The result is that the default prevents an inappropriate model from being used for estimation even though, otherwise that model would be highly improbable yet – without the default – the most probable of the specified models.

5.4.4 Corrupted Data: Finite Mixtures for Sensor Failure Modes

Discussed in Section 5.3.3, the error model pdf contains information about the measurement sensors. There it was shown that a finite mixture model is capable of representing knowledge about multiple failure modes for sensors. Each mode is modeled as a specified pdf, usually one of the canonical pdfs. Johnston and Kramer (1995) demonstrate this idea in the context of data rectification. The method they present is also applicable to error-in-variables regression, as was discussed earlier. Moreover, representing the error modes as normal distributions with different variances and means allows efficient estimation of the state using methods described in Chapter 6.

5.4.5 Process Faults: Finite Mixtures for Process Faults

Finite mixture distributions can be used to represent not only sensor failure modes but also arbitrary process faults. In such a case, the evidential dependence model becomes a finite mixture where each component represents the process behavior under a specific fault scenario and each is weighted by the probability that the fault will occur. Rojas-Guzmán (1995) investigated the use of probability theory using discrete-variate probability distributions in the form of Bayesian belief networks. Incorporating continuous-variate pdfs is straight-forward in principle, however as we see in Chapter 6, the solution methods tend to be computationally intensive and thus warrant approximative approaches.

5.4.6 Partitioned Data: Dynamic and Steady-State Data

It is clear from the state estimation problem decomposition described in Section 4.4 and summarized in Figure 4.1 that the process data enter into the problem in only two places: the likelihood of the parameters and the marginal pdf of the data (via the likelihood). In the case in which the data can be partitioned into independent sets, such as when some of the data have been collected during dynamic process operation and some at steady-state operation, the likelihood factors into two sub-likelihoods, one for each set of data. In this way, additional information gathered from steady-state operation can assist in parameter and state estimation. Another example is dynamic data series gathered from several independent runs: the full data likelihood is the product of the likelihoods of each run. The partitioned data and the factored likelihood are represented as follows:

$p(X, Y|\theta, H) = \prod_{p=1}^{P} p(X_p, Y_p|\theta, H)$, where P is the number of partitions, and X_p, Y_p are the data in partition $p=1,2,...,P$.

The probabilistic paradigm not only places the additional data in its proper place in the estimation problem, it also weights it appropriately according to the degree of certainty we wish to ascribe to it, which is represented by the dispersion of the error model for that data partition. In the case that this model is a normal distribution, it is the variance of the

steady-state errors that dictates the appropriate weight of each data set in the likelihood of the parameters and subsequently, the objective function of the parameter estimation problem (see Chapter 6).

5.5 Specialization to the Function-Oriented Paradigm

This chapter concludes by presenting the theoretical basis for the empirically derived semiparametric modeling approach of Chapter 3. Though that approach was derived from a function-oriented standpoint, we show that the probablistic paradigm makes explicit the assumptions and approximations that were made in Chapter 3. This is done by examining the probabilistic analogue to the role assignments summarized in Table 3.1 and noting the assumptions and approximations implied in transforming the general probabilistic assignments to the specific function-oriented assignments. Thus, the probabilistic paradigm provides an explanation for the empirical success of the method, yet identifies room for improvement.

5.5.1 Role Assignment of Prior Knowledge

This section answers the question "Into which pdf does a modeler put a given specific type of prior knowledge?" This is in contrast to Section 5.3, which answered the question "How does a modeler derive the pdfs required for state estimation?" That question adopts a probabilistic view of the problem, looking at it from the standpoint of what one needs for state estimation. The question posed here adopts a knowledge-based view of the modeling problem, looking at it from the standpoint of what one knows about the process. For most chemical engineers, this view is more familiar: we know our process and wish to represent our knowledge mathematically.

Table 5.2 inverts the previous statements of "this pdf includes that knowledge" to a summary stating "this knowledge goes into that pdf." The table is the analogue of Table 3.1 for the function-oriented approach. Note that because Table 5.2 is expressed in terms of inputs and outputs only, there is no explicit entry for knowledge about the error model.

This is because the function-oriented approach in Chapter 3 assumed normal, iid errors, thus knowledge about the errors was implicit in the approach.

Table 5.2: Probabilistic assignment of prior knowledge.

1. Relation Type	2. Functional Form	3. Precedence w.r.t. Data	4. Variables Involved	Role
Equality	Known	Greater	Inputs only	1. $p(\tilde{x} \mid H)$, as $\delta(\tilde{x}, g(\tilde{x}))$
			any Outputs	2. $p(\tilde{y} \mid \tilde{x}, \theta, H)$, as $\delta(\tilde{y}, f(\tilde{x}; \theta))$
		Lower	Any	3. $p(\tilde{y} \mid \tilde{x}, \theta, H)$, as canonical pdf
	Unknown	Greater	Any	4. $p(\tilde{y} \mid \tilde{x}, \theta, H)$, as nonparametric mixture
		Lower	Any	*provides no information*
Inequality	Known	Greater	Inputs only	5. $p(\tilde{x} \mid H)$, as canonical pdf or $I(\tilde{x} > a)$
			any Outputs	6. $p(\tilde{y} \mid \tilde{x}, \theta, H)$, as canonical pdf or $I(\tilde{y} > a(\tilde{x}))$
		Lower	Any	7. $p(\tilde{y} \mid \tilde{x}, \theta, H)$, as canonical pdf or $I(\tilde{y} > a(\tilde{x}; \theta))$ (dependent on θ); or $p(\theta \mid H)$
	Unknown	Greater	Any	*provides no information*
		Lower	Any	*provides no information*

The probabilistic role assignment reflects the different sources of uncertainty inherent in the problem and the appropriate ways in which to represent them probabilistically. *Roles 1* and *2* arise in the limit of assumed certainty about the equality of variables, the functional form of the equality relation, and the precedence of the constraint over the data. These "hard" equality constraints are represented by the delta function. "Soft" equality constraints – relations with precedence lower than that of the data – acknowledge that our prior knowledge is information with uncertainty to be compensated for by the information

contained in the data even though we can presuppose a definite functional form for the relation. Therefore, we admit uncertainty in the associated pdf of *Role 3*, choosing the appropriate canonical distribution according to what we assume about the relation. When the functional form of the equality constraint is unknown (*Role 4*), a nonparametric pdf is most appropriate.

Inequality constraints with known functional form and precedence greater than the data have an assumed certainty greater than the information contained in the data. They are represented as canonical pdfs if information about their expected values are injected, otherwise they are represented as the (filtering) pdf $I(u>a)$, where u is $\tilde{x}$ for inputs and $\tilde{y}$ for outputs (*Roles 5* and *6*). We shall see that in the solution approaches that seek a point estimate for the parameters, these constraints become penalty functions on the parameter estimation objective function (i.e., the Lagrangian of the constrained objective). However, as discussed in Chapter 3, it is most preferable to impose constraints on the outputs $\tilde{y}$ as parametric bias (e.g., as output models). Finally, a "soft" inequality constraint has the semantics of a preference that influences the estimation but may be violated if the data indicate behavior that is inconsistent with the constraint. Therefore, the constraint $\tilde{y}>a$, may be relaxed to $E[\tilde{y}\,|H]>a$. Such a constraint on the expectation is consistent with the knowledge assumed in a truncated exponential pdf (Table 5.1) (*Role 7*).

5.5.2 Probabilistic Representation of Function-Oriented Hybrid

The preceding sections of this chapter have demonstrated how process modeling knowledge is represented probabilistically. In this section we demonstrate the probabilistic representation of the hybrid model structure proposed in Chapter 3. In that chapter, intuitive arguments and empirical success motivated the incorporation of prior knowledge into model structure. Here, we show how the same knowledge injected into the probabilistic framework yields a more general structure that under certain specializing assumptions, coincides with the hybrid presented in Chapter 3.

The predictive model of our function-oriented approach is built from three types of functions dictated by what is known about the variables they relate. These functions are (1) a radial basis function neural network (RBFN) $f_{net}(\tilde{x};w)$, with weights w, that nonparametrically models equality relations with uncertain form and compensates for mismatch between a prior parametric model and the process; (2) a prior parametric model $f_{def}(\tilde{x})$ that serves the dual purpose as (*i*) a guide function for the RBFN that captures the bulk of the dependence between variables and (*ii*) a default model that provides a reasonable basis for extrapolation beyond the domain of the data-derived RBFN; and (3) a parametric output model $f_{out}(\tilde{x},v)$ that enforces equality constraints relating inputs $\tilde{x}$ and intermediates v to the model outputs. These models are combined into a hybrid model $f_{hyb}(\tilde{x};w)$ that computes predictions y_p from the following equation:

$$y_p = f_{hyb}(\tilde{x};w) = f_{out}(\tilde{x},f_{def}(\tilde{x})+f_{net}(\tilde{x};w)). \qquad (5.13)$$

In the probabilistic paradigm, this same knowledge is combined in the following manner. Given a point estimate of the parameters, the state estimate for the prediction under quadratic loss is $y_p = \int \tilde{y}\, p(\tilde{y}|\tilde{x},\theta,H)d\tilde{y}$. Knowing that $\tilde{y}$ is related to $\tilde{x}$ and unmeasured intermediates $v(\tilde{x})$ by $f_{out}(\tilde{x},v)$, we decompose the evidential dependence model $p(\tilde{y}|\tilde{x},\theta,H)$ to account for v:

$$p(\tilde{y}|\tilde{x},\theta,H) = \int p(\tilde{y}|\tilde{x},v,\theta,H)p(v|\tilde{x},\theta,H)dv = \int \delta\left(\tilde{y},f_{out}(\tilde{x},v)\right)p(v|\tilde{x},\theta,H)dv.$$

Our knowledge states that v and $\tilde{x}$ are related by a prior function $f_{prior}(\tilde{x})$ plus an unknown bias $b(\tilde{x})$, which we choose to model nonparametrically. Moreover, we know that the nonparametric component will not exhaustively cover the input domain; therefore, we augment it by a parametric default model $f_{def}(\tilde{x})$. Consequently, the relationship between v and $\tilde{x}$ is semiparametric and the resulting pdf is a finite mixture distribution:

$$p(v|\tilde{x},\theta,H) = P\left(M_{Np}|\tilde{x},H\right)p\left(v|\tilde{x},M_{Np},\theta,H\right)+\left[1-P\left(M_{Np}|\tilde{x},H\right)\right]p\left(v|\tilde{x},M_{p},\theta,H\right).$$

The parametric default model is represented with certainty:

$$p(v|\tilde{x}, M_P, \theta, H) = \delta(v, f_{def}(\tilde{x})).$$

The nonparametric model M_{Np} captures the unknown bias b (as in Section 5.3.1):

$$\begin{aligned} p(v|\tilde{x}, M_{Np}, \theta, H) &= \int p(v|\tilde{x}, b, \theta, H) p(b|\tilde{x}, \theta, H) db \\ &= \int \delta(v, f_{prior}(\tilde{x}) + b) \sum_{j=1}^{K} \delta(b, \theta_j) K(\tilde{x}; \alpha_j) db \\ &= \sum_{j=1}^{K} \delta(v, f_{prior}(\tilde{x}) + \theta_j) K(\tilde{x}; \alpha_j). \end{aligned}$$

Combining the equations, we construct the evidential dependence model:

$$\begin{aligned} p(\tilde{y}|\tilde{x}, \theta, H) = P(M_{Np}|\tilde{x}, H) \sum_{j=1}^{K} \delta(\tilde{y}, f_{out}(\tilde{x}, f_{prior}(\tilde{x}) + \theta_j)) K(\tilde{x}; \alpha_j) \\ + [1 - P(M_{Np}|\tilde{x}, H)] \delta(\tilde{y}, f_{out}(\tilde{x}, f_{def}(\tilde{x}))). \end{aligned}$$

Finally, the predictive model based on the probabilistic representation is obtained:

$$y_p = P(M_{Np}|\tilde{x}, H) \sum_{j=1}^{K} f_{out}(\tilde{x}, f_{prior}(\tilde{x}) + \theta_j) K(\tilde{x}; \alpha_j) + [1 - P(M_{Np}|\tilde{x}, H)] f_{out}(\tilde{x}, f_{def}(\tilde{x})). \quad (5.14)$$

Up to this point, we have simply taken our prior knowledge and combined it in the manner dictated by probability theory. Obviously the result does not correspond exactly to the predictive model in Eqn. 5.13 from the function-oriented approach. To specialize the more general Eqn. 5.14 to coincide with Eqn. 5.13, we must explicitly impose the assumptions implicit in the derivation of the function-oriented hybrid.

5.5.3 Qualifying Assumptions

Probability theory validates the assignment of prior knowledge in our function-oriented approach. The validity of the assignments are subject to the validity of the qualifying assumptions. These assumptions specialize the probabilistic formulation to the function-oriented approach. Note, that, to its credit, the approach in Chapter 3 can be rationalized in this manner, but not all function-oriented approaches can.

The first assumption necessary to specialize the probabilistic approach to the function-oriented hybrid is one that is commonly made (implicitly) in regression applications. The assumption is that the inputs $\tilde{x}$ are uniformly distributed. As shown in Section 5.3.3, this condition renders $P(M_{Np}|\tilde{x},H)$ (the probability that the nonparametric model is applicable) to be proportional to the nonparametric density $p(\tilde{x}|M_{Np},H)$:

Assumption 1: $p(\tilde{x}|H) = U(\tilde{x};D_x) = c$; therefore, $P(M_{Np}|\tilde{x},H) = \gamma\, p(\tilde{x}|M_{Np},H)$; $\gamma \equiv P(M_{Np}|H)/c$:

$$y_p = \sum_{j=1}^{K} f_{out}(\tilde{x}, f_{prior}(\tilde{x}) + \theta_j) q_j\, h_j(\tilde{x};a_j) + \left[1 - \gamma\, p(\tilde{x}|M_{Np},H)\right] f_{out}(\tilde{x}, f_{def}(\tilde{x})).$$

The next assumption is necessary to combine the K+1 evaluations of the output function of Eqn. 5.14 into the single evaluation of Eqn. 5.13. This can be done by evaluating $f_{out}(\tilde{x},v)$ at the expected value of v given $\tilde{x}$, $E[v|\tilde{x},H]$, rather than taking the expectation of $f_{out}(\tilde{x},v)$ over v, which was done in deriving the exact solution Eqn. 5.14. For these two calculations to be equivalent, the output model must be linear in v and $\tilde{x}$:

Assumption 2: $f_{out}(\tilde{x},v)$ is linear in v and $\tilde{x}$:

$$y_p = f_{out}\left(\gamma\, p(\tilde{x}|M_{Np},H)\tilde{x}, \gamma\, p(\tilde{x}|M_{Np},H) f_{pnor}(\tilde{x}) + \sum_{j=1}^{K} \gamma\theta_j q_j\, h_j(\tilde{x};a_j)\right)$$

$$+ f_{out}\left(\left[1 - \gamma\, p(\tilde{x}|M_{Np},H)\right]\tilde{x}, \left[1 - \gamma\, p(\tilde{x}|M_{Np},H)\right] f_{def}(\tilde{x})\right);$$

$$y_p = f_{out}\left(\tilde{x}, \gamma\, p(\tilde{x}|M_{Np},H) f_{pnor}(\tilde{x}) + \left[1 - \gamma\, p(\tilde{x}|M_{Np},H)\right] f_{def}(\tilde{x}) + \sum_{j=1}^{K} \gamma\theta_j q_j\, h_j(\tilde{x};a_j)\right).$$

The third assumption is simply that the same parametric model serves the dual purpose of prior guide function and extrapolative default, which is true of our function-oriented approach:

Assumption 3: $f_{prior}(\tilde{x}) \equiv f_{def}(\tilde{x})$:

$$y_p = f_{out}\left(\tilde{x}, f_{def}(\tilde{x}) + \sum_{j=1}^{K} \gamma\theta_j q_j\, h_j(\tilde{x};a_j)\right).$$

Finally, it becomes obvious that if normal distributions are used as the component pdfs of the nonparametric model, then the summation term in the above equation corresponds to a

radial basis function network (without a constant bias term). Therefore, applying this fourth assumption completes the specialization of the theoretically sound probabilistic approach of Eqn. 5.14 to our empirically motivated function-oriented approach of Eqn. 5.13:

$$\textit{Assumption 4}: \; h_j(\tilde{x};a_j)=\phi(\tilde{x};\mu_j,\Sigma_j);\; w_j=\gamma\theta_j q_j;\; \text{and } f_{net}(\tilde{x};w)=\sum_{j=1}^{K} w_j\phi(\tilde{x};\mu_j,\Sigma_j)$$

$$y_p = f_{out}(\tilde{x}, f_{def}(\tilde{x}) + f_{net}(\tilde{x};w)).$$

Of the assumptions imposed, certainly the second, assuming an output model linear in $\tilde{x}$ and v, is the most restrictive. In cases in which the output model is nonlinear, Eqn. 5.14 would give the predictions that are optimal with respect to a quadratic loss function, while Eqn. 5.13 would serve as an approximation to the optimal solution.

Also, the first assumption poses some mild concern in that for $P(M_{Np}|\tilde{x},H)$ proportional to $p(\tilde{x}|M_{Np},H)$, the predictive model can be more variable than is usually desirable because of the typical variability in the mixture of normal distributions used for $p(\tilde{x}|M_{Np},H)$. Our function-oriented approach manages this issue by choosing a nearest-neighbor factor λ by cross validation (CV), as presented in Johnston and Kramer (1993), that tends to broaden $p(\tilde{x}|M_{Np},H)$. The result, as observed in simulations, is that the effective density function employing λ chosen by CV is smoother than the maximum likelihood density estimator, which we have calculated using Expectation-Maximization and a uniform prior. Thus, λ is akin to a hyperparameter that is imposed on a prior distribution of the parameters in the nonparametric density estimator. This broadens and smooths the distribution relative to the maximum likelihood estimator and yields a predictor with lower variance than if $P(M_{Np}|\tilde{x},H)$ were set proportional to the maximum likelihood estimate of $p(\tilde{x}|M_{Np},H)$.

Chapter 6

Solution Methods

This chapter presents solution methods for the state estimation problem. Methods for nonparametric density estimation, parameter estimation, state prediction and hypothesis selection are derived. Finite mixtures of normal distributions and the optimization technique known as Expectation-Maximization (EM) play pivotal roles in these solutions. Moreover, most of the equations presented here were derived using the formulas of Appendix B. The chapter concludes with a description of the computer software written to implement these methods.

There are two main approaches to solving the probabilistic estimation problems posed in this research:

Approach 1: Given hyperparameters α, estimate parameters θ and predict S assuming $S^*=f(E;\theta,\alpha)$.

Approach 2: Sample the posterior distribution $p(\theta|D,H)$ many times and predict S using the sample mean of $f(E;\theta,\alpha)$ evaluated at each of the sampled θ.

Approach 1 is the probabilistic analogue to non-probabilistic modeling. For any given set of hyperparameters α, the parameter estimation problem is solved to find the optimal parameters θ^*. State prediction is then performed by evaluating the evidential dependence model. This evaluation is usually simple and assumes that the model is certain or at least that the predictive model represents the expected value of a symmetric evidential dependence model. The use of a single point estimate θ^* yields only an approximate solution for S, which is most accurate when both the posterior pdf of θ and the evidential model are delta functions or highly peaked. However, the motivating factors for using this approach are (1) the convenience offered by summarizing the data in a single parameter

set, (2) the efficient estimation of the parameter set by conventional optimization methods, and (3) the straight-forward evaluation of the model.

The first approach involves determining suitable values for the hyperparameters α, which may be the parameters of the kernel distributions in a nonparametric model or the lag (window length) used in windowing data from a dynamic process. Whatever are the α of the specific problem, each instantiation of the α to values α_i^* represents a separate modeling hypothesis H_i, and thus the choice of α is dictated by solving the hypothesis selection problem given a set of candidate H_i.

Given α, conventional mathematical programming methods may be used to estimate θ^* and usually offer more efficient solution means than the second approach discussed below. The optimization solution methods are nonlinear programming (NLP) and semi-infinite programming (SIP). Additionally, when finite mixture distributions are involved, the statistically motivated expectation-maximization (EM) algorithm enhances the efficiency of the solution algorithm.

Approach 2 involves sampling $p(S|E,D,H)$, which, assuming either certainty or symmetric uncertainty in the evidential dependence model $p(S|E,\theta,H)$, amounts to evaluating $f(E;\theta)$ at each θ sampled from $p(\theta|D,H)$. This approach potentially offers a more accurate solution than the optimization methods of the first approach. It also comes complete with an estimate of the solution's certainty in the form of the sample variance. However, this approach is often more computationally intensive than the first due to the requirement of generating a suitable sample from $p(\theta|D,H)$. The state of the art in solving problems using this approach are Markov chain Monte Carlo (MCMC) methods such as Gibbs sampling (GS) (Neal, 1993). These methods repeatedly draw from $p(\theta|D,H)$ to generate a sample $\{\theta_{(k)}\}$, and calculate the sample mean of $f(E;\theta_{(k)})$ to approximate the expectation of $p(S|E,\theta,H)$:

$$S^* = \int S\, p(S|E,\theta,H)\, p(\theta|D,H)\, d\theta\, dS \approx \sum_k f\left(E;\theta_{(k)}\right).$$

Section 2.1.2 described MCMC methods. In Section 6.6 we discuss our brief experimentation with these methods.

In this research, the first approach was investigated most extensively because of the promise of computationally efficient algorithms to solve the problem and the other motivating factors listed above. The EM algorithm provides an exceptionally efficient means of solving the problem if finite mixtures of normal distributions are used as nonparametric models. Such a representation is useful in a broad class of modeling problems. Moreover, even if the kernel distributions of the finite mixtures are arbitrary pdfs, the EM algorithm is useful in speeding up the solution. In these general cases, the M-step of the algorithm is itself an iterative procedure, yet still solutions are often arrived at more quickly than by generic mathematical programming methods.

For hypothesis selection, it is also usually the case that the multi-dimensional integration necessary to compute $p(D|H)$ and hence the Bayes' factors must be done numerically. And, as with prediction, MCMC methods are prime candidates. However, taking advantage of the special form offered by a finite mixture of normal distributions allows an analytical solution in some cases. We investigate these cases in Section 6.5.

6.1 Nonparametric Density Estimation

When the evidential and state dependence pdfs involve nonparametric estimators, as described in Section 5.2, the problem can be solved more readily by exploiting the finite mixture distribution representation. Especially efficient methods for estimating the hyperparameters α and parameters θ exist if the kernel distributions of the mixtures are normal distributions. We advocate using finite mixtures of normal distributions because of this property – in addition, of course, to their universal approximation property.

This section presents the expectation-maximization (EM) method for determining the hyperparameters α of the nonparametric density estimators presented in Eqn. 5.1 and used

in nonparametric regression. Under the assumption that there are no errors in the independent variables $\tilde{x}$, we wish to find the parameters of $p(\tilde{x}|H)$, which describes the probability density of the independent (input) variables $\tilde{x}$:

$$p(\tilde{x}|H) = \sum_{j=1}^{K} p(\tilde{x}|M_j, H) P(M_j|H) = \sum_{j=1}^{K} q_j \, h_j(\tilde{x}; a_j). \tag{6.1}$$

In the case of normal distribution kernels, $h_j(\tilde{x}; a_j)$ equals $\phi(\mu_j, \Sigma_j)$, where a_j is $\{\mu_j, \Sigma_j\}$, which are the kernel means and covariance matrices, respectively. Furthermore, normal distributions are useful when the quantities xi are not independent of each other and the conditional distribution $p(\tilde{x} | \tilde{X}_{(<i)}, H)$ is required, which is often true of dynamic systems. In that case, the joint pdf $p(\tilde{x}|H)$ can be estimated as described below and the conditional pdf calculated by applying the formulas of Appendix B.5 to each component of the distribution. This is demonstrated in the case studies of Chapter 7.

The nonparametric density estimation problem using the finite mixture of normal distributions is stated as follows:

Nonparametric Density Estimation Problem:

Given data $D=X$ and hypothesis H, which includes the assumption that $\tilde{X}=X$, find the set of parameters $\alpha \equiv \{a_j, q_j\}$, $j=1,2,...,K$, that maximizes $p(\alpha | \tilde{X}, H) = p(\alpha | X, H)$.

Note that in Section 6.4.1 we handle the "error-in-variables" case where $\tilde{X} \neq X$.

Applying Bayes' Theorem, $p(\alpha|X,H) = p(X|\alpha,H)p(\alpha|H)/p(X|H)$. The denominator is a scaling factor that can be ignored during maximization. Assuming independent observations x_i, the likelihood $p(X|\alpha,H)$ becomes a series product of $p(x_i|\alpha,H)$. The objective function $F(\alpha)$ to be maximized in the nonparametric density estimation problem is the following:

$$F(\alpha) = p(\alpha|H) \prod_{i=1}^{N} p(x_i|\alpha, H) = \prod_{i=1}^{N} \sum_{j=1}^{K} q_j \, h_j(x_i; a_j) p(a_j, q_j | H).$$

The motivation for applying the EM algorithm comes from recognizing that if we knew from which of the K kernel distributions each x_i were drawn, then we could replace the product of summations with a product of single kernels $h_j(.)$. So, to include this information, we introduce the latent indicator variables $Z=\{z_{ij}\}$, where $i=1,2,...,N$ and $j=1,2,...,K$. Z is an $N\times K$ matrix of zeros and ones: $z_{ij}=1$ indicates that x_i was drawn from $h_j(.)$, and for each x_i, one and only one $z_{ij}=1$ (i.e., each row of Z has a single element equal to one and all other elements in the row equal zero). By definition, $\mathrm{P}(z_{ij}=1|x_i,\alpha,H)=\mathrm{P}(M_j|x_i,\alpha,H)$; and therefore, because z_{ij} is binary, $\mathrm{E}[z_{ij}|x_i,\alpha,H]=\mathrm{P}(M_j|x_i,\alpha,H)$.

As usual, we use marginalization to inject the new quantities, namely Z, into the pdf:

$$\mathrm{p}(x_i|\alpha,H)=\sum_{z_i}\mathrm{p}(x_i,z_i|\alpha,H),$$

where z_i is the i^{th} row of Z and $\sum_{z_i}$ denotes the summation over the K possible combinations of the z_{ij} (i.e., $\{z_{i1}=1, z_{i2}=0, ..., z_{iK}=0\}$, $\{z_{i1}=0, z_{i2}=1, ..., z_{iK}=0\}$, ..., $\{z_{i1}=0, z_{i2}=0, ..., z_{iK}=1\}$). And, again because z_{ij} are binary, the continuous-variate form of the joint pdf $\mathrm{p}(x_i,z_i|a,H)$ can be expressed as a product of the kernel distributions and their prior probabilities of being selected raised to the appropriate z_{ij}:

$$\mathrm{p}(x_i,z_i|\alpha,H)=\mathrm{p}(x_i|z_i,\alpha,H)\mathrm{P}(z_i|\alpha,H)=\prod_{j=1}^{K}\left[q_j h_j(x_i;a_j)\right]^{z_{ij}},\ z_{ij}\in\{0,1\},$$

where $\mathrm{p}(x_i|z_i,\alpha,H)=\prod_{j=1}^{K}h_j(x_i;a_j)^{z_{ij}}$ and $\mathrm{P}(z_i|\alpha,H)=\prod_{j=1}^{K}q_j^{z_{ij}}$. The new objective function arises as follows:

$$\mathrm{F}(\alpha,Z)=-\sum_{i=1}^{N}\sum_{j=1}^{K}z_{ij}\left\{\log[q_j]+\log[\phi_j(x_i;\mu_j,\Sigma_j)]-\log[\mathrm{p}(\alpha_j|H)]\right\}. \quad (6.2)$$

But, we have incomplete information: we know only X and not the associated indicator matrix Z. The EM algorithm solves this problem.

The EM algorithm is a technique for estimating quantities for "missing data" problems in statistics. EM is defined and applied to a variety of such problems by Little and Rubin

(1987) and Dempster, *et al.* (1977). The algorithm is iterative and alternately performs an expectation or "E" step and a maximization or "M" step on each iteration. The E-step computes a surrogate objective function to $p(\alpha|X,H)$: the expectation of $\log[p(\alpha|X,Z,H)]$, where the expectation is over the distribution $P(Z|X,\alpha^{(t-1)},H)$, and $\alpha^{(t-1)}$ is the estimate for the parameters at the previous iteration. The M-step re-estimates the parameters α by maximizing the expectation from the E-step. It has been proven that this procedure increases the posterior probability of the parameter estimates $p(\alpha|X,H)$ at each iteration (Little and Rubin, 1987). Also, if the expectation is linear in the missing data, then the algorithm effectively maximizes $\log[p(\alpha|X,Z,H)]$ evaluated at the expected value of Z, $E[Z|X,\alpha^{(t-1)},H]$. The E-step in this case computes the expectation of the missing variables given X and $\alpha^{(t-1)}$.

The EM formulas used to solve the density estimation problem when the mixture kernels are normal distributions and the prior $p(\alpha|H)$ is uniform are given below (Titterington, *et al.* 1985):

E-Step: $\forall i=1,2,\ldots,N;\ j=1,2,\ldots,K$

$$E\left[z_{ij}|x_i,\alpha,H\right]=P\left(M_j|x_i,\alpha,H\right)=\frac{p\left(x_i|M_j,\alpha,H\right)P\left(M_j|H\right)}{p\left(x_i|H\right)}=\frac{q_j\phi\left(x_i;\mu_j,\Sigma_j\right)}{\sum_{l=1}^{K}q_l\phi\left(x_i;\mu_l,\Sigma_l\right)};\quad (6.3)$$

M-Step: $\forall j=1,2,\ldots,K$

$$q_j=\frac{1}{N}\sum_{i=1}^{N}z_{ij}\ ;\quad \mu_j=\frac{1}{N}\sum_{i=1}^{N}z_{ij}x_i\ ;\quad \Sigma_j=\frac{1}{N}\sum_{i=1}^{N}z_{ij}\left(x_i-\mu_j\right)^2\ .\quad (6.4)$$

Specifying Z effectively partitions the data into K clusters by assigning each point to a kernel M_j. After partitioning of the data, the problem has been decoupled into K separate, relatively easy, parameter estimation problems that can be solved quickly. This is the advantage of the EM approach.

6.2 Parameter Estimation

As stated in Section 4.3.2, the parameter estimation problem is as follows:

Parameter Estimation Problem:

> Given data D and subject to the assumptions imposed by hypothesis H, select the optimal θ under 0-1 loss. Thus, find θ that maximizes the posterior probability density function $p(\theta|D,H)$.

The pdf $p(\theta|D,H)$ is proportional to the product of the likelihood and the prior. Thus, parameter estimation requires the maximization of that product. Noting that the canonical probability distributions (Chapter 5) belong to the exponential family, it is usually convenient to maximize the natural logarithm of that product. For regression with independent observations, the parameter estimation problem becomes the following:

Parameter Estimation Problem:

> Given data $D=\{X,Y\}$ and subject to the assumptions imposed by hypothesis H, select the optimal θ under 0-1 loss. Thus, find θ that maximizes

$$F(\theta) = \log[p(X,Y|\theta,H)] + \log[p(\theta|H)]. \tag{6.5}$$

The parameter estimation objective function $F(\theta)$ derives its form from that of the likelihood $p(X,Y|\theta,H)$ and the prior $p(\theta|H)$. We have already seen, in Chapter 5, the functional forms of $p(X,Y|\theta,H)$ and $p(\theta|H)$. In what follows we will examine several methods for maximizing $F(\theta)$ that take advantage of these common forms.

The functional form of the likelihood $p(X,Y|\theta,H)$ arises from that of the state dependence model. All state dependence models fall in one of the following categories: (*I*) a single parametric model, (*II*) a single nonparametric model, (*III*) a combination of multiple parametric models, (*IV*) a combination of multiple nonparametric models, or (*V*) a semiparametric model that combines multiple parametric and nonparametric models. Below we show how the solution methods for case (*I*) and Expectation-Maximization

serve as the building blocks to a general solution method capable of estimating the parameters for cases (*II*) through (*V*).

In each case, our standard hypothesis H will assume (1) independent observations, (2) independent, identically distributed (iid) normal errors in y with variance σ^2, (3) exact measurement (no errors) in x, and (4) the uncertainty associated with a parametric model is captured as a normal distribution, i.e., $p(\tilde{y} \mid \tilde{x}, \theta, H) = \phi(\tilde{y}; f(\tilde{x}; \theta), \Omega)$. (Note σ^2 and Ω are hyperparameters set by hypothesis selection methods.) After examining the objective functions and solution methods under these assumptions, we will discuss parameter estimation methods when these assumptions do not hold. Regardless, it should be understood that these assumptions are not very restrictive. Assumption (1) almost always holds for steady-state modeling or, as in the case of dynamic modeling, can be approximately enforced by determining an appropriate lag using time series analysis and windowing the data. Assumption (2) usually is true, can be treated by windowing data if not true, and rarely is important if the data are ample. Assumption (3) is almost always valid or can be treated by the rectification/error-in-variables regression methods discussed in Section 6.4. Finally, Assumption (4) is usually valid given our state of knowledge about the process, i.e., predictions are expected to follow the proposed model $f(\tilde{x}; \theta)$ with symmetric variation about that mean.

6.2.1 Single Parametric Model

We showed earlier (Sections 4.4.6 and 4.4.7) that the state dependence model under the above assumptions is a series product that leads to a likelihood of the following form:

$$p(X, Y \mid \theta, H) = \prod_{i=1}^{N} \int \phi(y_i; \tilde{y}_i, \sigma^2) \phi(\tilde{y}_i; f(x_i; \theta), \Omega) d\tilde{y}_i \, p(x_i \mid X_{\{<i\}}, H).$$

The above convolution of the normal distributions has an analytical solution (App. B):

$$p(X,Y|\theta,H)=\prod_{i=1}^{N}\phi\left(y_i;f(x_i;\theta),\sigma^2 I+\Omega\right)p\left(x_i|X_{\{<i\}},H\right). \tag{6.6}$$

where I is the identity matrix. Therefore, the objective function in Eqn. 6.5 becomes a sum of the natural log of the prior and a series summation of quadratics, similar to the example of ordinary regression (Section 4.4.7):

$$F(\theta)=-\sum_{i=1}^{N}\tfrac{1}{2}\left[y_i-f(x_i;\theta)\right]^{T}\left(\sigma^2 I+\Omega\right)^{-1}\left[y_i-f(x_i;\theta)\right]+\log[p(\theta|H)]. \tag{6.7}$$

In general, mathematical programming methods are used to maximize this objective function. As in the function-oriented paradigm (Section 3.4), the solution method applied in a given problem is dictated by the types of constraints present, which are usually represented in the prior $p(\theta|H)$ or other multiplicative terms discussed in Chapter 5:

(a) If finite inequality constraints exist, use constrained nonlinear programming (NLP) methods.

(b) If infinite inequality constraints exist, use semi-infinite programming (SIP) methods.

(c) If no inequality constraints exist, use unconstrained NLP methods.

6.2.2 Single Nonparametric Model

The nonparametric state dependence model arises from a nonparametric estimator having the form given in Eqn. 5.3, which we present again here:

$$p(\tilde{y}|\tilde{x},\theta,H)=\sum_{j=1}^{K}\delta\left(\tilde{y},f_j(\tilde{x};\theta_j)\right)K(\tilde{x};\alpha_j), \text{ where } K(\tilde{x};\alpha_j)\equiv\frac{q_j\,h_j(\tilde{x};a_j)}{\sum_{j=1}^{K}q_j\,h_j(\tilde{x};a_j)}.$$

Therefore the likelihood is the following:

$$p(X,Y|\theta,H)=\prod_{i=1}^{N}\sum_{j=1}^{K}\phi\left(y_i;f_j(x_i;\theta_j),\sigma^2\right)K(x_i;a_j)\,p\left(x_i|X_{\{<i\}},H\right).$$

As in the case of nonparametric density estimation of the hyperparameters q_j and α_j (Section 6.1), this likelihood involving summations can be transformed into a surrogate function involving only products by introducing latent indicator variables Z. However, in the parameter estimation case, the probabilities and expectation of z_{ij} are conditioned upon both the x and y data. The resulting objective function is solved using EM where the E-step and M-step are described below. Again, $\log[p(x_i,y_i|\theta,M_j,H)]$ is linear in the missing values Z. So, the objective function to be maximized in the M-step is $F(\theta;Z)$ a surrogate function of $F(\theta)$ in Eqn. 6.5, and is evaluated at the expected values of Z. $F(\theta;Z)$ has the following form, after dropping all terms that are independent of θ:

$$\begin{aligned} F(\theta;Z) &= \sum_{j=1}^{K} \log\left[p\left(y_i|x_i,\theta,M_j,H\right)\right] + \log\left[p(\theta|H)\right] - C(x_i) \\ &= -\sum_{j=1}^{K}\sum_{i=1}^{N} \tfrac{1}{2} z_{ij}\left(y_i - f_j\left(x_i;\theta_j\right)\right)^{T}\left(\sigma^{-2} I\right)\left(y_i - f_j\left(x_i;\theta_j\right)\right) + \log\left[p(\theta|H)\right], \end{aligned} \tag{6.8}$$

where each z_{ij} that equals one allocates the point $\{x_i,y_i\}$ to model $f_j(x;\theta_j)$; and $C(x_i)$ contains all terms from $\log[p(x_i,y_i|\theta,M_j,H)]$ that are independent of θ.

The E-step is similar to that for density estimation (Eqn. 6.3), but importantly in the regression case the allocation of a data point to a kernel is dependent on how accurately the kernel's model $f_j(x;\theta_j)$ predicts y_i. The expected value of z_{ij} computed in the E-step is given as follows:

E-Step: $\forall i=1,2,\ldots,N;\ j=1,2,\ldots,K$

$$\begin{aligned} E\left[z_{ij}|x_i,y_i,\theta,H\right] &= P\left(M_j|x_i,y_i,\theta,H\right) = \frac{p\left(y_i|x_i,\theta,M_j,H\right)P\left(M_j|x_i,H\right)}{p\left(y_i|x_i,\theta,H\right)} \\ &= \frac{p\left(y_i|x_i,\theta_j,M_j,H\right)p\left(x_i|M_j,H\right)P\left(M_j|H\right)}{p\left(x_i,y_i|\theta,H\right)} \\ &= \frac{\phi\left(y_i;f_j\left(x_i;\theta_j\right),\sigma^2\right)q_j\, h_j\left(x_i;a_j\right)}{\sum_{l=1}^{K}\phi\left(y_i;f_j\left(x_i;\theta_j\right),\sigma^2\right)q_l\, h_j\left(x_i;a_j\right)}. \end{aligned} \tag{6.9}$$

Also important to note, if the parameters θ_j of each model $f_j(.)$ are independent of those in the other models $f_k(.)$, $k \neq j$, then $p(\theta|H)=p(\theta_1|H)p(\theta_2|H)...p(\theta_K|H)$. Therefore, the K maximization subproblems can be solved in parallel. The K subobjectives $F_j(\theta_j;Z_j)$ are defined as follows, where Z_j is the j^{th} column of Z:

$$F_j(\theta_j;Z_j) \equiv \sum_{i=1}^{N} -\tfrac{1}{2} z_{ij}\left(y_i - f_j(x_i;\theta_j)\right)^T (\sigma^{-2} I)\left(y_i - f_j(x_i;\theta_j)\right) + \log\left[p(\theta_j|H)\right]. \quad (6.10)$$

In general, the M-step involves solving the single-parametric-model problem of the previous section at each EM iteration and would be performed by mathematical programming methods described in Chapter 3. However, because a nonparametric estimator using linear models $f_j(x;\theta_j)$ possesses the universal approximation property, we use linear functions $f_j(x;\theta_j) = u_i^T \theta_j$, where u_i is x_i augmented by the constant 1 (intercept). The M-step solution in this case follows:

M-Step: $\forall j=1,2,...,K$

$$\theta_j = \left(U_j^T U_j\right)^{-1} U_j^T Y, \text{ where the } i^{th} \text{ row of } U_j \text{ equals } z_{ij} u_i^T. \quad (6.11)$$

6.2.3 Multiple Parametric Models

As discussed in Chapters 2 and 3, models may be combined in a parallel or serial fashion. The bases for these representations can be justified in the probabilistic paradigm, and we showed this in Chapter 5. Parallel models arise from a finite mixture representation of the evidential dependence model and serial models arise from the representation of the uncertainty introduced by unmeasured quantities. In this section we present solution methods for both types of models when the components are parametric.

6.2.3.1 Finite Mixture Distributions (Parallel Models)

When the state dependence model involves the combination of multiple models in a finite mixture distribution, we can again take advantage of the special structure and solve the parameter estimation problem more efficiently using EM than using conventional

mathematical programming methods. As we shall see, by applying EM, the probabilistic paradigm efficiently allocates data sets to parameter sets.

Similar to the case of nonparametric regression, the likelihood of θ when the state dependence model is comprised of multiple models involves a summation that complicates the optimization procedure. In such a case, the likelihood is as follows:

$$\begin{aligned} p(X,Y|\theta,H) &= \prod_{i=1}^{N} p(y_i|x_i,\theta,H)p(x_i|H) \\ &= \prod_{i=1}^{N} p(x_i|H)\sum_{j=1}^{K} p(y_i|x_i,\theta,M_j,H)P(M_j|x_i,H) \\ &= \prod_{i=1}^{N} p(x_i|H)\sum_{j=1}^{K} \phi(y_i;f(x_i;\theta),\sigma^2 I+\Omega_j)\omega_j(x_i) \end{aligned} \tag{6.12}$$

where $\omega_j(x)$ is the function representing the probability distribution $P(M_j|x,H)$; and Eqn. 6.12 is based on the individual model likelihoods derived as Eqn. 6.6 for the single parametric model.

Applying this likelihood in the objective function $F(\theta)$ of Eqn. 6.5 and injecting indicator variables Z allows us to apply EM to readily estimate θ. Thus, as in the case of the single nonparametric model, the objective function can be expressed in terms of the K subproblem objectives $F_j(\theta_j;Z_j)$:

$$F(\theta;Z) = \sum_{j=1}^{K} F_j(\theta_j;Z_j). \tag{6.13}$$

However, the sum of squares in the $F_j(\theta_j;Z_j)$ of this case are weighted by the broader covariance matrix $\sigma^2 I+\Omega_j$ rather than by $\sigma^2 I$ as in Eqn. 6.10.

The E-step is as follows:

E-Step: $\forall i=1,2,...,N;\ j=1,2,...,K$

$$\mathrm{E}\left[z_{ij}\mid x_i,y_i,\theta,H\right]=\mathrm{P}\left(M_j\mid x_i,y_i,\theta,H\right)=\frac{\mathrm{p}\left(y_i\mid x_i,\theta,M_j,H\right)\mathrm{P}\left(M_j\mid x_i,H\right)}{\mathrm{p}\left(y_i\mid x_i,\theta,H\right)}$$
$$=\frac{\phi\left(y_i;\mathrm{f}_j\left(x_i;\theta_j\right),\sigma^2\mathrm{I}+\Omega_j\right)\omega_j\left(x_i\right)}{\sum_{l=1}^{K}\phi\left(y_i;\mathrm{f}_j\left(x_i;\theta_j\right),\sigma^2\mathrm{I}+\Omega_j\right)\omega_j\left(x_i\right)}. \tag{6.14}$$

Note that both the accuracy of the model in predicting y_i (via $\phi(.)$) and its applicability at the point x_i (via $\omega_j(.)$) determine the allocation of the i^{th} data point to each model.

We use a hybrid NLP and EM algorithm to maximize $F(\theta)$ of Eqn. 6.13. The E-step evaluates Eqn. 6.14 to allocate the data points to the K models. The M-step is analogous to the M-step in the case of a nonparametric model. Given z_{ij} from Eqn. 6.14, the M-step maximizes $F(\theta)$. For efficiency's sake, our hybrid method performs these steps incrementally by alternately estimating the z_{ij} and performing a unidirectional search at each iteration until the algorithm converges. The theoretical basis for the success of such incremental EM algorithms is given by Lange (1995).

6.2.3.2 Parametric Output Model (Serial Models)

A parametric output model usually arises when unmeasured intermediates, such as the specific rates in the fed-batch penicillin fermentation process of Section 3.7, are used to compute the dependent values. Conservation equations such as mass and energy balances are common parametric output models. For our purposes we will extend our "standard" hypothesis stated in Section 6.2 with the assumptions that (1) unmeasured intermediates v exist, (2) they are related to x via a finite mixture distribution $p(v|x,\theta,H)$ with component pdfs $p(v|x,\theta,M_j,H)=\delta(v,f_j(x;\theta_j))$, $j=1,2,...,K$; and (3) they are related to the dependent variables through the pdf $p(y|x,v,\theta,H)=\phi(y;g(x,z;\theta),\Omega)$, where $g(x,z;\theta)$ is a known parametric model.

Therefore, the likelihood of the parameters is similar to the likelihood given in Eqn. 6.12:

$$p(X,Y|\theta,H) = \prod_{i=1}^{N} \int p(y_i|x_i,v_i,\theta,H)\,p(v_i|x_i,\theta,H)\,p(x_i|H)\,dv_i$$

$$= \prod_{i=1}^{N} p(x_i|H) \int \phi\left(y_i; g(x_i,v_i;\theta), \sigma^2 I + \Omega\right) \sum_{j=1}^{K} p(v_i|x_i,\theta,M_j,H)\,\omega_j(x_i)\,dv_i$$

$$= \prod_{i=1}^{N} p(x_i|H) \sum_{j=1}^{K} \phi\left(y_i; g\left(x_i, f_j(x_i;\theta_j);\theta\right), \sigma^2 I + \Omega\right)\omega_j(x_i).$$

The solution to the parameter estimation problem in this case is the same as the solution in the previous section except that the individual models $f_j(x;\theta_j)$ in the E- and M-steps of the parallel case are replaced by $g(x,f_j(x;\theta_j);\theta)$ in this case. Also, note that the same solution would have arisen if g(.) were linear in v and the hypothesis H had included the less restrictive assumption that pdfs $p(v|x,\theta,M=M_j,H)$ had expectations $f_j(x;\theta_j)$ rather than were certainly known to be $f_j(x;\theta_j)$. Also, the linear case solution can serve as the basis for an iterative solution for nonlinear g(.).

6.2.4 Semiparametric Models

Finally, all the ideas put forth in solving the above models are brought together in solving the most general of the modeling cases: the semiparametric model combining multiple parametric and multiple nonparametric models. As in the previous section, this section considers both parallel and serial model structures. These cases cover the various motivating examples; and thus the methods below – under the "standard" hypothesis – solve the parameter estimation problem when given the pdfs arising from unmeasured phenomena, process constraints, multiple models, extrapolation, process faults, and partitioned data sets. The only example not covered by this formalism is the corrupted data case, which we solve in Section 6.4.

6.2.4.1 Finite Mixture Distributions (Parallel Models)

In what follows, we will assume the existence of K models as usual; however, the first L will be parametric and the last K-L will be nonparametric. Each of the parametric state

dependence models will have K_k kernels, where $k=L+1,L+2,...,K$; $h=1,2,...,K_k$; and K_k not necessarily equal to K_g, for $k \neq g$. The likelihood of θ in such a case is as follows:

$$p(X,Y|\theta,H)=\prod_{i=1}^{N} p(y_i|x_i,\theta,H)p(x_i|H). \tag{6.15}$$

$$\text{where } p(y_i|x_i,\theta,H)=\sum_{j=1}^{L}\phi\left(y_i;f_j(x_i;\theta_j),\sigma^2 I+\Omega_j\right)\omega_j(x_i)$$
$$+\sum_{k=L+1}^{K}\sum_{h=1}^{K_k}\phi\left(y_i;f_{kh}(x_i;\theta_{kh}),\sigma^2 I\right)K(x_i;\alpha_{kh})\omega_k(x_i).$$

The surrogate objective function involving the indicator variables becomes a bit more complicated because now there are not only indicator variables for each model but also indicators for each kernel of each nonparametric model:

$$F(\theta;Z)=\sum_{j=1}^{L}F_j(\theta_j;Z_j)+\sum_{k=L+1}^{K}G_k(\theta_k;Z_k), \text{ where}$$

$$F_j(\theta_j;Z_j)\equiv\sum_{i=1}^{N}-\tfrac{1}{2}z_{ij}\left(y_i-f_j(x_i;\theta_j)\right)^T\left(\sigma^2 I+\Omega_j\right)^{-1}\left(y_i-f_j(x_i;\theta_j)\right)+\log\left[p(\theta_j|H)\right] \tag{6.16}$$

$$G_k(\theta_k;Z_k)\equiv\sum_{h=1}^{K_k}\sum_{i=1}^{N}-\tfrac{1}{2}z_{ikh}\left(y_i-f_{kh}(x_i;\theta_{kh})\right)^T\left(\sigma^2 I\right)^{-1}\left(y_i-f_{kh}(x_i;\theta_{kh})\right)+\log\left[p(\theta_{kh}|H)\right]$$

Because Eqn. 6.16 is linear in Z, the E-step estimates the indicators of the parametric models z_{ij} and those of the nonparametric models z_{ikh} by their conditional expectations using the following equations:

E-Step (parametric): $\forall i=1,2,...,N$; $j=1,2,...,L$

$$E\left[z_{ij}|x_i,y_i,\theta_j,H\right]=P\left(M_j|x_i,y_i,\theta_j,H\right)=\frac{\phi\left(y_i;f_j(x_i;\theta_j),\sigma^2 I+\Omega_j\right)\omega_j(x_i)}{p(y_i|x_i,\theta,H)} \tag{6.17}$$

E-Step (nonparametric): $\forall i=1,2,...,N$; $k=L+1,L+2,...,K$; $h=1,2,...,K_k$

$$E\left[z_{ikh}|x_i,y_i,\theta_k,H\right]=P\left(M_{kh}|x_i,y_i,\theta_k,H\right)=\frac{\phi\left(y_i;f_{kh}(x_i;\theta_{kh}),\sigma^2 I\right)K(x_i;\alpha_{kh})\omega_k(x_i)}{p(y_i|x_i,\theta,H)}. \tag{6.18}$$

And, again, as in the case of multiple parametric models, we use a hybrid or incremental EM algorithm to estimate the parameters θ by maximizing $F(\theta;Z)$ of Eqn. 6.16.

6.2.4.2 Parametric Output Model (Serial Models)

Parameter estimation in this case is related to that of the previous section in exactly the same way that the solutions of Sections 6.2.3.2 and 6.2.3.1 are related: the individual models $f_j(x;\theta_j)$ and $f_{kh}(x;\theta_{kh})$ in the E- and M-steps of the parallel model case are replaced by $g(x,f_j(x;\theta_j);\theta)$ and $g(x,f_{kh}(x;\theta_{kh});\theta)$ in the serial model case.

6.3 Prediction

As mentioned at the start of this chapter, there are two main approaches to state estimation: (1) obtaining a point estimate of θ, then evaluating the predictive model $f(x;\theta)$ to make the prediction; and (2) sampling the posterior pdf of θ, $p(\theta|D,H)$, then averaging the evaluations of the model at each sampled θ to make the prediction. Our research focused on the first approach to state estimation. The second approach was only briefly investigated by the experimentation described in Section 6.6.

In the first approach, the predictive model $f(x;\theta)$ is defined as the conditional expectation of the state, where the expectation is taken over the evidential dependence model evaluated at the optimal parameters. For our standard hypothesis of the previous sections and for general semiparametric models (e.g., Eqn. 6.15), the predictive model is a linear combination of the component functions weighted by their probabilities of applicability:

$$y_p = \sum_{j=1}^{K} f_j(x_i;\theta_j)P(M_j|x_i,H)$$

For example, the evidential dependence model used in Eqn. 6.15 yields the following predictive model:

$$y_p = \sum_{j=1}^{L} f_j(x_i;\theta_j)\omega_j(x_i) + \sum_{k=L+1}^{K}\sum_{h=1}^{K_k} f_{kh}(x_i;\theta_{kh})K(x_i;\alpha_{kh})\omega_k(x_i)$$

6.4 Solutions for Corrupted Data

Robust regression and data rectification (reconciliation), which are important applications in systems engineering, address state estimation when the data are corrupted. Regression in such cases requires the joint estimation of the parameters and rectification of the data. Furthermore, predicting with an error-in-variables regression model requires essentially the same solution methods as data rectification because both assume errors exist in the independent variable x as well as the dependent variable y. The major difference is that in prediction we are estimating the state at some time in the future while in rectification we are estimating the current or past states.

To make the regression robust, the same techniques are applied that are used in rectifying data with gross errors. For dynamic processes, the approach we present has similarities to techniques proposed in the context of outlier detection in time series, which were mentioned in Chapter 2. These approaches include maximum likelihood estimation (MLE) and Bayesian methods. In this section, the Expectation-Maximization formulas are applied to the problem of state estimation from corrupted data. For quick comparison, Table 6.1 lists the attributes of the estimation approach proposed in this work and those approaches mentioned in Section 2.1.3.

Table 6.1: Attributes of Selected Approaches

Reference	Approach	Outlier model
This research	Bayesian w/EM	probabilistic: mixture of Gaussians
Abraham and Chuang (1993)	MLE w/EM	probabilistic: mixture of Gaussians
Schick and Mitter (1994)	Bayesian (robust Kalman filtering)	probabilistic: general mixture
Ljung (1993); Chen and Liu (1993)	MLE w/ detection by tests	deterministic

The solution of Section 6.2.4 is extended to the corrupted data case by allowing for noise, gross errors and missing values in both the independent variables x and the dependent

variables y. Therefore, our standard hypothesis, which assumed that x was measured exactly, is modified by the substitution of an error model that captures the uncertainty in x. Additionally, we account for the possibility that some of the measurements of both x and y may be missing or may have gross errors. The error models we use are finite mixture distributions capturing the several sensor fault modes, as given by case 4 of Section 5.3.2: $\mathrm{p}(e|H)=\sum_{l=1}^{L}\mathrm{p}(e|F_l,H)\mathrm{P}(F_l|H)$. The resulting solution methods, presented below, will be general enough to solve all of the motivating examples we have presented earlier.

6.4.1 Nonparametric Density Estimation

As in Section 6.1, the objective here is to derive a nonparametric density estimator in the form of Eqn. 6.1. However, when $X\neq\tilde{X}$ we must jointly estimate the hyperparameters α and rectify the measurements. Therefore, the hypothesis H now includes the assumptions that $\tilde{X}+\mathcal{E}=X$; the errors $\mathcal{E}=\{e_{ik}\}$ (i^{th} observation, k^{th} variable) are iid from the mixture pdf $\mathrm{p}(e_{ik}|H)=\sum_{l=1}^{L}\mathrm{p}(e_{ik}|F_{kl},H)\mathrm{P}(F_{kl}|H)$; and the F_{kl} are independent of the models M_j. So, the problem statement is as follows:

Nonparametric Density Estimation Problem:

> Given data $D=X$ and hypothesis H, find true values $\tilde{X}$ and the parameters $\alpha\equiv\{a_j,q_j\}$, $j=1,2,\ldots,K$, that maximize $\mathrm{p}(\alpha,\tilde{X}|X,H)$.

The objective function to be maximized is proportional to $p(\alpha, \tilde{X} | X, H)$:

$$\begin{aligned} F(\alpha) &= p\left(\alpha, \tilde{X}, X | H\right) = p\left(X | \tilde{X}, H\right) p\left(\tilde{X} | \alpha, H\right) p(\alpha | H) \\ &= p(\alpha | H) \prod_{i=1}^{N} p(x_i | \tilde{x}_i, H) p\left(\tilde{x}_i | \tilde{X}_{\{<i\}}, \alpha, H\right) \\ &= p(\alpha | H) \prod_{i=1}^{N} \left[\prod_{k=1}^{n} \sum_{l=1}^{L} p(x_{ik} | \tilde{x}_{ik}, F_l, H) P(F_l | H) \right] \left[\sum_{j=1}^{K} p\left(\tilde{x}_i | \tilde{X}_{\{<i\}}, \alpha, M_j, H\right) P\left(M_j | H\right) \right]. \end{aligned}$$

Now, $F(\alpha, \tilde{X})$ has an L-component mixture to describe the L sensor fault modes for each of the n x variables. (Note, each $x_i \in \Re^n$.) This makes for a complex objective function.

Again, the inclusion of indicator variables allows a solution to be obtained efficiently using EM. So as before, we introduce the model indicator variables $Z=\{z_{ij}\}$, where $i=1,2,...,N$ and $j=1,2,...,K$. Moreover, for each of the n variables, we introduce a set of fault indicator variables $\zeta_k=\{\zeta_{kil}\}$, where $k=1,2,...,n$; $i=1,2,...,N$; and $l=1,2,...,L$. Each of the n ζ_k is an $N \times L$ matrix of zeros and ones: $\zeta_{kil}=1$ indicates that the sensor measuring the k^{th} variable of x_i was in fault mode F_{kl}, and for each x_{ik}, one and only one $\zeta_{kil}=1$. By definition, $P(\zeta_{kil} = 1 | x_{ik}, \tilde{x}_{ik}, H) = P(F_{kl} | x_{ik}, \tilde{x}_{ik}, H)$; and therefore, $E[\zeta_{kil} | x_{ik}, \tilde{x}_{ik}, H] = P(F_{kl} | x_{ik}, \tilde{x}_{ik}, H)$.

The surrogate objective function to be maximized in the M-step is the expectation of $\log F(\alpha, \tilde{X})$ conditioned on X, ζ, Z, and H, which is calculated in the E-step:

$$E\left[\log F\left(\alpha, \tilde{X}\right) | X, \zeta, Z, H\right] = \sum_{\zeta, Z} \log F\left(\alpha, \tilde{X}\right) P\left(\zeta, Z | X, \tilde{X}, \alpha^{(t-1)}, H\right).$$

The $\log F(\alpha, \tilde{X})$ is linear in both ζ and Z, therefore, the E-step simply computes the conditional expectations of ζ and Z:

E-Step: $\forall i=1,2,...,N$; $k=1,2,...,n$; $l=1,2,...,L$

$$E\left[\zeta_{kil} | X, \tilde{X}, \alpha, H\right] = P(F_{kl} | x_{ik}, \tilde{x}_{ik}, \alpha, H) = \frac{p(x_{ik} | \tilde{x}_{ik}, F_{kl}, \alpha, H) P(F_{kl} | H)}{p(x_{ik} | \tilde{x}_{ik}, H)}; \quad (6.19)$$

$\forall i=1,2,...,N;\ j=1,2,...,K$

$$\mathrm{E}\left[z_{ij}|x_i,\tilde{x}_i,\alpha,H\right]=\mathrm{P}\left(M_j|\tilde{x}_i,\alpha,H\right)=\frac{\mathrm{p}\left(\tilde{x}_i|M_j,\alpha,H\right)\mathrm{P}\left(M_j|H\right)}{\mathrm{p}\left(\tilde{x}_i|H\right)}$$
$$=\frac{q_j\phi\left(\tilde{x}_i;\mu_j,\Sigma_j\right)}{\sum_{l=1}^{K}q_l\phi\left(\tilde{x}_i;\mu_l,\Sigma_l\right)}. \tag{6.20}$$

The M-step maximizes E[logF($\alpha, \tilde{X}$)|X,ζ,Z,H] with respect to $\alpha=\{q_j,\mu_j,\Sigma_j\}$ and $\tilde{X}$, and with ζ and Z set to their expectations calculated in the E-step. Therefore, the optimal α has an analytical solution based on the latest estimate of $\tilde{X}$:

M-Step for $\alpha=\{q_j,\mu_j,\Sigma_j\}$: $\forall j=1,2,...,K$

$$q_j=\frac{1}{N}\sum_{i=1}^{N}z_{ij}\ ;\quad \mu_j=\frac{1}{N}\sum_{i=1}^{N}z_{ij}\tilde{x}_i\ ;\quad \Sigma_j=\frac{1}{N}\sum_{i=1}^{N}z_{ij}\left(\tilde{x}_i-\mu_j\right)^2. \tag{6.21}$$

Conversely, the optimal $\tilde{X}$ must in general be solved using a mathematical programming routine that alternately calculates α by Eqn. 6.21 and updates $\tilde{X}$ to increase E[logF($\alpha, \tilde{X}$)|X,ζ,Z,H].

Due to the universal approximation property of a mixture of normal distributions and the tractable mathematics that result, it is convenient to let each fault model's pdf be a normal distribution. Then the pdf for the k^{th} variable in the i^{th} observation is $\mathrm{p}(x_{ik}|\tilde{x}_{ik},F_{kl},H)=\phi(x_{ik};\tilde{x}_{ik}+\nu_{kl},\sigma_{kl}^2)$, which allows each component to have a bias ν_{kl} and variance σ_{kl}^2. An example of applying this type of sensor fault distribution is the work of Johnston and Kramer (1995), which performs data rectification.

Using normal distributions for the sensor faults admits an analytical solution for the optimal $\tilde{X}$ as well as α. Therefore, the M-step can be executed quickly by cycling between Eqn. 6.21 and Eqns. 6.22 and 6.23 below. Under the assumption of normal components in the error model, the E-step for ζ becomes the following:

E-Step: $\forall i=1,2,...,N;\ k=1,2,...,n;\ l=1,2,...,L$

$$\mathrm{E}\left[\zeta_{kil} \mid X, \tilde{X}, \alpha, H\right] = \mathrm{P}\left(F_{kl} \mid x_{ik}, \tilde{x}_{ik}, \alpha, H\right) = \frac{\mathrm{p}\left(x_{ik} \mid \tilde{x}_{ik}, F_{kl}, \alpha, H\right)\mathrm{P}\left(F_{kl} \mid H\right)}{\mathrm{p}\left(x_{ik} \mid \tilde{x}_{ik}, H\right)}$$

$$= \frac{\mathrm{P}\left(F_{kl} \mid H\right)\phi\left(x_{ik}; \tilde{x}_{ik} + \nu_{kl}, \sigma_{kl}^2\right)}{\sum_{h=1}^{L} \mathrm{P}\left(F_{kh} \mid H\right)\phi\left(x_{ik}; \tilde{x}_{ik} + \nu_{kh}, \sigma_{kh}^2\right)}. \tag{6.22}$$

The M-step for $\tilde{X}$ follows:

M-Step for $\tilde{x}_i$: $\forall i=1,2,...,N$

$$\mathrm{Q}(\alpha,\zeta,Z)^{-1}\tilde{x}_i = \mathrm{R}(\zeta)^{-1}\left(x_i - \mathrm{u}(\zeta)\right) + \mathrm{S}(\alpha,Z)^{-1}\,\mathrm{m}(\alpha,Z), \tag{6.23}$$

where the matrices and vectors in Eqn. 6.23 are defined as

$$\mathrm{Q}(\alpha,\zeta,Z)^{-1} \equiv \mathrm{S}(\alpha,Z)^{-1} + \mathrm{R}(\zeta)^{-1}; \quad \mathrm{S}(\alpha,Z)^{-1} \equiv \sum_{j=1}^{K} z_{ij}\Sigma_j^{-1}; \quad \mathrm{u}(\zeta) = r_{kk}(\zeta)\sum_{l=1}^{L}\left(\zeta_{kil}\nu_{kl}/\sigma_{kl}^2\right);$$

$\mathrm{m}(\alpha,Z) \equiv \mathrm{S}(\alpha,Z)\sum_{j=1}^{K} z_{ij}\Sigma_j^{-1}\mu_j$; and $\mathrm{R}(\zeta)$ is an $n \times n$ diagonal matrix with elements

$$r_{kk}(\zeta) = \left[\sum_{l=1}^{L}\left(\zeta_{kil}/\sigma_{kl}^2\right)\right]^{-1}.$$

(Again, note that i indexes the N observations $X=\{x_i\}$; k indexes the n variables in each x_i; j indexes the K models in the density estimator; and l indexes the L sensor fault modes in the measurement error pdf.)

Missing values are managed by deriving the EM formulas as above while augmenting the E-step to estimate each missing measurement x_{ik} along with the indicator variables. When normal mixtures are used as the sensor fault models, the result is that, if measurement x_{ik} is missing, then $\tilde{x}_i$ is calculated using Eqn. (6.23) while setting to zero the k^{th} element of the term $\mathrm{R}(\zeta)^{-1}[x_i\text{-}\mathrm{u}(\zeta)]$. This corresponds to letting σ_{kl}^2 go to infinity for each of the L fault modes of the k^{th} variable.

Implementation of this solution method for the 2-component fault models used by Johnston and Kramer (1995) in data rectification is presented in Section 6.7. That section derives the initial guesses and illustrates the details of the EM iterations.

6.4.2 Parameter Estimation and Prediction

Once density estimation for $p(\tilde{x}|\alpha,H)$ has been performed, the model parameters θ must be estimated. In the context of corrupted data, the y-data must also be rectified. Plus, the rectification of the x-data, which was performed while nonparametrically estimating α, can be improved upon by including information from the y-data.

In this context, the modeling hypothesis H includes the assumptions of the previous section plus $\tilde{Y}+\mathcal{E}_y=Y$; the errors $\mathcal{E}_y=\{d_{ik}\}$ (i^{th} observation, k^{th} variable) are iid from the mixture pdf $p(d_{ik}|H)=\sum_{l=1}^{L} p(d_{ik}|G_{kl},H)P(G_{kl}|H)$; and the G_{kl} are independent of the models M_j.

The parameter estimation problem in this context is the following:

Parameter Estimation Problem:

> Given hypothesis H and data $D=\{X,Y\}$, under 0-1 loss function, find the parameters θ and the true values $\tilde{X}$ and $\tilde{Y}$ that maximize $p(\theta,\tilde{X},\tilde{Y}|X,Y,H)$.

We solve this problem using EM. We inject the model indicator variables Z and x-variable fault indicators ζ as in Section 6.4.1. Also, for each of the m y variables, we introduce fault indicator variables $\xi_k=\{\xi_{kil}\}$, where $k=1,2,...,m$; $i=1,2,...,N$; and $l=1,2,...,L$. Each of the m ξ_k is an $N\times L$ matrix of zeros and ones: $\xi_{kil}=1$ indicates that the sensor measuring the k^{th} variable of y_i was in fault mode G_{kl}, and for each y_{ik}, one and only one $\zeta_{kil}=1$. As usual, by definition, $P(\xi_{kil}=1|y_{ik},\tilde{y}_{ik},H)=P(G_{kl}|y_{ik},\tilde{y}_{ik},H)$; and therefore,

$$E[\xi_{kil}|y_{ik},\tilde{y}_{ik},H]=P(G_{kl}|y_{ik},\tilde{y}_{ik},H).$$

The E-Step calculates the conditional expectation of $\log[p(\theta,\tilde{X},\tilde{Y}|X,Y,Z,\zeta,\xi,\alpha,H)]$, which serves as the surrogate objective function. Again, this objective function is linear in Z, ζ

and ξ, so the E-step computes the expected values of these quantities, and the M-step maximizes $\log[p(\theta, \tilde{X}, \tilde{Y} | X, Y, Z, \zeta, \xi, \alpha, H)]$ evaluated at these expected values.

Assuming finite normal distributions to describe the error pdfs for x as before and for y as $p(y_{ik} | \tilde{y}_{ik}, G_{kl}, H) = \phi(y_{ik}; \tilde{y}_{ik} + \gamma_{kl}, \eta_{kl}^2)$ gives the following EM formulas for θ, $\tilde{Y}$, Z and ξ:

E-Step: $\forall i=1,2,\ldots,N;\ k=1,2,\ldots,n;\ l=1,2,\ldots,L$

$$E[\xi_{kil} | X, Y, \tilde{X}, \tilde{Y}, \theta, \alpha, H] = P(G_{kl} | y_{ik}, \tilde{y}_{ik}, H) = \frac{p(y_{ik} | \tilde{y}_{ik}, G_{kl}, H) P(G_{kl} | H)}{p(y_{ik} | \tilde{y}_{ik}, H)}$$
$$= \frac{P(G_{kl} | H) \phi(y_{ik}; \tilde{y}_{ik} + \gamma_{kl}, \eta_{kl}^2)}{\sum_{h=1}^{L} P(G_{kh} | H) \phi(y_{ik}; \tilde{y}_{ik} + \gamma_{kh}, \eta_{kh}^2)}. \quad (6.24)$$

$\forall i=1,2,\ldots,N;\ j=1,2,\ldots,K$

$$E[z_{ij} | X, Y, \tilde{X}, \tilde{Y}, \theta, \alpha, H] = P(M_j | \tilde{x}_i, \tilde{y}_i, \theta, \alpha, H) = \frac{p(\tilde{x}_i, \tilde{y}_i | \theta, M_j, \alpha, H) P(M_j | H)}{p(\tilde{x}_i, \tilde{y}_i | \theta, \alpha, H)}. \quad (6.25)$$

Mathematical programming estimates θ by maximizing the following function (which is a linear function of $\log[p(\theta | \tilde{X}, X, Y, Z, \xi, H)]$) evaluated at the expectation of Z and ξ from the E-step and at the latest estimate of $\tilde{X}$. The posterior pdf $p(\theta | \tilde{X}, X, Y, Z, \xi, H)$ can be represented as follows by applying the formulas in Appendix B:

$$p(\theta | \tilde{X}, X, Y, Z, \xi, H) \propto \prod_{i=1}^{N} \int p(y_i | \tilde{y}_i, \xi_i, H) p(\tilde{y}_i | Y_{\{<i\}}, \tilde{x}_i, \theta, z_i, H) d\tilde{y}_i$$
$$\propto \prod_{i=1}^{N} \int \phi(\tilde{y}_i; y_i - u(\xi_i), R(\xi_i)) \phi(\tilde{y}_i; m(\tilde{x}_i, Y_{\{<i\}}, \theta, z_i), S(z_i)) d\tilde{y}_i$$
$$\propto \prod_{i=1}^{N} \phi(y_i; u(\xi_i) + m(\tilde{x}_i, Y_{\{<i\}}, \theta, z_i), R(\xi_i) + S(z_i)),$$

where $u(\xi_i) = r_{kk}(\xi_i) \sum_{l=1}^{L} (\xi_{kil} \gamma_{kl} / \eta_{kl}^2)$; $m(\tilde{x}_i, \tilde{Y}_{\{<i\}}, \theta, z_i) \equiv S(z_i) \sum_{j=1}^{K} z_{ij} \Omega_j^{-1} f_j(\tilde{x}, \tilde{Y}_{\{<i\}_i}; \theta_j)$;

$S(z_i)^{-1} \equiv \sum_{j=1}^{K} z_{ij} \Omega_j^{-1}$; and $R(\xi_i)$ is $n \times n$ diagonal with elements $r_{kk}(\xi_i) = \left[\sum_{l=1}^{L} \xi_{kil} / \eta_{kl}^2 \right]^{-1}$.

Therefore, the objective function for estimating θ is the following:

M-Step for θ:

$$F(\theta;Z,\xi)=\sum_{i=1}^{N}-\tfrac{1}{2}\left(y_i-y_i^*\right)^T\left[R(\xi_i)+S(z_i)\right]^{-1}\left(y_i-y_i^*\right)+\log[p(\theta|H)], \tag{6.26}$$

where $y_i^* \equiv u(\xi_i)+m\left(\tilde{x}_i,\tilde{Y}_{\{<i\}},\theta,z_i\right)$.

Estimates of the true values, $\tilde{Y}$, are found by maximizing $\log[p(\tilde{Y}\,|\,\tilde{X},\theta,Y,Z,\xi,H)]$:

$$\begin{aligned}
p\left(\tilde{Y}|\tilde{X},\theta,Y,Z,\xi,H\right) &\propto \prod_{i=1}^{N} p(y_i|\tilde{y}_i,\xi_i,H)\,p\left(\tilde{y}_i|\tilde{Y}_{\{<i\}},\tilde{x}_i,\theta,z_i,H\right)\\
&\propto \prod_{i=1}^{N}\phi\left(\tilde{y}_i;y_i-u(\xi_i),R(\xi_i)\right)\phi\left(\tilde{y}_i;m\left(\tilde{x}_i,\tilde{Y}_{\{<i\}},\theta,z_i\right),S(z_i)\right)\\
&\propto \prod_{i=1}^{N}\phi\left(\tilde{y}_i;Q(\xi_i,z_i)\left[R(\xi_i)^{-1}\left(y_i-u(\xi_i)\right)+S(z_i)^{-1}\,m\left(\tilde{x}_i,\tilde{Y}_{\{<i\}},\theta,z_i\right)\right],Q(\xi_i,z_i)\right),
\end{aligned}$$

where $Q(\xi_i,z_i)^{-1}\equiv S(z_i)^{-1}+R(\xi_i)^{-1}$. Therefore, $\tilde{Y}$ is estimated by solving the following linear algebraic equation for each point:

M-Step for $\tilde{y}_i$: $\forall i=1,2,\ldots,N$

$$Q(\xi_i,z_i)^{-1}\tilde{y}_i = R(\xi_i)^{-1}\left(y_i-u(\xi_i)\right)+S(z_i)^{-1}\,m\left(\tilde{x}_i,\tilde{Y}_{\{<i\}},\theta,z_i\right). \tag{6.27}$$

Along with estimating θ, $\tilde{Y}$, Z and ξ using the above equations, the full EM algorithm for simultaneous parameter estimation and rectification uses Eqns. 6.21, 6.22, and 6.23 to re-estimate α, ζ and $\tilde{X}$ because Z changes on each iteration. Note that once the parameters θ and hyperparameters α are fixed at their optimum values, Eqns. 6.22 to 6.25 and 6.27 serve as the basis for a data rectification EM algorithm or equivalently, a prediction algorithm for an error-in-variables model.

Also, missing y values are treated similarly to missing x values. If measurement y_{ik} is missing, then $\tilde{y}_i$ is calculated using Eqn. (6.27) while setting to zero the k^{th} element of the

term $R(\xi)^{-1}[y_i-u(\xi)]$. This corresponds to letting η_{kl}^2 go to infinity for each of the L fault modes of the k^{th} variable.

The solution methods presented in this section extend the generality of the methods of Section 6.5 by accounting for corrupted data. This represents a significant enhancement in the robustness of process modeling over that of the function-oriented approach of Chapter 3. Of course, the function-oriented approach could be augmented with one of the wide variety of outlier detection and error-in-variable techniques developed for classical statistics. However, the developments above illustrate that in the probabilistic paradigm, the additional information about the data can be readily incorporated into model synthesis in a unified framework and in a tractable manner. As a consequence, implementation of the solution methods for corrupted data is a manageable extension to that of the simpler cases. Section 6.7 demonstrates this by presenting the implementation of these algorithms in Matlab.

6.5 Hypothesis Selection

The decision problem for hypothesis selection was presented in Section 4.3.4 as follows:

Hypothesis selection problem:

> Given data D and a set of hypotheses $\{\mathcal{H}: H_i, i=1,2,...,N_{\mathcal{H}}\}$, select the hypothesis $H_i \in \mathcal{H}$ that maximizes the marginal distribution of the data, $p(D|H_i)$ or equivalently, that maximizes $\log p(D|H_i)$.

To achieve this maximization we must identify the candidate hypotheses H_i and estimate $\log p(D|H_i)$ for each of them. However, calculation of $p(D|H)$ usually involves the multi-dimensional integration necessary to marginalize the joint pdf $p(D,\theta|H)$, which is the product of the likelihood $p(D|\theta,H)$ and the prior $p(\theta|H)$. Below, we present two

approximations to p(D|H) that take advantage of the nice mathematical properties of the normal distribution.

6.5.1 Analytical Solution for Mixtures of Normal Distributions

Under our standard hypothesis as described in Section 6.1 and using formulas to manipulate mixtures of normals (Appendix B), the likelihood p(D|θ,Z,H) can be expressed as a multivariate normal distribution. Also, if the prior on θ has an exponent that is quadratic or linear in θ, then the resulting joint pdf p(D,θ|Z,H) will be a normal distribution in θ. The result is as follows:

$$p(D,\theta|Z,H)=\frac{p(D,Z|\theta,H)p(\theta|H)}{p(Z|H)}=p(\theta|D,Z,H)p(D|Z,H) \tag{6.28}$$

Therefore, p(D|H) can be readily approximated by integrating Eqn. 6.28 with respect to θ to get p(D|Z,H):

$$p(D|Z,H)=p(Y|Z,X,H)p(X|H) \tag{6.29}$$

Then, summing p(D|Z,H)p(Z|H) over all K^N combinations of the N z_j gives p(D|H). However, because such a summation would be expensive for even modest K and N, we seek a more computationally tractable estimate.

The EM algorithm suggests an alternative to summing over Eqn. 6.29 by attempting to directly approximate log p(D|H). In EM, we maximize an objective function log p(α|H) by maximizing the conditional expectation of log p(α|Z,H). Moreover, when log p(α|Z,H) is linear in Z, the expectation of the function equals the function evaluated at the expectation of Z. So, similarly, we propose using the expectation of log p(D|Z,H) as an approximation to log p(D|H) in hypothesis selection. Then, we approximate E[log p(D|Z,H)] by log p(D|Z=E[Z|D,H],H), where E[Z|D,H] is the estimate for Z obtained from the E-step during parameter estimation. Furthermore, evaluating log p(D|Z,H) at E[Z|D,H] is exactly equal to E[log p(D|Z,H)] in the case that log p(D|Z,H) is linear in Z.

6.5.2 Approximation of the Posterior by a Normal Distribution

An alternative to the above approach, is to approximate the log of the joint pdf $p(D,\theta|H)$ by its second-order Taylor series around the optimal parameters θ^*. This Taylor series and the appropriate normalizing constant are then combined into a normal distribution (the expansion serves as the exponent along with terms to complete the square). Then, the integration over θ to get $p(D|H)$ is readily performed analytically. MacKay (1992b) presents such an approach and derives the following formula for $p(D|H)$:

$$p(D|H) \approx p(D|\theta=\theta^*,H)\left[p(\theta|H)_{\theta^*}\cdot\sqrt{(2\pi)^n/|S|}\right], \tag{6.30}$$

where S is the Hessian of the log-posterior, $\log p(\theta|D,H)$.

MacKay (1992a, 1992b) refers to the bracketed term in Eqn. 6.30 as the "Occam factor." Thus, $p(D|H)$ is a product of the likelihood and the Occam factor. The likelihood arises directly from the goodness of fit: the more accurately the model describes the training data the greater the likelihood. Conversely, the Occam factor is proportional to the collapse of the hypothesis space when presented with the data. The hypothesis space refers to the accessible volume in parameter space. The hypothesis space is representative of the expressiveness of the model. If the best-fit likelihoods for each of two models are approximately equal, the larger Occam factor of the simpler model will give it greater evidence. Thus, $p(D|H)$ incorporates the fundamental tradeoff between goodness of fit (model accuracy on the training data) and generalization (model accuracy on future data).

6.6 Markov Chain Monte Carlo Approaches

As mentioned in Section 2.1.2, Markov chain Monte Carlo (MCMC) approaches are an alternative to the parameter estimation methods described above. A few MCMC codes from academic research groups are publicly available. We obtained two and performed a brief series of trial experiments. We also implemented the Gibbs sampler of Albert and Mueller, *et al.* (1992) for density estimation with Dirichlet priors on normal mixtures. Our experiences with the Gibbs sampler using Dirichlet priors and the results of our MCMC experiments are described below.

6.6.1 Markov Chain Monte Carlo using Dirichlet Priors

An alternative to the EM algorithm for nonparametric density estimation is the sampling method presented by Mueller, *et al.* (1992). They use a Dirichlet distribution to specify a prior on the *family* of multivariate normal distributions. The Dirichlet prior assigns a single generating distribution to each observation. To the extent that multiple observations are assigned to the same distribution, the resulting mixture may have significantly fewer components than number of data points. However, this specification allows the number of components K to range from one to the number of observations N.

At each iteration, the Gibbs sampling they propose randomly assigns the N observations, in essence establishing a configuration of components. Unfortunately, coupling the sampling of the configuration with the possibility of rejecting the new configuration and parameters at each iteration of the algorithm makes the computation required for such an approach orders of magnitude greater than that of the EM methods of the previous sections. After implementing this algorithm in Matlab and briefly experimenting with it, we rejected it in favor of EM because of its long computation times relative to EM.

6.6.2 Experience with Two Markov Chain Monte Carlo Codes

We briefly experimented with two publicly available MCMC codes. However, due to the nature of the algorithms, implementations, and the available time, we limited the study to

execution of the example problems supplied with the codes and simple tests to assess their applicability to mixture distribution problems. No attempt was made to benchmark the methods because of the very different environments and problems.

We drew the following conclusions:

- Gibbs sampling implemented in the BUGS system by Spiegelhalter *et al.* (1995): We tested the PC version of BUGS, which includes a wide variety of probability density functions, but is restricted to univariate problems. Multivariate problems can be solved in this implementation if the necessary pdfs can be posed as combinations of univariate problems. Only small (N=50) univariate problems with mixtures (K=2) were attempted. In tests of nonparametric density estimation sampling times were extremely long (one to two orders of magnitude) relative to our parameter estimation techniques. This was primarily due to the low frequency with which the Gibbs samplers sampled from high-probability-density regions of the distribution. It is believed that larger problems would quickly become infeasible for this implementation of Gibbs sampling.

- Hybrid Markov chain Monte Carlo simulation implemented as a C program by Neal (1995): This code worked well on Neal's examples of Bayesian learning networks (discussed in Chapter 2). Training times were comparable to times of similar sized back-propagation neural networks using nonlinear programming parameter estimation methods describe in Chapter 3. The major disincentive for using the MCMC engine provided by Neal in our work was the need to write custom C code that matched the data structures and calling protocols of the engine while maintaining the flexibility needed to model a wide variety of problems. Such a programming project was infeasible given the

priorities of this research, the time constraints, and the late acquisition of the code.

Along with the computational expense, the interpretability of the results from these routines was a drawback. For the Gibbs samplers and the hybrid MCMC method, determination of whether the resultant sample was representative enough for accurate estimation is nontrivial. Issues such as assessing the correlation of the members of the sampled sequence of estimates, determining if the sequence has reached equilibrium, and determining if the sample size was sufficient to accurately estimate the state required the interpretation of several statistics – like the ACF (autocorrelation function) – that varied for each implementation and were sensitive to the parameters (hyperparameters) that defined each run.

Despite the above disadvantages, MCMC methods – especially, the more elaborate hybrid dynamical simulation method of Neal (1995) – are practically the only recourse for solving many difficult continuous-variate probabilistic estimation problems. When tactics such as approximating distributions by mixtures of normals and obtaining point estimates of the parameters fail to produce accurate estimates, it may well be worth the additional effort to use a hybrid MCMC method. Also, as available computing power continues to dramatically increase each year, the computational expense of MCMC methods becomes less of a disincentive for implementing them.

6.7 Implementation

The solution methods presented in Sections 6.1 to 6.5 were implemented as functions in Matlab, the matrix and numerical analysis software created by The Mathworks, Inc. In keeping with the convention of naming such Matlab suites "toolkits," we call this set of functions the "Probabilistic Modeling Toolkit for Matlab." This section briefly

summarizes the Toolkit functions. Chapter 7 describes the application of the Toolkit to example case studies. The source code for the functions appear in Appendix D.

6.7.1 Nonparametric Density Estimation

The EM algorithm described in Section 6.1 was implemented to solve the nonparametric density estimation problem for determining the hyperparameters of Eqn. 5.2. The code is written for kernel distributions that are multivariate normal with means μ_j and covariance matrices Σ_j. The algorithm estimates these distribution parameters given an *a priori* value for the number of kernel distributions (or clusters) K, which should be validated after the fact by hypothesis selection methods. The basic algorithm executes the following steps:

(1) Initialize the indicators z_{ij} using K-means clustering, which determines strict 0-1 membership, setting $z_{ij}=1$ if the i^{th} point has been assigned to the j^{th} cluster.
(2) Perform the M-step of Eqn. 6.4 to estimate q_j, μ_j and Σ_j.
(3) Check if convergence has been achieved by comparing μ_j of the current iteration with that of the past iteration. If so, STOP. Else, continue.
(4) Perform the E-step of Eqn. 6.3 to estimate the z_{ij} of the next iteration.
(5) Go to Step 2.

In practice, to make this algorithm robust, provisions must be taken to account for clusters with too few points assigned to them to compute the required statistics in Eqn. 6.4 and for singular covariance matrices. The first problem arises because for $x \in \Re^n$, i.e., n-variate, each cluster should have more than n members to compute Σ_j, yet it is possible that less than n members be assigned during solution. It is resolved by simply deleting any cluster with too few members and assigning their members to other clusters. The second problem is resolved by representing any kernel distribution with singular Σ_j as a singular multivariate normal distribution given by the formulas of App. B.4.

So, our implementation extends the basic algorithm by executing the following steps:

(1) Initialize the indicators z_{ij} using K-means clustering.

(2) Check if each cluster has enough members: $\forall j = 1,\ldots,K$: $\sum_{i=1}^{N} z_{ij} > n$. If not, delete the cluster and its associated z_{ij}, decrement K, and renormalize the indicator matrix Z.

(3) Perform the M-step of Eqn. 6.4 to estimate q_j, μ_j and Σ_j.

(4) Check if Σ_j is singular for any cluster. If Σ_j is singular, use the singular normal distribution formulas of App. B.4 and flag that kernel as singular.

(5) Check if convergence has been achieved by comparing μ_j of the current iteration with that of the past iteration. If so, STOP. Else, continue.

(6) Perform the E-step of Eqn. 6.3 to estimate the z_{ij} of the next iteration, but use the singular normal distribution formula for any kernel that was flagged as singular in Step 4.

(7) Go to Step 2.

6.7.2 Parameter Estimation

Mathematical programming methods include constrained and unconstrained NLP and SIP, which were already discussed in Section 3.4. The main routine is function *train.m*, which implements a general nonlinear programming (NLP) algorithm that adaptively selects between steepest descent, conjugate gradient and quasi-Newton methods. The other functions set up the optimization problem prior to calling *train.m*. These include performing nonparametric density estimation, as described in the previous section, to estimate the hyperparameters.

The EM solution for general parameter estimation in robust regression was presented in Section 6.4.2. Function *nlpem.m* implements this method. It partitions $F(\theta)$ of Eqn. 6.26 into the $F_j(\theta_j)$ subobjectives and alternatively performs an E-step to allocate data points to each F_j and an M-step that executes a unidimensional (line) search in the appropriate

direction of θ-space to maximize F(θ). The current implementation uses the same NLP routine of *train.m* on the full objective function. The code is written for the general case in which the full F(θ) is maximized rather than separately maximizing the individual F_j because of the possibility of cross-model correlation among parameters of the different models.

6.7.3 Prediction

Prediction of the state is performed by the routine *predictp.m*. The routine implements the general solution of Eqn. 6.27, which can handle corrupted data and missing values as well as the common case of no errors in the x values. When the data are uncorrupted, routine simply evaluates each of the component models and weights them with their appropriate probabilities before summing them. Along with the predictions, the probabilities of applying each model are returned.

6.7.4 Hypothesis Selection

The Toolkit also includes a routine called *pdh.m* that computes P(D|Z,H) to assist in hypothesis selection. It uses Eqn. 6.29, which is exact when (1) the models M_j are linear in θ and have uncertainty represented by normal distributions and (2) the prior p(θ|H) has an exponent linear or quadratic in θ. When these conditions do not hold, the result returned by the routine will be an approximation to P(D|Z,H).

Chapter 7

Demonstration of the Methodology

The purpose of this chapter is twofold. First, it demonstrates in a step-by-step manner the application of the modeling methodology we propose. Second, it illustrates the utility and properties of the probabilistic modeling paradigm by presenting the results of two case studies. In the next section, a step-by-step recipe for applying the methodology is given. Simply due to the nature of modeling, a generic prescription is too abstract to be useful unless put in the context of actual applications. This chapter adds concreteness to the generic step-by-step recipe by showing its application and performance in case studies.

The first case study illustrates the probabilistic methodology in compensating for corrupted data in dynamic processes by combining time series analysis with robust estimation of the temperature profile in an actual fermentation. The second case study extends this approach in modeling the fed-batch penicillin fermentation investigated in the function-oriented approach of Chapter 3. It augments the difficulty of that problem by accounting for corrupted multivariate dynamic data. Together these case studies demonstrate (1) the unified representation of uncertainty from many types of sources, (2) the robustness offered by this representation, (3) the feasibility of applying the pragmatic solution methods of our modeling methodology, and (4) the benefits this approach offers over a function-oriented approach in the form of improved prediction accuracy and interpretability of performance metrics and process diagnostics that quantify model behavior.

7.1 Flowchart of the Methodology

The methodology we propose relies heavily on probability theory for its basis. Yet, it is tempered with enough pragmatism to readily represent process knowledge and to efficiently solve the resulting state estimation problems. Below we list the steps of this

methodology. Of course, much of it coincides with the thorough analysis of process behavior and data that is common to statistical modeling and traditional approaches to system identification (e.g., Ljung, 1987; Box, *et al.*, 1978; etc.). Also, much of it arises from methods proven to be effective in Bayesian estimation (e.g., Berger, 1985; Box and Tiao, 1974; etc.). These aspects of the methodology are mentioned for completeness but are not elaborated upon in any detail because ample references exist to fill in the gaps.

What is stressed here are the techniques necessary to apply the methodology presented in Chapter 6. This includes interpretation of the quantities that help us to iteratively evolve a model. For example, in a nonparametric model built from component models M_j, $j=1,2,...,K$, the probability of model applicability $z_{ij}=P(M_j|x_i,y_i,H)$ weights the contribution of model M_j at point i; therefore, if the sum of the z_{ij} for a given model M_j is zero or very small compared to the sum for other models, then M_j may be unnecessary. Consequently, a new model with fewer components would probably predict as well.

In this chapter, we break down the methodology into five steps summarized in Figure 7.1. The methodology is necessarily iterative, and it is the adequacy of the model at meeting the project goal and the cost of continuing the modeling project that determine when to stop. Fortunately, good decisions can be made during this iterative process due to the ample information available in the form of the diagnostic metrics described in Section 7.1.5. The steps of the methodology are described in detail here.

7.1.1 Plot and Analyze the Data

Every modeling project requires that the data be suitable for model synthesis. A standard technique in making this determination is the plotting of the data – i.e., data visualization. Coupling plots with the analysis of summary statistics, like the sample mean and correlation matrix, helps a modeler decide what types of distributions are appropriate and which variables are correlated with each other. Standard texts on statistical methods, such as Box and Hunter (1978) describe techniques to assist in this step. Key issues are (1) determining to what extent the data cover the operating space over which the model will

be used to make prediction; (2) deciding whether the data are satisfactorily described by the canonical distributions; (3) checking that the data are appropriate for nonparametric methods – e.g., assuring they have satisfactory spread; and (4) identifying obvious outliers or corrupted data. These issues will not be detailed here because they are described by other authors.

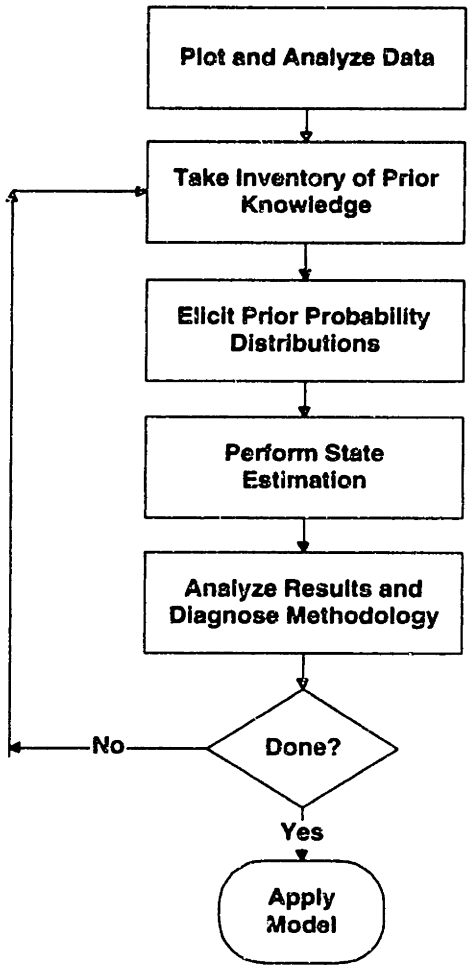


Figure 7.1: Flowchart of the Probabilistic Modeling Methodology

7.1.2 Take Inventory of the Prior Knowledge

The next step is to explicitly account for the relations that are known to exist between process variables. Process operating knowledge and the first principles of engineering and the physical sciences specify constraints that govern the process. These constraints and their relevance to the modeling objectives should be explicitly stated. Depending on the

complexity of the process, this step may necessitate a large-scale formal system analysis of the process or structured knowledge engineering sessions with the engineers, scientists and process operators most knowledgeable about process operation. The tactics used are strongly problem-dependent, but generally require determining the attributes needed to perform the role assignment discussed in Chapter 5. Gathering this information can be driven by analyzing the relations implied by the pdfs of Figure 4.1. A modeler who follows the decomposition of the major pdfs shown there should achieve a systematic accounting of the relations relevant to the quantities to be estimated.

7.1.3 Elicit the Prior Probability Distribution Functions

The next step is the representation as probability density functions of the knowledge gathered in Step 2. This role assignment procedure is described in Chapter 5. By examining the probabilistic representation of knowledge for the motivating examples presented there, a modeler should be able to pattern his or her pdfs after those shown. Notably, to maintain tractability as the problem complexity increases, it is useful to use canonical distributions and their mixtures unless there are compelling reasons not to. The solution methods presented in Chapter 6 assume that mixtures of normal distributions are adequate for process modeling. For the most part this is true, but when it is not, either simplifying assumptions should be made in order to justify using normal distributions or the more computationally intensive Markov chain Monte Carlo methods may be required.

Other issues, such as reducing the dimensionality of the data, are important, too. For example, in the fed-batch penicillin case study of Section 7.3, we generate a nonparametric model in the 3-dimensional space of state variables at time t-1 rather than in the 6-dimensional space that includes the three exogenous variables. Even though the 6-D space adds more information, considering the sparseness of the data to begin with, nonparametric methods would be expected to perform more poorly in 6-D than 3-D. This judgement was made *a priori*, but a 6-D model could have been attempted and then the ranks of the component covariances could have been analyzed to determine whether the

components collapsed back to a lower dimensional space based on correlation and/or sparsity of information in the higher dimensional space. This analysis of the component hyperparameters is a useful step for deciding the next hypothesis to pose during evolutionary model synthesis.

7.1.4 Perform State Estimation

The implementation described in Chapter 6 provides reliable and efficient tools for state estimation. The source codes for the routines are provided in Appendix D. They are sufficiently commented that a modeler should be able to readily apply them to problems similar to the case studies presented below.

A key issue is the use of robust nonparametric density estimation. A modeler must either ensure that sufficient data exist from which to estimate the hyperparameters of the finite mixture distributions or provide ample prior knowledge to essentially place and size the normal pdfs. Fortunately, by the nature of Bayesian estimation, the methodology automatically weights the data and the prior knowledge appropriately. However, it is still important that the modeler use the diagnostic methods of the next step to assess the influence of the injected knowledge.

7.1.5 Analyze the Results and Diagnose the Methodology

Several important diagnostic metrics are computed while estimating the parameters and true process variables. These metrics include the indicator variables Z, ζ, and ξ, which denote the posterior probabilities of each model's applicability and of the different fault modes of the independent and dependent variables, respectively. The following sections discuss these metrics and present examples of their interpretation and utility.

7.1.5.1 Probability of Model Applicability

Each z_{ij} in Z indicates the probability that model M_j is applicable at point i given x_i and y_i. As mentioned above, this gives an indication of how useful each model is in estimating the

state within a given operating regime. In fact, plotting the function $z_{ij}=P(M_j|x,y,H)$ versus subsets of x and y allows one to explicitly delineate the precise operating regime over which each model M_j applies. Assessing such plots, which of course are limited to bivariate renderings (thus, 3-D plots), provides a modeler with an indication of the reasonableness of each model. Moreover, if z_{ij} is small for all non-default models in a region that is particularly important in process operation, then it may be necessary to obtain more data so that the region is satisfactorily covered. Alternatively, such a case may indicate an inadequacy in the forms of the models that have been posed and thus, provide information on posing a new hypothesis.

During prediction, if the z_{ij} are small or dominated by the zij for the default model at a given x, then the point is relatively novel with respect to the database from which the model was synthesized. This is an indication that the model is extrapolating and caution should be used in applying the model results. In fact, when applied in a real-time online mode for steady-state processes, this feature is useful for flagging when the process has gone "out of control" in a statistical process control sense. This can provide process operators with a valuable warning that the quality of the product or the safety of the operations may be questionable.

7.1.5.2 Probability of Gross Errors or Faults

The other indicators ζ and ξ are useful to detect gross errors (outliers) in the data. For example, Johnston and Kramer (1995) show that a useful error fault model is a pdf with two normal distribution components: the first representing normally distributed sensor noise, having a small standard deviation; and the second representing gross errors, having a large standard deviation. In that case, points i can be flagged as gross errors if ζ_{ij} or ξ_{ij} is less than some threshold, which depends on the decisionmaker's utility or loss function accounting for the benefits and costs of detection and misdetection (i.e., of making Type I and Type II errors).

7.1.5.3 Full Posterior Distributions

Analysis of the resulting posterior distribution of the parameters $p(\theta|D,H)$ and the predictive distribution $p(S|D,\theta,H)$ are also important. If the dimensionality of the problem is low enough, direct plotting of the distribution indicates the uncertainty associated with the quantities. The spread of the distributions can be used for determining the uncertainty in the parameter estimates and the predictions, respectively. When the dimensionality is greater than three, it is necessary to either project the distributions into suuspace; analyze their lower dimensional marginal distributions; or analyze the summary (sufficient) statistics of the distributions, which is readily done for canonical distributions.

7.1.5.4 Probability of the Data under the Given Hypothesis

The probability of the data $p(D|H)$ is another important metric that should be analyzed. The higher the value of $p(D|H)$ or its natural logarithm $\log(p(D|H))$, then the more likely that the hypothesis H is correct. As discussed in Section 6.5, hypothesis selection is based on this metric. Moreover, general statements about the form of $p(D|H)$ for a family of models gives us some insight into what hypotheses to attempt next. Thus, search heuristics for exploring the hypothesis space may be developed.

7.1.5.5 Validation of Assumed Distributions

Another indication of the model validity is whether the state variables and the estimated errors actually follow the assumed pdfs used in model synthesis. For example, it is a common error in traditional linear regression to fail to verify that the model residuals are indeed normally distributed, which is the guiding assumption implicit in linear regression. Analogously, plots and summary statistics of the model residuals should be analyzed in this framework to verify that the pdfs are indeed valid.

7.2 Case Study: Industrial Fermentation with Corrupted Data

This case study applies EM to solve the state estimation problem when the error model is a two-component mixture of normal (Gaussian) distributions. Both components in such a mixture are zero-mean, with the first representing the probability density function (pdf) of the process noise, and the other, having a much larger variance, representing the pdf of the large errors. The approach applies the Expectation-Maximization method to the predictive distribution for state estimation, $p(S|E,\theta,H)=p(\tilde{X},\tilde{Y}\,|X,Y,\theta,H)$. In doing so, it estimates the true state (smooths the data) and the posterior probabilities of each observation being an outlier given the true state, i.e. the expected value of the gross-error component indicator $E[\zeta|X,Y]$. When applied to actual industrial fermentation data, the method performs well and is computationally efficient. These encouraging results led us to pursue the implementation of the full method as described in Section 6.4.

7.2.1 Problem Description

This case study applied probabilistic state estimation to actual data from an industrial fermentation process that produces a pharmaceutical. The data were obtained courtesy of Prof. Charles Cooney and Dr. Jack Prior, a member of Prof. Cooney's research group. The database contained seventeen separate fed-batch runs. Each run had nineteen variables. This preliminary analysis of the methodology focused on modeling and rectifying only the evolution of the temperature over time.

Phrased generally, this case study solved the following problem:

State Estimation Problem with Corrupted Data:

> Given process data $U=\{X,Y\}$, corrupted by gross errors and/or missing values, estimate the true process state $W=\{\tilde{X},\tilde{Y}\}$.

Our hypothesis H consisted of the following assumptions based upon past experience with the process (i.e., prior knowledge):

Evidential dependence model: We assume that the process follows an autoregressive moving-average (ARMA(p,q)) process, which generates true states $W=\{\tilde{X},\tilde{Y}\}$. This means W can be represented as a time series, i.e., $\{w_t\}=\{x_t, y_t\}$, t=0, 1, ..., N, and $\psi(B)w_t = \theta(B)a_t$. The lag operator is defined as $B^k w_t \equiv w_{t-k}$; and $\psi(B) \equiv 1 - \psi_1 B - \ldots - \psi_p B^p$; $\theta(B) \equiv 1 - \theta_1 B - \ldots - \theta_q B^q$; where ψ_i and θ_j are parameters. The process is assumed stationary and invertible, i.e., the roots of $\psi(B)$ and $\theta(B)$ all lie outside the unit circle (Wei, 1990; Choi, 1992). It has expectation $E[w_t|H]=0$, since $\{a_t\}$ is a sequence of independent and identically distributed $N(0,\sigma_a^2)$ random variables.

Adopting this ARMA model for the process is equivalent to stating that our prior knowledge dictates a linear model relating each observation w_t to some unknown number K of prior observations, which will be determined from the data. Thus, we have chosen a linear nonparametric model with uncertainty modeled as a normal distribution, manifested by the sequence $\{a_t\}$.

Therefore, the evidential dependence model, based on the autoregressive (AR(K)) representation of the process is expressed in terms of $W_{t:K} \equiv \{w_t, w_{t-1}, \ldots, w_{t-K}\}$: $p(W_{t:K}|\theta,\psi,H)=\phi(W_{t:K};0,\Sigma_f(\theta,\psi,\sigma_a^2))$, $\forall\ t>K$, where the data have been windowed according to lag K and the forecast covariance matrix $\Sigma_f(.)$ is dependent on the model parameters and the noise covariance. Note that the predictions of w_t given values $w_{t-1}, w_{t-2}, \ldots, w_{t-K}$ will be $E[w_t|w_{t-1}, w_{t-2}, \ldots, w_{t-K}, H] = f(w_{t-1}, w_{t-2}, \ldots, w_{t-K};\theta,\psi)$, the (linear) AR($K$) model.

Error model: Measurement errors e_t only allows us to observe $u_t=\{x_t,y_t\}$: $u_t \equiv w_t + e_t$. The errors e_t are assumed to be independent random variables identically distributed according to the two-component mixture model of Gaussians:

$$p(e_t|H) = (1-\varepsilon)\phi(e_t;0,\sigma_a^2) + \varepsilon\phi(e_t;0,(b\sigma_a)^2) \qquad (7.1)$$

where ε is the known prior probability of an additive outlier (gross error) occurring, and b is the assumed ratio of the standard deviation of the outlier distribution to the sensor noise distribution and satisfies $b>>1$.

Again, the problem was to determine the best estimates of w_t, which are denoted as rectified values $w_r(t)$, given corrupted observations u_t. In doing so, observations were detected and flagged as outliers by determining if the estimated error $e_r(t)\equiv u_t - w_r(t)$ was most likely to have come from the broad Gaussian component of (1.3). Given process measurement u_t and parameters $\{\theta,\psi\}$, the rectified value $w_r(t)$, which is the estimate of the true value w_t, was found by maximizing the posterior pdf $p(w_t|u_t,\theta,\psi,H)=p(u_t|w_t,\theta,\psi,H)\ p(w_t|\theta,\psi,H)/p(u_t|\theta,\psi,H)$ with respect to w_t.

Expectation-Maximization (EM) was used to perform the maximization because, under the assumed distributions for $p(w_t|\theta,\psi,H)$ and $p(e_t|H)$, $p(w_t|u_t,\theta,\psi,H)$ is a finite mixture of Gaussians that is maximized at the same w_t as its joint distribution $p(w_t,\zeta_t|u_t)$ defined as

$$\begin{aligned} p(w_t,\zeta_t|u_t,\theta,\psi,H) &= \frac{p(w_t|\theta,\psi,H)\,p(u_t,\zeta_t|w_t,\theta,\psi,H)}{p(u_t|\theta,\psi,H)} \\ &= \left[(1-\varepsilon)p_1\phi(w_t;\mu_1,\Sigma_1)\right]^{(1-\zeta_t)}\left[\varepsilon\, p_2\phi(w_t;\mu_2,\Sigma_2)\right]^{\zeta_t} \end{aligned} \tag{7.2}$$

where ζ_t is the posterior probability of u_t being an outlier; and p_1, p_2 and the parameters of the Gaussians are given by the standard formulas for multiplication (and normalization) of two Gaussian distributions centered at the observation u_t and the forecast $w_f(t)$, respectively (see Appendix B). These formulas for the multivariate case with noise covariance Σ and forecast error covariance Σ_f, assuming b and ε are the same for each variable, are as follows:

$$p_1^{-1} = (1-\varepsilon)\int \phi(u_t;w_f(t),\Sigma_f)\phi(u_t;0,\Sigma)\,du_t\ ;\quad p_2^{-1} = \varepsilon\int \phi(u_t;w_f(t),\Sigma_f)\phi(u_t;0,b^2\Sigma)\,du_t\ ;$$

$$\mu_1 = \Sigma_f(\Sigma_f+\Sigma)^{-1}y_t + \Sigma(\Sigma_f+\Sigma)^{-1}w_f(t)\ ;\quad \mu_2 = \Sigma_f(\Sigma_f+b^2\Sigma)^{-1}y_t + b^2\Sigma(\Sigma_f+b^2\Sigma)^{-1}w_f(t)\ ;$$

$$\Sigma_1 = \Sigma_f(\Sigma_f+b^2\Sigma)^{-1}b^2\Sigma\ ;\ \text{and}\ \Sigma_2 = \Sigma_f(\Sigma_f+\Sigma)^{-1}\Sigma\,.$$

Introducing ζ_t into the problem has two purposes: (*1*) as discussed in Chapter 6, EM provides an efficient means of maximizing such mixtures expressed in this form (Dempster *et al.*, 1977) and (*2*) during the Expectation-Step of the algorithm, EM computes the "missing" information ζ_t for each observation u_t, and thus, provides a quantitative measure of the plausibility that u_t is an outlier. So, in solving this problem by EM, both detection of the outliers and compensation for their effects can be accomplished simultaneously.

7.2.2 Application to Dynamic Processes

The EM formulas for maximizing (7.2) with respect to w_t are as follows:

E-Step:
$$\zeta_t \equiv \mathrm{E}[\zeta_t | y_t, H] = \frac{\varepsilon\, p_2 \phi(y_t; \mu_2, \Sigma_2)}{(1-\varepsilon) p_1 \phi(y_t; \mu_1, \Sigma_1) + \varepsilon\, p_2 \phi(y_t; \mu_2, \Sigma_2)} \tag{7.3}$$

M-Step:
$$w_r(t) = \mathrm{Q}(\zeta)\mathrm{S}(\zeta), \tag{7.4}$$

where $\mathrm{Q}(\zeta)^{-1} \equiv (1-\zeta_t)\Sigma_1^{-1} + \zeta_t \Sigma_2^{-1}$; and $\mathrm{S}(\zeta) \equiv (1-\zeta_t)\Sigma_1^{-1}\mu_1 + \zeta_t \Sigma_2^{-1}\mu_2$.

A benefit of using EM is that the rectified data $w_r(t)$ are simultaneously optimized with the estimates of the outlier probabilities $\zeta_t(u_t)$. As a simple first approach to outlier detection, the computation needed to converge successive M-Steps of the iterative EM method is avoided. This simple approach only uses the E-Step formula to compute $\zeta_t(e_t)$. To do so, the measurement error e_t is estimated, $\zeta_t(e_t)$ is calculated by (7.3), and an observation u_t is flagged as an outlier if $\zeta_t(e_t)$ is greater than 0.5. Outlying observations can then be replaced with estimates of their true values. This approach is fast and simple, but has the obvious problem that the true values w_t and hence the true errors e_t are not known. However, if the process truly follows the ARMA(p,q) model derived from uncorrupted data, then the one-step-ahead forecast $w_t(t)$ serves as a good estimate of w_t. Thus, a good estimate of e_t is the forecast error w_t-$w_t(t)$. So, a simple outlier detection scheme is described by the following procedure.

Table 7.1. Simple Outlier Detection Approach.

Given: observations u_t, and the parameters θ and the pdfs.

For each observation u_t:

(*1*) Compute the forecast $w_f(t)$.

(*2*) Estimate error e_t as $e_r(t)=w_t-w_f(t)$.

(*3*) Check which component of the error pdf has the greatest value at $e_r(t)$.

(*4*) If the outlier-component pdf has greater value than the noise-component pdf. then mark the data point as an outlier and remove the outlier by replacing w_t with $w_f(t)$.

(*5*) If more observations, go to *Step 1*.

This simplistic approach is appealing but does not yield the jointly most probable values for all process measurements. The robust dynamic estimation approach proposed here succeeds in finding at least locally optimal values.

Table 7.2. Robust Dynamic Estimation Approach.

Given: Observations u_t, the parameters θ, and the pdfs.

Over the full data set:

(*1*) Compute the forecasts $w_f(t)$.

(*2*) Perform an E-Step to estimate unknown probabilities ζ_t.

(*3*) Perform an M-Step to estimate the rectified value $w_r(t)$ that maximizes the posterior (1.5).

(*4*) Converged? No: go to *Step 1*. Yes: continue to *Step 5*.

(*5*) Return the rectified (smoothed) data $w_r(t)$ and the probabilities of outliers ζ_t.

(*6*) Detect outliers: an outlier is detected in u_t, if ζ_t exceeds 1/2.

Of course, both of the above approaches assume that an accurate ARMA(p,q) model can be derived from prior data and that the model still holds over the subsequent time series. In a time series analysis framework, this amounts to having a suitably long starting block of uncorrupted observations. In a process data rectification framework, it is assumed that process operation is sufficiently reproducible that separate production runs have the same nominal correlation structure. Therefore, a prototypical run may be selected to serve as the basis for time series analysis and ARMA model synthesis. Subsequent runs are rectified using the $p(w_i|H)$ dictated by this underlying ARMA(p,q) model. Of course, such a well-behaved process is a big assumption, and it would be preferable to dynamically estimate model parameters during each run as the data evolve over time. This is done in the robust Kalman filtering approaches, an example of which is given by Schick and Mitter (1994).

7.2.3 Time Series Analysis

Time series analysis of the prototype data was performed to determine the evidential dependence model. This procedure included mean-centering the series, plotting the data and its computed sample autocorrelation function (ACF) and partial autocorrelation function (PACF) (Wei, 1990), and identifying the model order. The final step included differencing the data, computing and plotting the ACF and PACF of the differenced series, estimating the parameters, testing for a deterministic trend parameter, performing a time series analysis on the residuals a_t to check for correlation, computing one-step-ahead forecasts and the 95% forecast limits, and plotting the forecast errors. The series chosen as the prototypical run from which the ARMA model was generated had N=402 temperature measurements. The process temperature in each run was sampled at 20-minute intervals.

The procedure was done using Mathcad Plus and its Signal Processing Handbook. Appendix E is a hardcopy of the Mathcad Plus worksheet. Equations appearing in the worksheet are not merely typeset formulas but rather are the actual computer language

used by Mathcad Plus to perform the calculations. Appendix E includes ample comments on the time series analysis, so only the results of the analysis will be mentioned here. The resulting ARIMA model turned out to be an IMA(1,1) with single parameter θ=0.8441. Checks of the residuals, the forecasts, the 95% forecast limits and forecast errors all proved that the process was adequately represented as IMA(1,1).

The simple approach to outlier detection and the more elaborate dynamic data rectification approach described in Section 6.3.3 were tested using a single series from another process run. The new series contained N_2=426 observations with six additive outliers appearing in observations t={419, 420, 421, 423, 424, 425}. Note that observation t=422 is not an outlier.

The hyperparameters in the problem were set as follows. The prior probability of an outlier ε was chosen to be 0.01. The ratio of the gross error standard deviation to the noise standard deviation b was set to 20. These values were not based on actual prior knowledge of the process but were settled upon as suitable after experimenting with different combinations.

7.2.4 Results

As a first approach to outlier detection, the posterior probabilities z_t were used to flag outlying observations. Figure 7.2 (a bar chart of ζ_t at t=415, 416,...,426) and Figure 7.3 show that even this simple approach was capable of detecting the six outliers in the new data. However, more probable values for the non-outlying observations cannot be estimated using this approach.

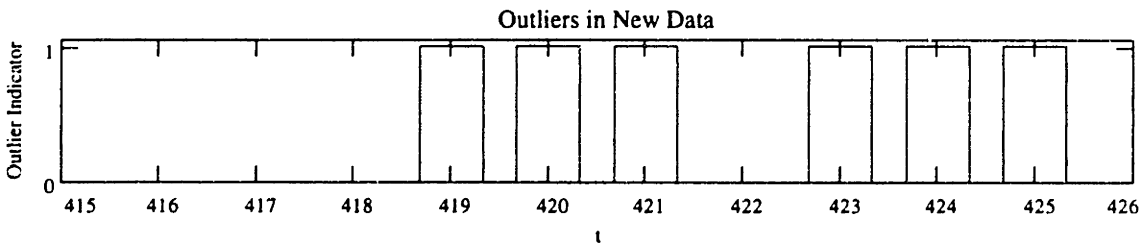


Figure 7.2: Simple approach to outlier detection

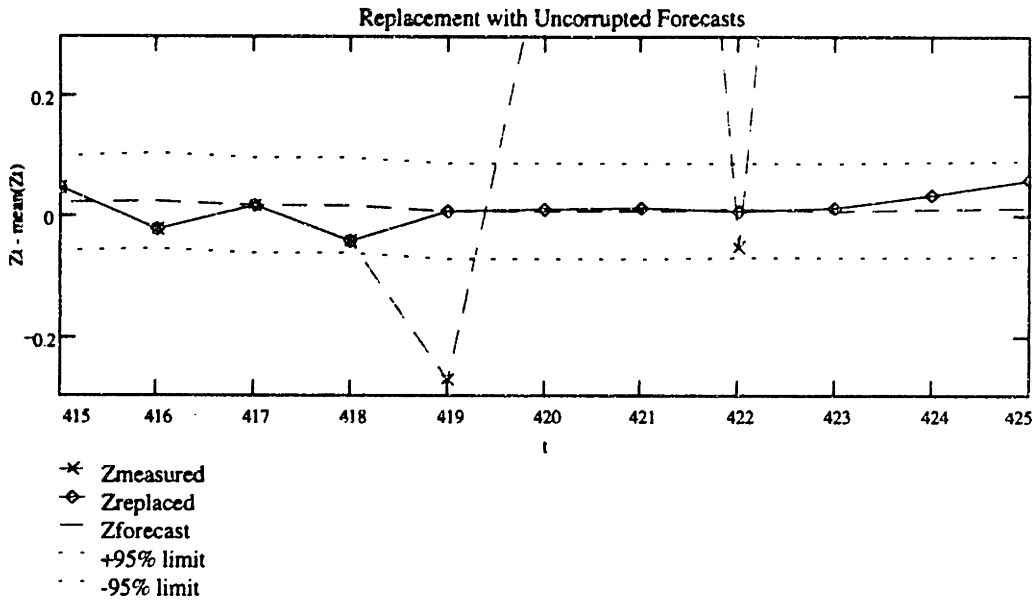


Figure 7.3: Removal of outliers using simple approach

On the other hand, the dynamic data rectification approach simultaneously estimated rectified values $w_r(t)$ and outlier probabilities w_t. Figures 7.4, 7.5 and 7.6 show the results of applying the technique to the new data. Not only are the outliers detected and removed, all process observations are re-estimated with more probable values.

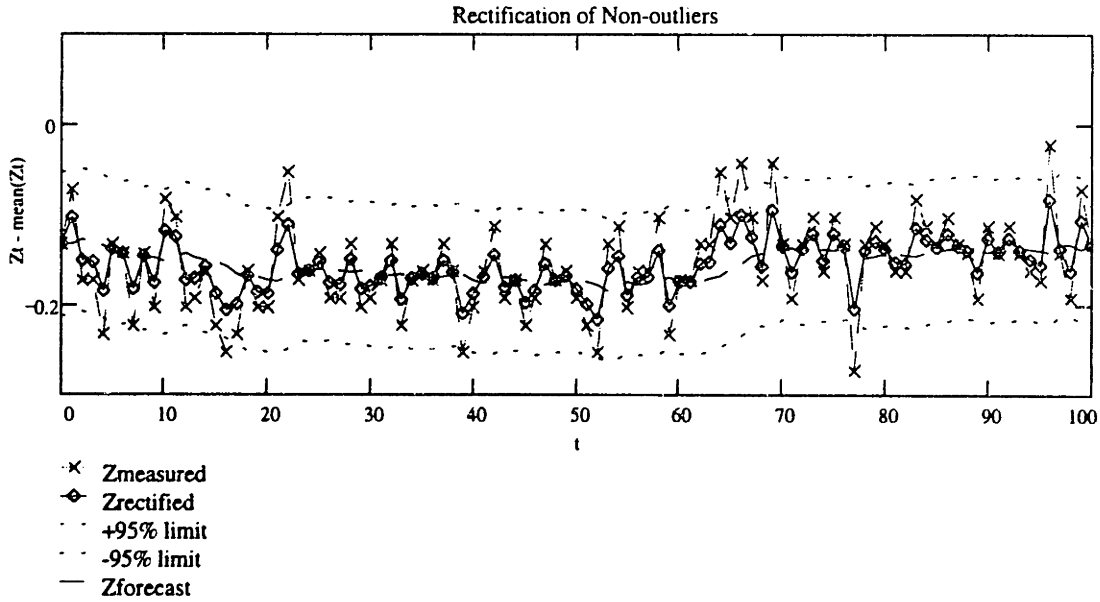


Figure 7.4: Rectification of non-outliers.

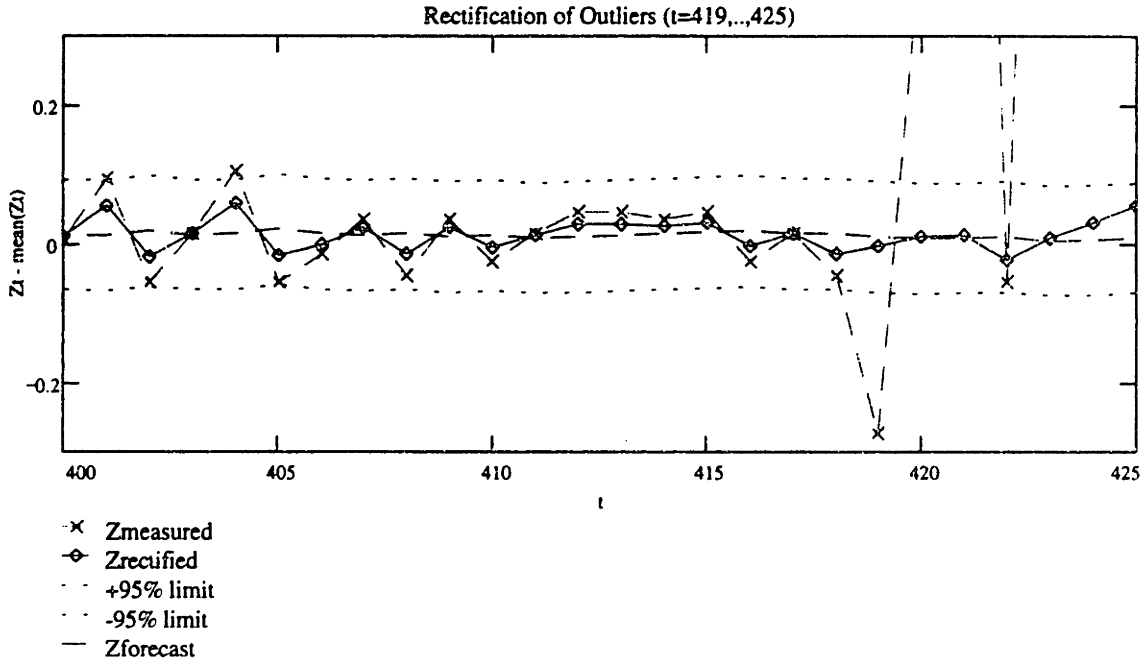


Figure 7.5: Rectification of outliers

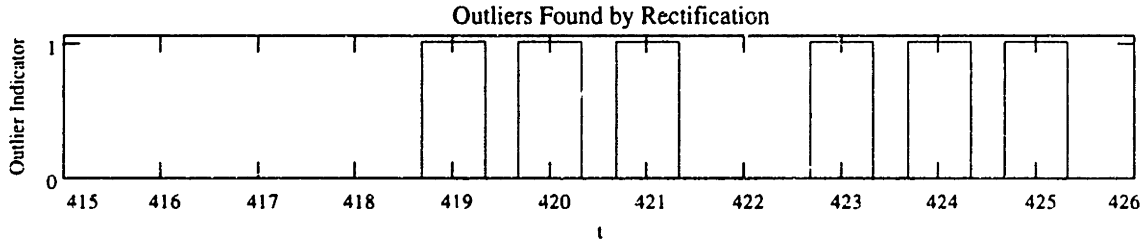


Figure 7.6: Outlier detection using rectification

Both the simple outlier detection scheme and the full dynamic estimation approach were sensitive to the values for the hyperparameters of the pdf, ε and b, which are determined from prior information. However, experiments showed that as long as b was sufficiently large — on the order of 10 — the prior probability of a gross error ε was the determining factor in how sensitive each approach was to detecting gross errors (i.e., the power of the method) and how heavy-handed it was to smoothing the data. Also, experiments have shown that the forecast values $w_f(t)$ are good initial guesses for $w_r(t)$, and rapid convergence ensues (e.g., less than five iterations in the univariate cases tested). Additionally, it would be more realistic in the multivariate case to have a separate ε and b

for each variable, as is done in Johnston and Kramer (1995). However, this dramatically increases the computation required in the M-Step of the EM algorithm and necessitates approximations to $p(e|H)$ to maintain tractability.

7.2.5 Summary

This case study demonstrated the solution of the state estimation problem when the data are corrupted with gross errors. The value of using the EM solution method of Chapter 6 is apparent: the method was capable of simultaneously detecting grossly corrupted points and compensating for their effects in estimating future states. This approach has the advantage that it is simple to implement, can be computed efficiently, and provides acceptable performance in the case of the univariate industrial data to which it was applied. The study served as a good starting point for the following case studies involving multivariate dynamic data.

7.3 Case Study: Fed-Batch Penicillin Fermentation

This case study demonstrates robust state estimation for dynamic systems. The approach is that of Section 6.4, which applies the probabilistic paradigm to corrupted data. The case study extends the investigation presented in the study of Section 7.2. It does so by simultaneously estimating model parameters while compensating for noise, gross errors and missing values in the multivariate measurements. It also represents an extension over the function-oriented hybrid modeling approach applied to penicillin fermentation in the case study of Section 3.7.2. Here, we show that the probabilistic paradigm yields a model with enhanced performance over that of the hybrid neural network.

7.3.1 Process Description

In this study, we revisit the fed-batch penicillin fermentation of Section 3.7.2, which applied the function-oriented paradigm to the process. As we did there, we assume that our process knowledge includes a candidate parametric model based on our prior experience with similar microorganism strains. The knowledge base as applied here also includes information concerning the noise and potential gross errors (or outliers) in the measurements.

The assumed prior knowledge for this problem is the same as that presented in Section 3.7.2: a prior model $v_t=f_{prior}(\tilde{y}_t)$ that relates the unmeasured specific rates v_t to the state variables $\tilde{y}_t$; and the mass balances $\tilde{y}_t=f_{out}(\tilde{x}_t, \tilde{y}_{t-1}, v_t)$ that relate the state variables at time t to the exogenous variables $\tilde{x}_t$ at time t, the state variables at time t-1, and the specific rates at time t. Along with this information, errors are assumed in the measurements x_t and y_t. The error model is the two-component mixture of normals as used in the previous section and shown above in Eqn. 7.1.

7.3.2 Model Synthesis

Given the prior model and the mass balance equations, we can attack this problem using many approaches. We choose the following one:

> Apply the prior model, limiting its applicability to regions of sparse or absent data; apply the mass balance constraints; and augment this knowledge with a nonparametric model to induce relations from data.

This approach fully exploits the knowledge we possess while allowing the data to maximally influence the aspects of the model for which we have least prior knowledge. In other words, both the prior knowledge and the data are allowed to have the greatest impact on the aspects of the model for which they are most needed.

Some alternatives to this approach include the following:

- Use the given prior model and re-estimate its parameters. The given parameter values serve as expected values from which we derive prior pdfs for new parameters. This approach has the advantage of being a simple means of exploiting the prior model. However, it has the disadvantage that if the model form is incorrect in the first place, we will retain this detrimental bias.
- Use the prior model (with its given parameter values) as an expected value for the evidential dependence model and add uncertainty through a variance about this model. This approach compensates for any potential detrimental bias. However, unless the uncertainty we inject is asymmetric about the expected value, using the expectation as a point estimate is equivalent to simply predicting with the prior model – which we have already said is inadequate. Recognizing this, we still lack prior knowledge upon which to base the asymmetry. Besides, even if we had a basis for injecting asymmetry, computationally intensive sampling methods would be required to estimate the state.

- Discard the prior model, use a fully nonparametric model, yet retain the mass balance constraints. This semiparametric modeling approach eliminates from consideration the most questionable aspect of the model and injects a fully data-derived (nonparametric) model in its stead. Unfortunately, this approach wastes potentially valuable information contained in the prior model. Also, it increases the burden placed on the sparse data.

We chose our approach because it more fully exploits the available knowledge and the data than the other approaches while maintaining mathematical tractability. The approach implements a semiparametric hybrid model with origins similar to the hybrid derived in Section 3.7.2. However, as will become more evident below, the probabilistic paradigm presents a different take on semiparametric modeling by allowing probability theory to dictate the form of the nonparametric model, the objective function, and the impact of our prior knowledge. Moreover, the probabilistic paradigm provides a consistent, theoretical basis for incorporating information about missing values and gross errors in the data.

7.3.2.1 Evidential Dependence Model

Given our prior knowledge *H*, as described above, the evidential dependence model can be constructed by following the path of information from what we know to what we would like to know. In doing so, we refer back to Chapter 5 to select appropriate distributions for the pdfs of state estimation shown in Figure 4.1. These are combined into Eqns. 6.22, 623, 6.26 and 6.27 of Section 6.4 to estimate the true state and the parameters in spite of the corrupted data.

Examining the state estimation problem depicted in Figure 4.1 of Section 4.4.8, we recognize that the evidential dependence model will be $p(\tilde{y}_t \mid \tilde{x}_t, \tilde{Y}_{\{<t\}}, \theta, H)$, where $\tilde{Y}_{\{<t\}}$ represents all $\tilde{y}$ prior to $\tilde{y}_t$ that are correlated with $\tilde{y}_t$. In this model, parameters θ have not yet been explicitly identified but have been introduced in anticipation of their

existence and capability to summarize the data X, Y. To estimate these parameter we need the posterior $p(\theta|X,Y,H)$. Referring to Section 4.4, we are able to decompose this into more fundamental pdfs, noting the appropriate assumptions (which become part of H).

The posterior is proportional to the product of the likelihood and the prior (from Bayes' Theorem); and the likelihood is expressible as a series product (from the Product Rule) with terms $p(x_t,y_t|X_{\{<t\}},Y_{\{<t\}},\theta,H)$. Each term of the series is a marginal distribution of a joint pdf that includes the true values $\tilde{x}_t$ and $\tilde{y}_t$. The true values are missing information in the same sense as the indicator variables are in a mixture estimation problem. So, for now we will manipulate the joint distribution and deal with them later in the solution using an EM approach. Assuming uncorrelated errors (i.e., x_t is conditionally independent of x_j when $\tilde{x}_t$ is known – and analogously for y), this joint pdf can be factored:

$$p(x_t,y_t,\tilde{x}_t,\tilde{y}_t|\tilde{X}_{\{<t\}},\tilde{Y}_{\{<t\}},\theta,H) = p(x_t|\tilde{x}_t,H)p(y_t|\tilde{y}_t,H)p(\tilde{y}_t|\tilde{x}_t,\tilde{X}_{\{<t\}},\tilde{Y}_{\{<t\}},\theta,H)p(\tilde{x}_t|\tilde{X}_{\{<t\}},H).$$

The existence of the unmeasured specific rates v is made explicit in order to take advantage of our prior knowledge. The specific rates are missing information like the true values and will also be dealt with below. For know we will manipulate the joint pdf from which the marginal pdf $p(\tilde{y}_t|\tilde{x}_t,\tilde{X}_{\{<t\}},\tilde{Y}_{\{<t\}},\theta,H)$ is derived. The mass balance dictates factoring the joint pdf according to the Product Rule:

$$p(\tilde{y}_t,v_t|\tilde{x}_t,\tilde{X}_{\{<t\}},\tilde{Y}_{\{<t\}},\theta,H) = p(\tilde{y}_t|\tilde{x}_t,v_t,\tilde{Y}_{\{<t\}},\theta,H)p(v_t|\tilde{x}_t,\tilde{X}_{\{<t\}},\tilde{Y}_{\{<t\}},\theta,H). \qquad (7.7)$$

Therefore, the function to be maximized for parameter estimation becomes the following:

$$G(\theta) = \prod_{t=1}^{N} p(x_t|\tilde{x}_t,H)p(y_t|\tilde{y}_t,H)p(\tilde{y}_t|\tilde{x}_t,v_t,\tilde{Y}_{\{<t\}},H)p(v_t|\tilde{x}_t,\tilde{Y}_{\{<t\}},H)p(\tilde{x}_t|\tilde{X}_{\{<t\}},H).$$

(Noting that the observation models and the pdf for v will be finite mixtures, we will also ultimately introduce appropriate indicator variables: ζ for $p(x_t|\tilde{x}_t,H)$, ξ for $p(y_t|\tilde{y}_t,H)$ and Z for $p(v_t|\tilde{x}_t,\tilde{Y}_{\{<t\}},H)$.)

Now, we must specify the forms of the pdf in Equation 7.7. The first term in the factorization of the joint pdf is the mass balance:

$$p(\tilde{y}_t \mid \tilde{x}_t, v_t, \tilde{Y}_{(<t)}, \theta, H) = \delta(\tilde{y}_t, f_{out}(\tilde{x}_t, v_t, \tilde{Y}_{(<t)})).$$

The second term is shaped by our knowledge of the inadequacy of the data, thus the need for a default model; and the fact that even within the data, the default model can serve as a useful guide function, with additional uncertainty represented as a nonparametric bias term b (Section 5.3.1):

$$p(v_t \mid \tilde{x}_t, \tilde{Y}_{(<t)}, H) = \int p(v_t \mid \tilde{y}_t, \theta, H)\, p(\tilde{y}_t \mid \tilde{x}_t, \tilde{Y}_{(<t)}, H)\, d\tilde{y}_t \text{ ; and}$$

$$p(v_t \mid \tilde{y}_t, \theta, H) = P(M_{def} \mid \tilde{y}_t, H)\, p(v_t \mid \tilde{y}_t, M_{def}, H) + P(M_{Np} \mid \tilde{y}_t, H) \int p(v_t \mid \tilde{y}_t, b_t, M_{Np}, H)\, p(b_t \mid \tilde{y}_t, \tilde{Y}_{(<t)}, \theta, M_{Np}, H)\, db_t$$

The nonparametric model for the bias b_t is a finite mixture distribution as in Section 5.3.1:

$$\begin{aligned} &\int p(v_t \mid \tilde{y}_t, b_t, M_{Np}, H)\, p(b_t \mid \tilde{y}_t, \tilde{Y}_{(<t)}, \theta, M_{Np}, H)\, db_t \\ &\quad = \sum_{j=1}^{K} p(v_t \mid \tilde{y}_t, \theta_j, M_j, M_{Np}, H)\, P(M_j \mid \tilde{y}_t, M_{Np}, H) \qquad (7.8) \\ &\quad = \sum_{j=1}^{K} \delta(v_t, f_{def}(\tilde{y}_t) + \theta_j)\, P(M_j \mid \tilde{y}_t, M_{Np}, H) \end{aligned}$$

7.3.2.2 Solution Method

The parameter estimation problem in this case is a classic missing value or incomplete data problem, where $\tilde{x}_t$, $\tilde{y}_t$, and v_t are missing. The usual EM strategy for such a problem is to maximize $E[\log(G(\theta^{(k)})) \mid \theta^{(k-1)}]$ on the kth iteration given θ from the k-1st iteration, where the expectation is taken over $p(\tilde{x}_t, \tilde{y}_t, v_t \mid \theta=\theta^{(k-1)})$. If $\log(G(\theta))$ is linear in the missing values, then this is equivalent to maximizing $\log(G(\theta(k)))$ evaluated at the expectations of the missing values. When, it is nonlinear, though, evaluation at the expectations can still be used to generate an approximate solution.

That is what we do here – evaluate $\log(G(\theta^{(k)}))$ at the expectations of the missing values – with judgement of appropriateness hinging on the accuracy of the model given the parameters we obtain. Alternatively, this can be viewed as using a sequence of point estimate solutions for the missing values under quadratic loss: i.e., first solving the estimation of $\tilde{x}_t$ by robust nonparametric density estimation; then estimating v_t given $\tilde{x}_t$ and $\tilde{Y}_{(<t)}$; and finally, estimating $\tilde{y}_t$ given $\tilde{x}_t$, v_t and $\tilde{Y}_{(<t)}$.

It becomes clear that to use our knowledge of the default model, we will either need to integrate out the current state $\tilde{y}_t$ in Equation 7.8 or supply an intermediate estimate for $\tilde{y}_t$ so that we can compute v_t. Thus, our solution strategy is to use robust nonparametric density estimation to determine the best estimate of $\tilde{x}_t$. Then, likewise, use density estimation to get a preliminary $\tilde{y}_t$. Next, we estimate $v(\tilde{y}_t)$ by simple evaluation of $E[v_t|\tilde{y}_t]=f_{def}(\tilde{y}_t) + P(M_{Np}|\tilde{y}_t,H)\Sigma[\theta_j P(M_j|\tilde{y}_t,M_{Np},H)]$. Then, parameters θ are found by maximizing Eqn. 6.26 of Section 6.4.2. Finally, we refine the estimate of $\tilde{y}_t$, by applying Eqn. 6.27. Although an appeal to the beneficial convergence properties of the EM algorithm can help rationalize this approach, the overriding factor in its choice are its pragmatism and good results, shown below.

7.3.3 Robust Nonparametric Density Estimation

As shown in Section 6.4, this state estimation with corrupted data can be solved by alternately estimating $\tilde{X}$ (rectifying the independent variables) during nonparametric density estimation, then estimating the parameters θ given the data and $\tilde{X}$. Noting that v_t has a semiparametric dependence on $\tilde{y}_t$, we see that it is necessary to not only perform nonparametric density estimation on X (using Eqns. 6.22 and 6.23), but also on the Y, in order to determine $p(\tilde{y}_t|\tilde{y}_{t-1},H)$. Therefore, robust density estimation was used to

compute $p(\tilde{y}_t, \tilde{y}_{t-1} | H)$, and the conditional pdf $p(\tilde{y}_t | \tilde{y}_{t-1}, H)$ was then calculated from $p(\tilde{y}_t, \tilde{y}_{t-1} | H)$ using the formula in Appendix B.5.

7.3.4 Robust State Estimation

The parameters θ of the nonparametric mixture model for the specific rates were estimated using $\tilde{X}$ from robust density estimation, estimates v_t and $\tilde{y}_{t-1}$, and data Y. The initial $\tilde{y}_1$ was assumed to simply equal the measured value. From that starting value, each estimate of $\tilde{y}_t$ given $\tilde{y}_{t-1}$ was computed as the expectation of $p(\tilde{y}_t | \tilde{y}_{t-1}, H)$. This preliminary estimate of $\tilde{y}_t$ was then used to estimate v_t, which was calculated as the expectation of $p(v_t | \tilde{y}_{t-1}, \theta, H)$ of Eqn. 7.7. Finally, given the estimate of θ at each iteration, Eqn. 6.27 was used to get a new estimate of $\tilde{y}_t$, which is includes all information from the data of the four training runs described in Section 3.72 and the other estimated quantities $\tilde{x}_t$, $\tilde{y}_{t-1}$, and v_t.

In all of these calculations, all $S(z_i)$ (of Eqns. 6.26 and 6.27) equal zero because the local models of the finite mixture in the evidential dependence model are assumed exact. Therefore, $\tilde{y}_t = m(\tilde{x}_t, \tilde{y}_{t-1}, \theta, z_i) = \sum z_{ij} f_{out}(\tilde{x}_t, f_{def}(\tilde{y}_{t-1}), \theta_j)$. Also, the components of the error models were unbiased, so the u(.) terms in Eqns. 6.26 and 6.27 equaled zero.

7.3.5 Results and Analysis

Below, we present the results of this case study. They illustrate the improved performance in prediction over the neural network hybrid of Section 3.7.2 and show the value of the indicator variables in compensating for errors and interpreting model and process behavior.

7.3.5.1 Robust Nonparametric Density Estimation

The results of the nonparametric density estimation of $p(\tilde{y}_t \mid \tilde{y}_{t-1}, H)$ are shown in Figures 7.7 to 7.9. They show the substrate, biomass and product concentrations, respectively, of the first training run. Similar estimates were obtained for the other three training runs. Depicted are the estimates of $\tilde{y}_t$ based solely on information contained in data Y and summarized by $p(\tilde{y}_t, \tilde{y}_{t-1} \mid H)$. During estimation, the automatic deletion of collapsed components was performed (as described in Section 6.7), starting with 25 components and ultimately ending with the five-component mixture used in estimating $\tilde{y}_t$.

The estimates in each figure are the expectations of $p(\tilde{y}_t \mid \tilde{y}_{t-1}, H)$. These estimates are essentially robust nonparametric linear regressions of $\tilde{y}_t$ versus $\tilde{y}_{t-1}$, weighted with a contribution from the measurement y_t. The influence of y_t is dictated by the standard deviations of the noise and the gross error components. Thus, the estimates are equivalent to forecasts of a first-order autoregression that have been updated by the measurement.

The estimates are based on the pooled data of the four training runs, which provided four partitions of N=66 sets of observations each, windowed to lag 1 (i.e., each observation contained the three state variables at time t and at time t-1, for a total of six variables per observation). So, density estimation took place in the six-dimensional windowed state space. Additionally, it was performed assuming that the true standard deviation of the measurements for each variable was known.

Although, these estimates do not include the information about $\tilde{y}_t$ that is contained in the measurements on X and the knowledge of the role v_t plays in the mass balance, the estimated variables follow quite closely those of the true process states. Explicit representation of the error model pdf has resulted in the predictions being smoothed, unlike those of the hybrid neural network (Figures 3.8 and 3.9), which showed much greater variance and also a slight lag.

Figures 7.7 to 7.9 also show the benefit of using the probabilistic paradigm to elucidate the underlying process behavior. The near-vertical dashed lines delimit the different process operating regimes that are implied by the z_{ij} of each component of the mixture. We see that the regimes were automatically allocated densely enough to compensate for the fast-changing high-growth conditions early in the fermentation, while later, during the slow-growth production phase, the regimes are more spacious. Importantly, each z_{ij} can be examined to determined exactly which measurements affected which components most strongly.

This study also investigated the sensitivity of the results to a modeler's uncertainty in setting the hyperparameters of the error model. Figures 7.10 to 7.12 show the results of setting the standard deviation of the measurement noise to a value that is 25% higher than the actual value. Conversely, Figures 7.13 to 7.15 present results when the hyperparameter is set 25% lower than the actual value.

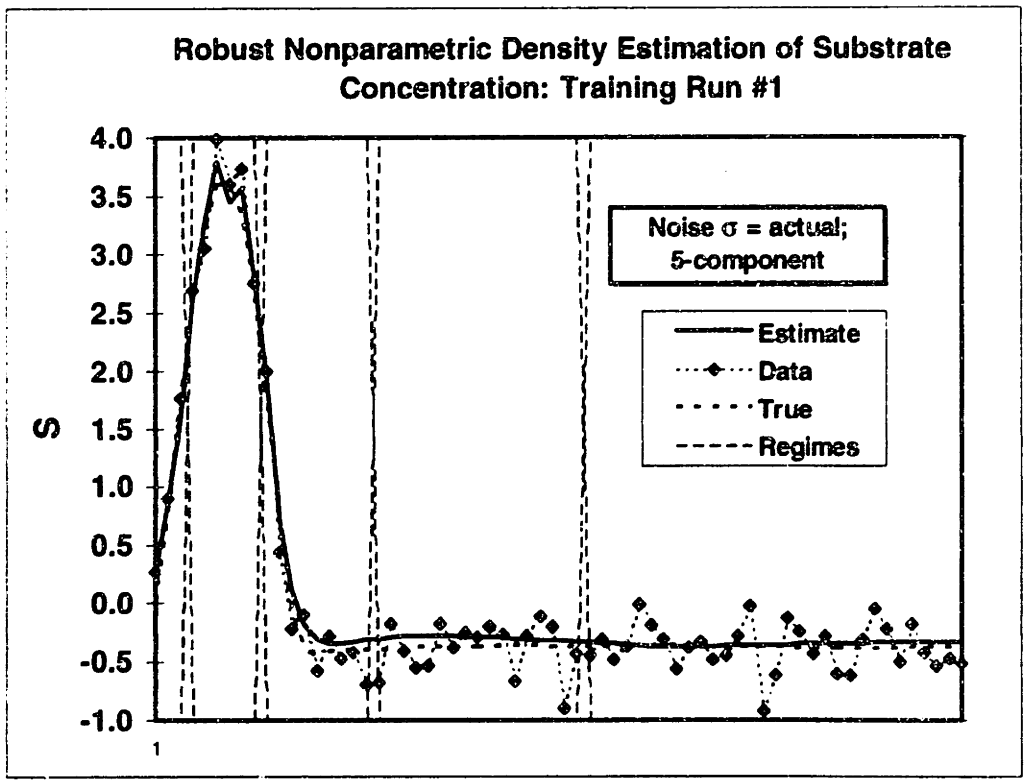


Figure 7.7: Robust nonparametric density estimation: substrate, σ=actual

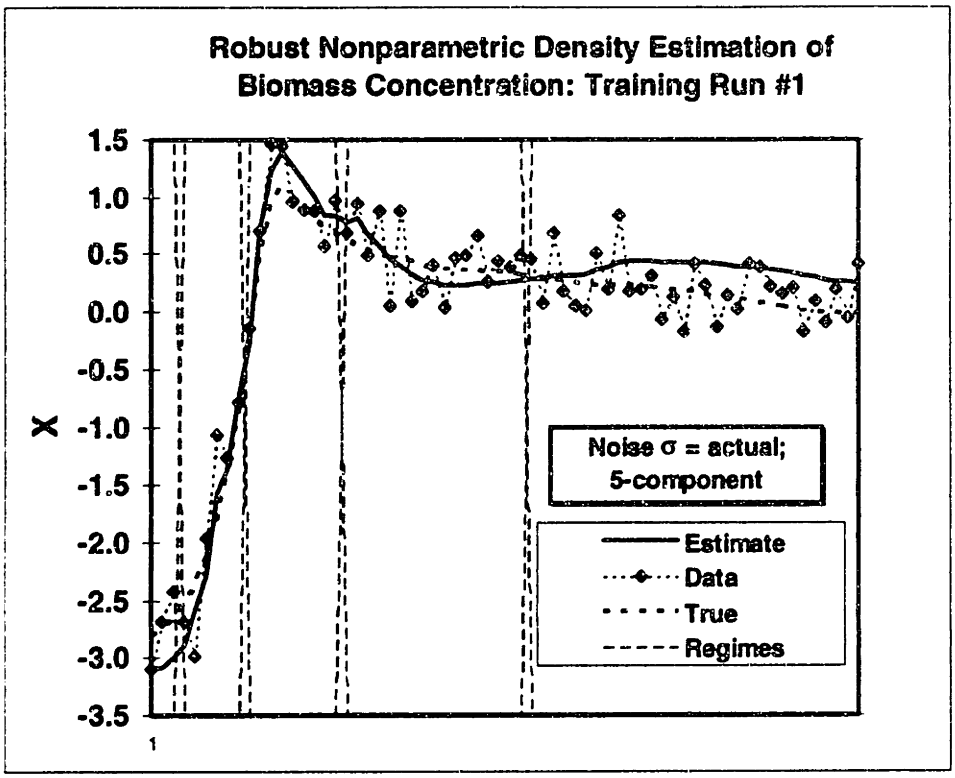


Figure 7.8: Robust nonparametric density estimation: biomass, σ=actual

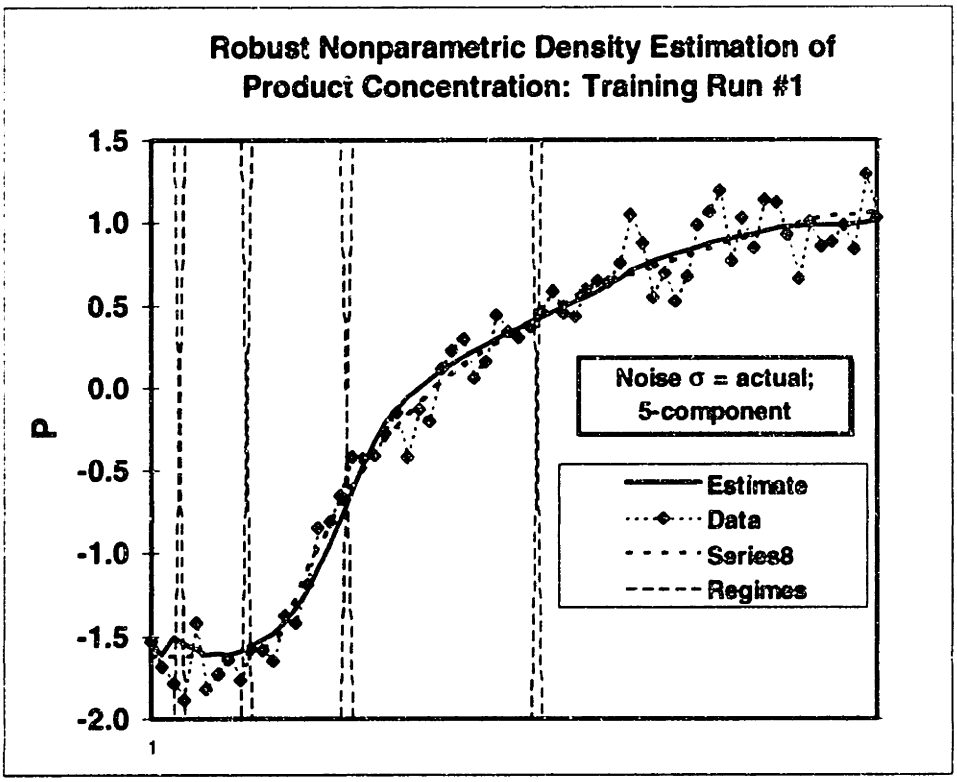


Figure 7.9: Robust nonparametric density estimation: product, σ=actual

The noise standard deviation moderates the weighting of each data point versus the prior knowledge, i.e. the expectation of the distribution $p(\tilde{y}_t \mid \tilde{y}_{t-1}, H)$, which, in this context, is the prior estimate of $\tilde{y}_t$. The larger the standard deviation is, the less weight is given to the data and the more to the prior. Therefore, when the noise standard deviation is set too high, the data are given little weight and the estimated values become essentially the prior values dictated by the expectation of $p(\tilde{y}_t \mid \tilde{y}_{t-1}, H)$. As a result, the values can be easily summarized by fewer components -- because they basically become clustered at the expectations of the components. This is what has happened in Figures 7.10 to 7.12, which required only a 2-component mixture to describe the data. It is apparent that over-smoothing has taken place. However, the automatic allocation of process operating regime boundaries is sensible, allocating fast-growth and slow-growth regimes.

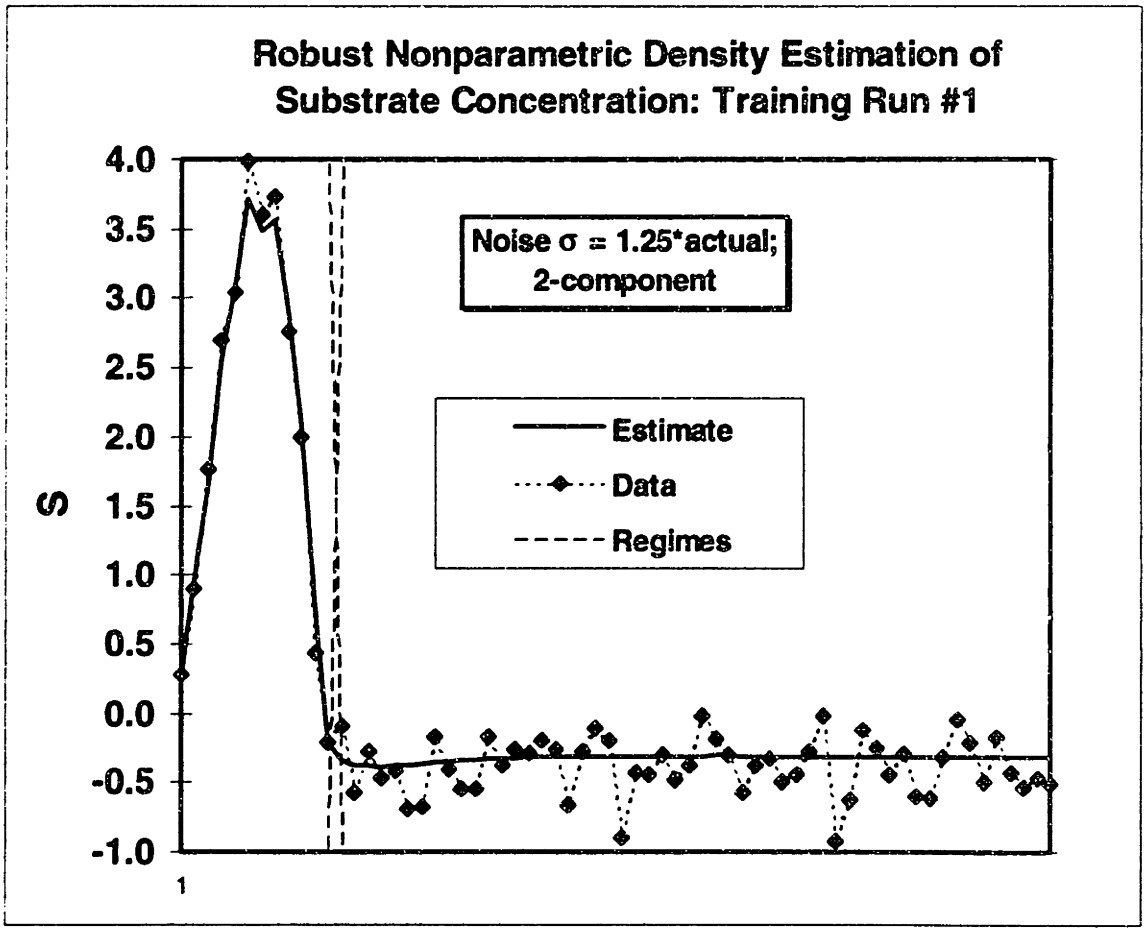


Figure 7.10: Robust nonparametric density estimation: substrate, σ=1.25*actual

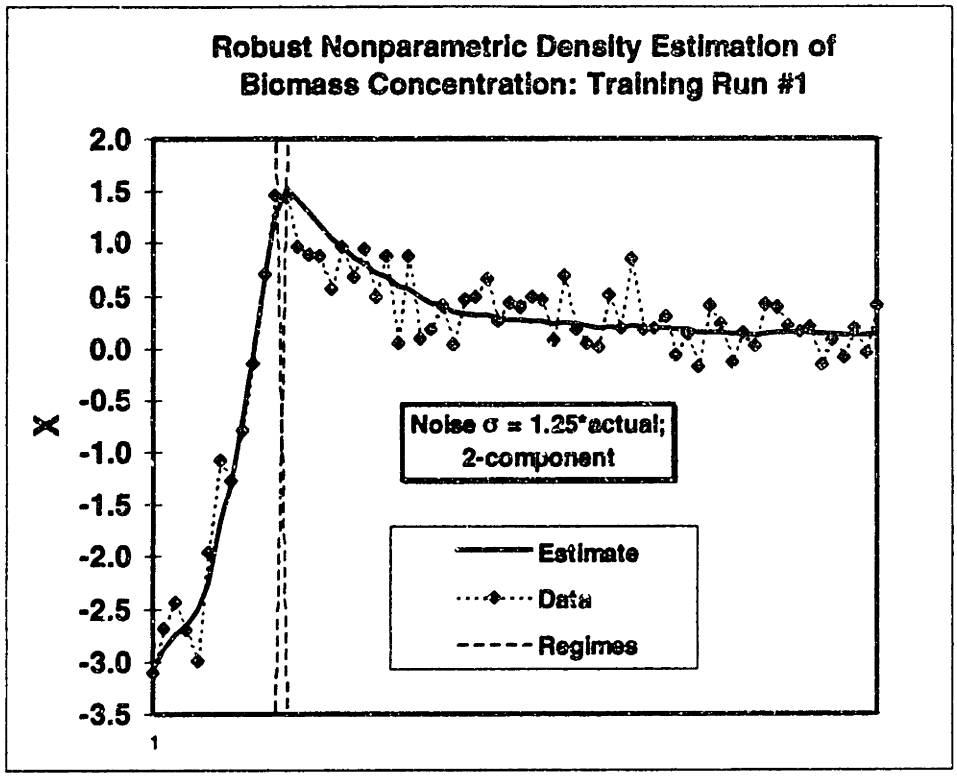


Figure 7.11: Robust nonparametric density estimation: biomass, σ=1.25*actual

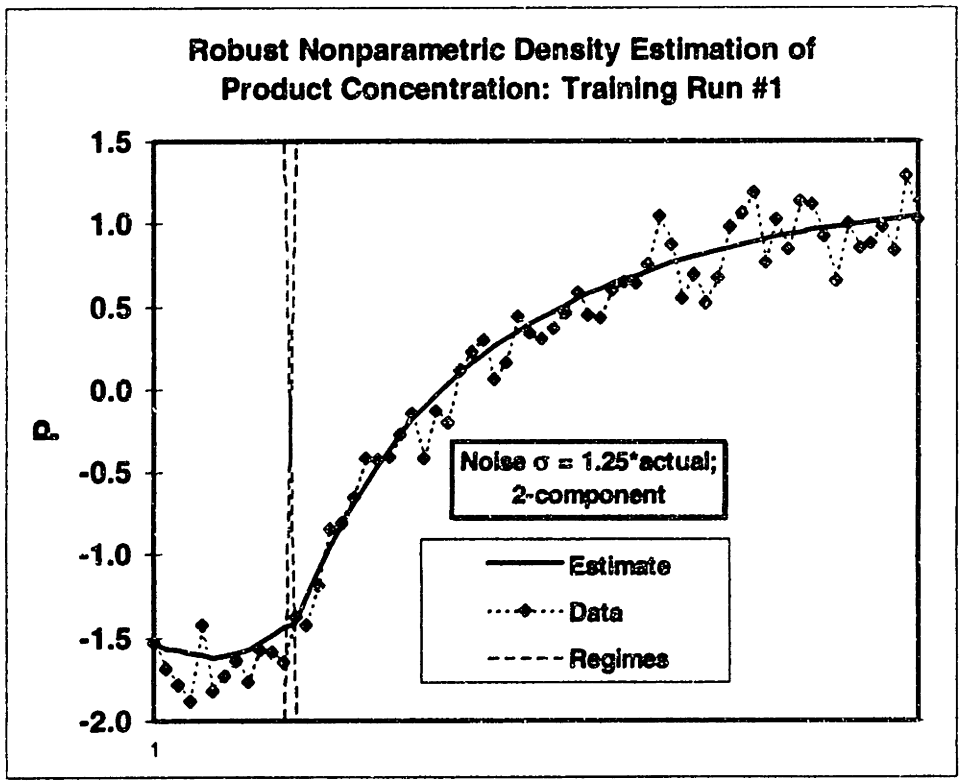


Figure 7.12: Robust nonparametric density estimation: product, σ=1.25*actual

Consistent with the above explanation are the results shown in Figures 7.13 to 7.15, which arose from using a value 25% lower than the actual standard deviation of the noise. The pdf ended up with ten components. Consequently, the resulting estimates show considerable overfit as noise is erroneously included in the estimate along with the meaningful information from the data. However, unlike the predictions of the function-oriented hybrid neural network of Section 3.7.2, this robust autoregression does not lag the actual process. Thus, the predictions are slightly better than those of the hybrid neural network.

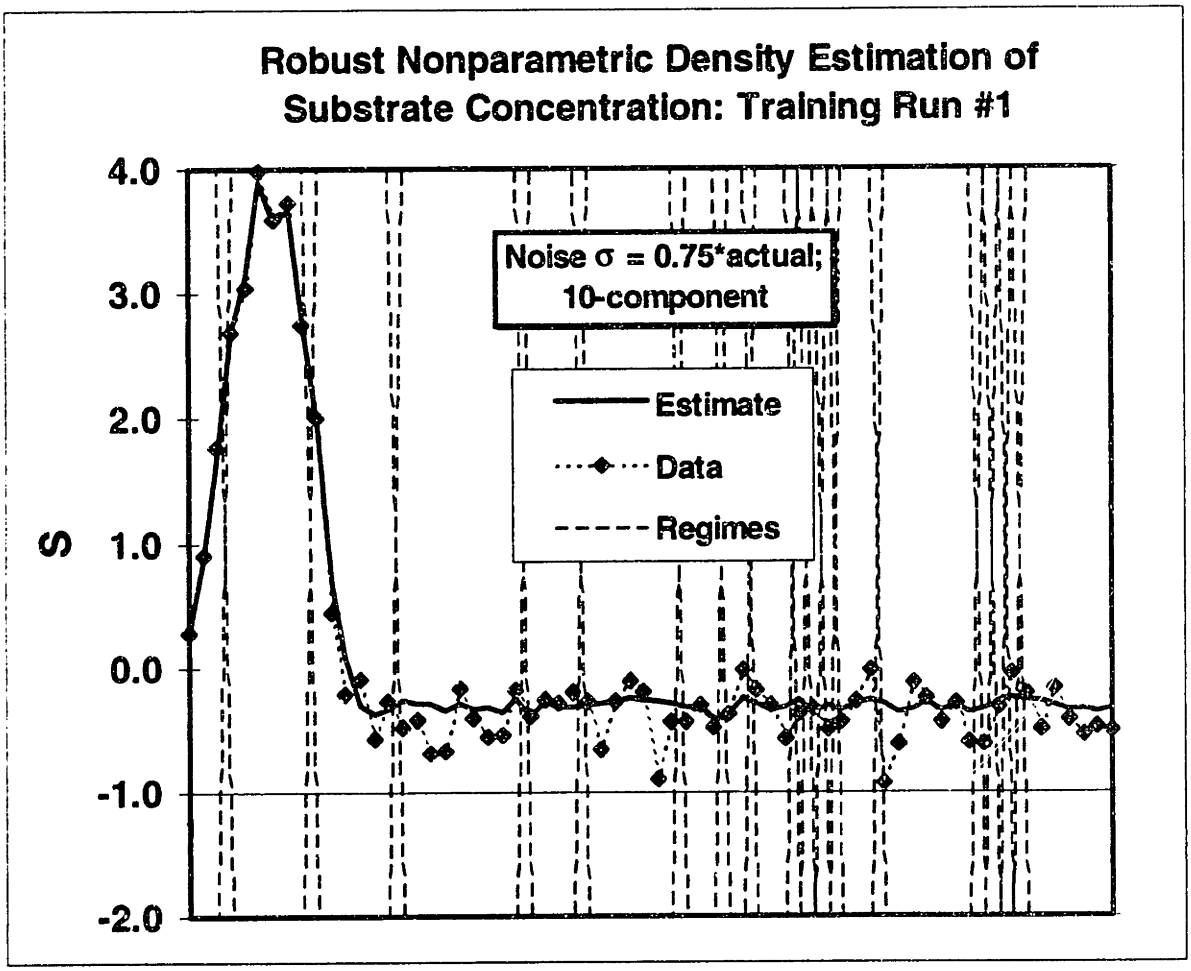


Figure 7.13: Robust nonparametric density estimation: substrate, σ=0.75*actual

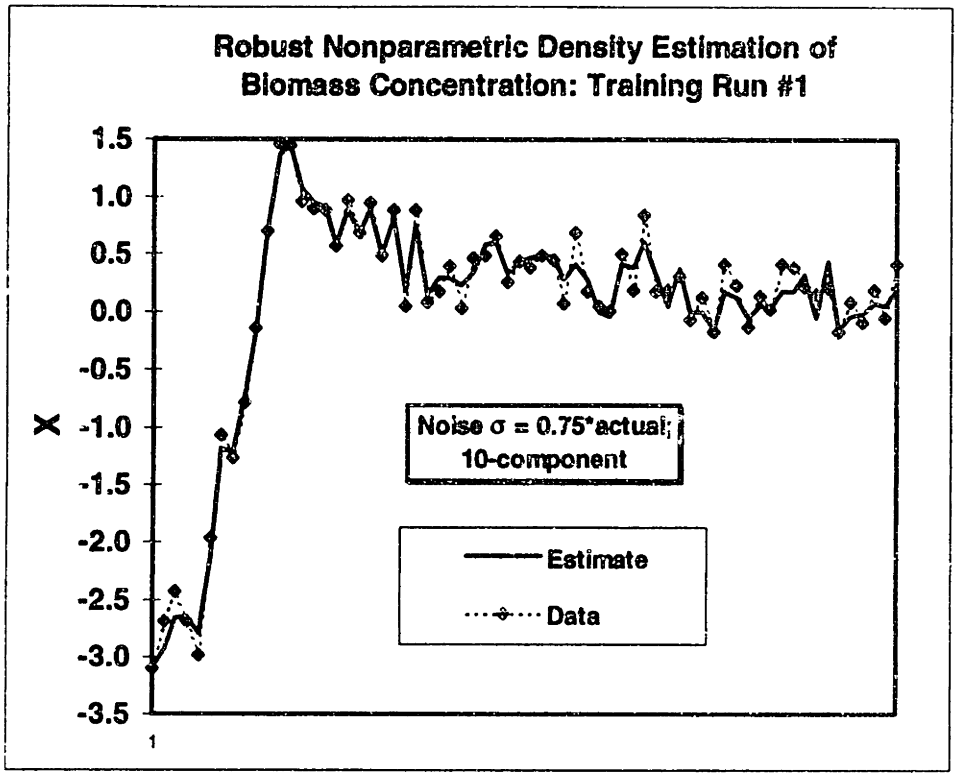


Figure 7.14: Robust nonparametric density estimation: biomass, σ=0.75*actual

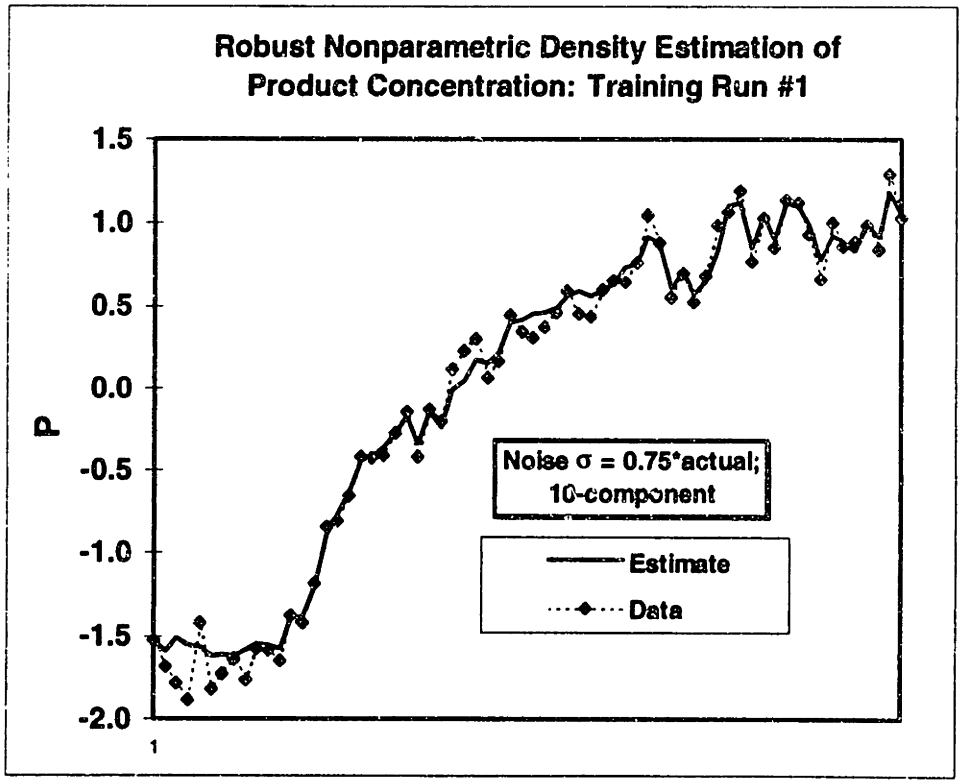


Figure 7.15: Robust nonparametric density estimation: product, σ=0.75*actual

7.3.5.2 Robust State Estimation

This section describes the results of applying the full probabilistic semiparametric model, which combines the nonparametric estimator of the previous section with the default and output models. Figures 7.16 to 7.18 show the one-step-ahead forecasts of the model assuming knowledge of the actual noise standard deviation. As expected from the results of the previous section, the semiparametric model performs well.

Given the probabilistic formulation of this problem, it is possible to update these forecasts as measurements become available. Figures 7.19 to 7.21 show the updated forecasts, which for this problem are the average of the forecasts in Figs. 7.16 to 7.18, respectively, and the measurement at each time. In this case study, the update is a simple arithmetic average because the uncertainty in the forecasts, assuming delta functions for the local models as we have done, is the same as the uncertainty in the measurements. It is obvious, that the data point influences the forecasts by adding more variance in the trajectory.

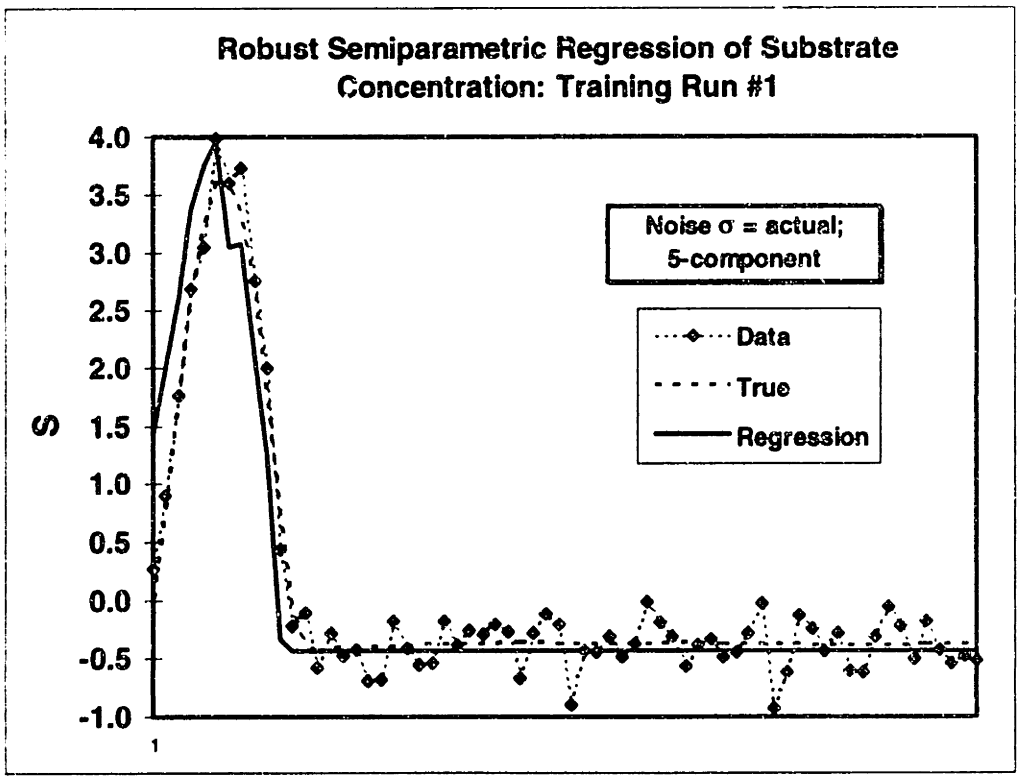


Figure 7.16: One-step-ahead forecasts: substrate, σ=actual

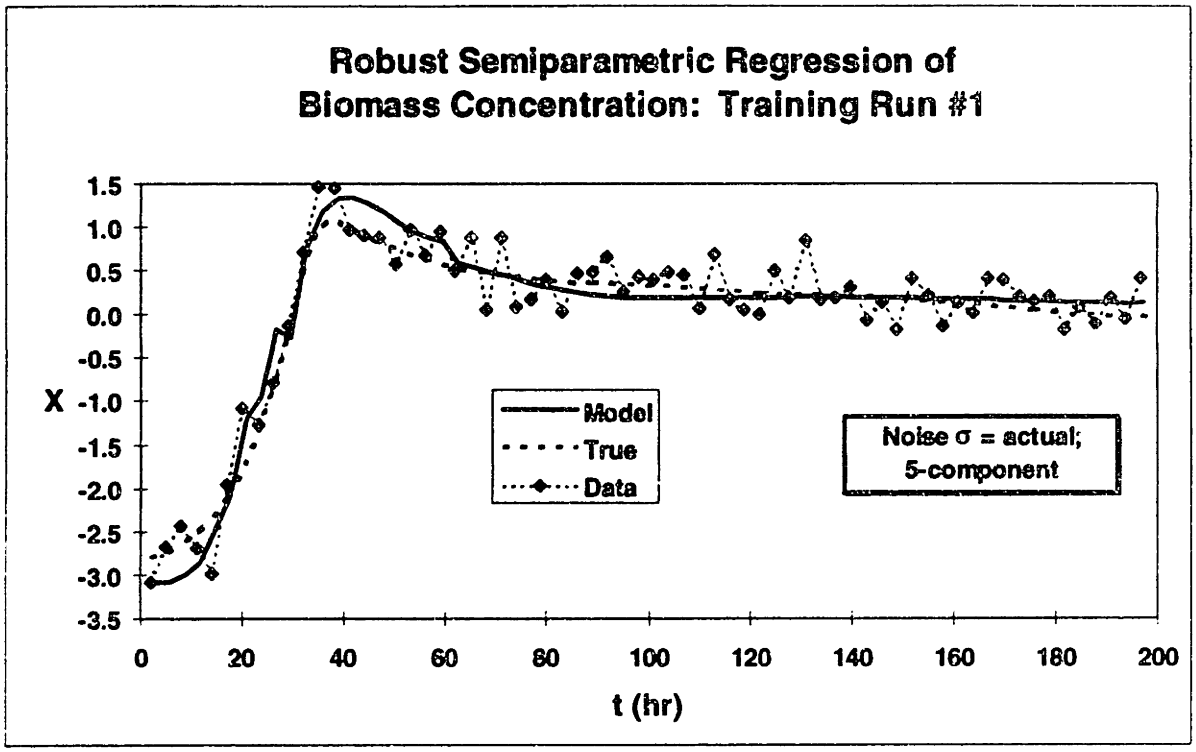


Figure 7.17: One-step-ahead forecasts: biomass, σ=actual

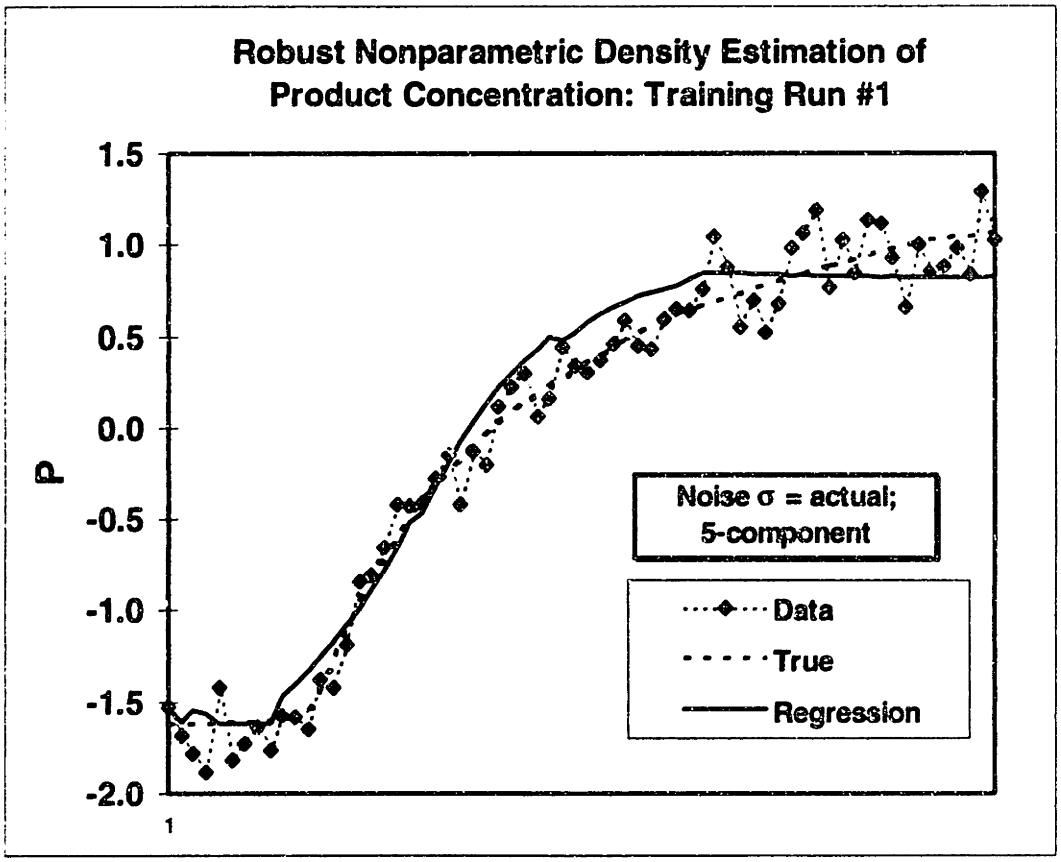


Figure 7.18: One-step-ahead forecasts: product, σ=actual

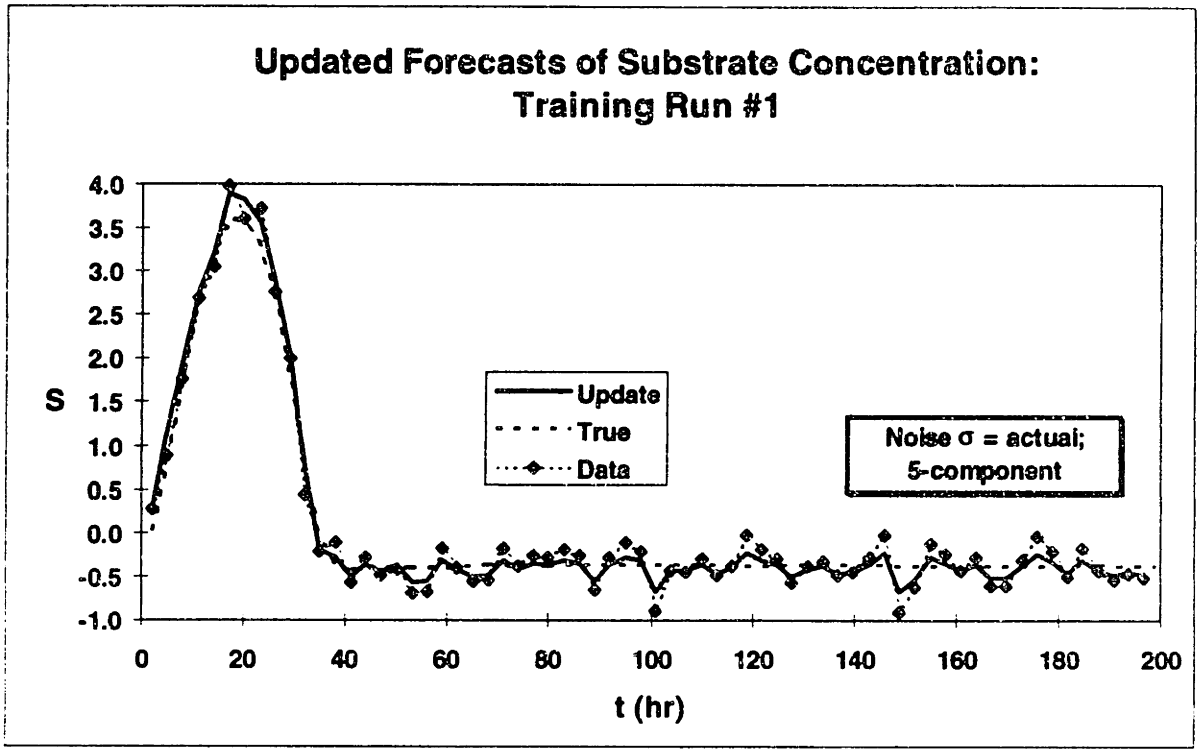


Figure 7.19: Updated forecasts: substrate, σ=actual

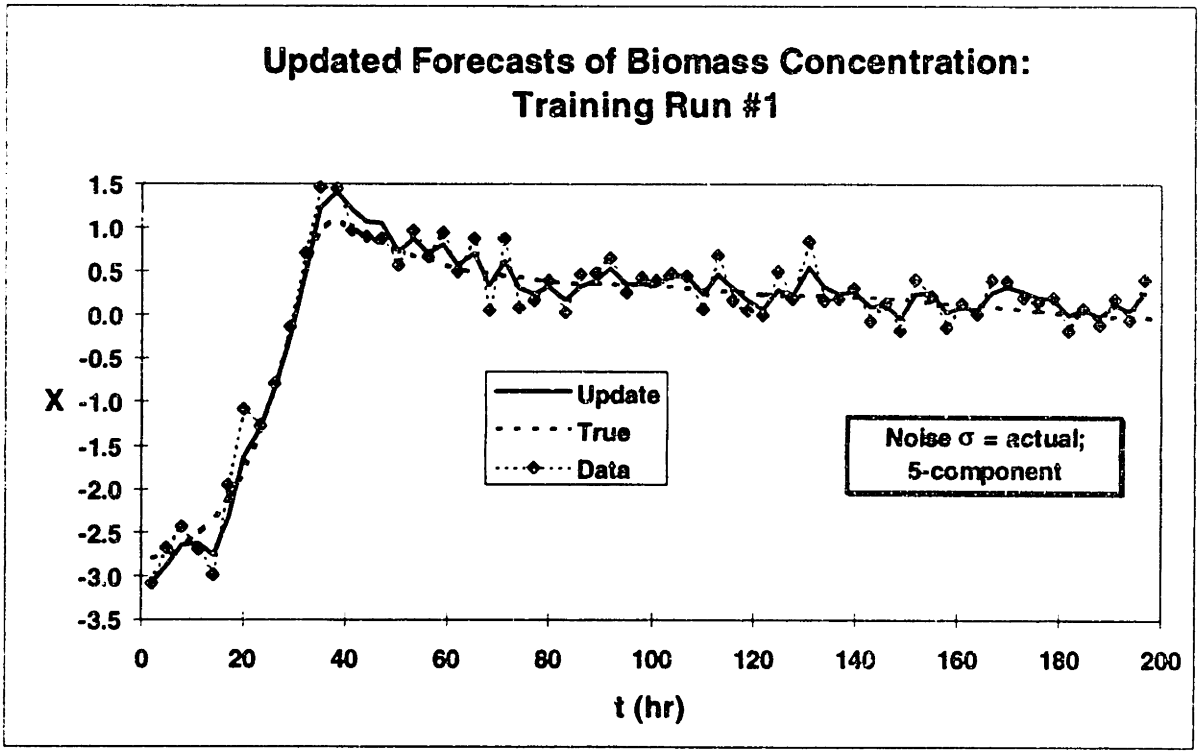


Figure 7.20: Updated forecasts: biomass, σ=actual

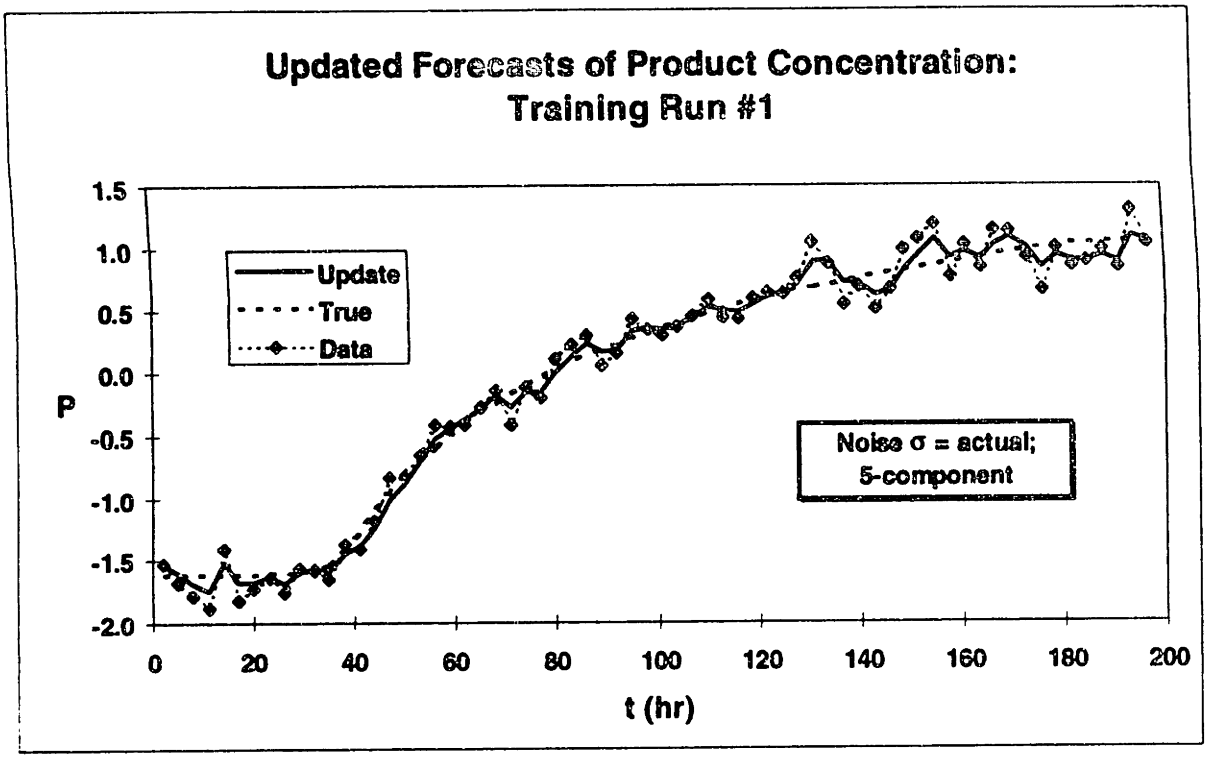


Figure 7.21: Updated forecasts: product, σ=actual

Chapter 8

Summary

This research demonstrated a practical means of combining prior knowledge with nonparametric models. It did so by applying probability theory to the domain of process modeling. The results show that the theory provides a coherent basis for pragmatic solutions in process modeling. These solutions resolve the following difficulties faced by modelers in the chemical and biochemical process industries:

- Applying constraints dictated by process design, operating policy and physical laws in the synthesis of data-derived models.
- Maximizing the information obtainable from small data sets.
- Explicitly accounting for the effects of unmeasured phenomena.
- Utilizing dynamic and steady state data simultaneously.
- Combining multiple models.
- Extrapolating reliably.
- Accurately estimating the true process state despite missing values and gross errors in the data.
- Inferring and accounting for the presence of process faults.

The results of this research are applicable to other areas of systems engineering and quantitative modeling, in general. The next section briefly discusses some of these applications. Some of the more specific issues that could still benefit from further investigation are mentioned in Section 8.2. All told, this research serves as a demonstration of the promise that probability theory holds for engineers, scientists, applied mathematicians, statisticians and econometricians in synthesizing accurate models.

8.1 Potential Impact in Other Areas of Systems Engineering

As the foundation for plausible reasoning and, hence, scientific inference, probability theory has broad applicability. Therefore, the research documented above has the potential to touch all endeavors associated with quantitative process modeling. Within the process industries, the integrated automation of the scheduling, control and analysis of processes has become commonplace. Yet, a uniform representation of the uncertainty and stochastic phenomena inherent in an industrial process could potentially enhance such automated systems. For example, data rectification is an essential part of a modern process plant's monitoring and control system. In this research and the work of Johnston and Kramer (1995) and Johnston *et al.* (1996), the probabilistic paradigm has been shown to improve the accuracy of these methods. Obviously, the increased accuracy of the data will improve control and analysis of the process. Likewise, improvements in the accuracy of predictive models has great potential to improve the performance of control, design, and diagnostic technologies. Possible future research projects in some of these areas are mentioned in the next section.

8.2 Future Research

Several research projects that could benefit from the theoretical basis elucidated in this work are listed below. They attack the important problems of developing more sensitive methods for quality control and safety through probabilistic analysis; deriving adaptive controllers that are robust to process migration and sensor failure; and implementing probabilistic fault diagnosis of combined discrete and continuous dynamic processes. Moreover, the development of computational tools is necessary to bring this technology to its full fruition in large-scale industrial process modeling. Ultimately, an "umbrella" project that integrates these tasks in a single interactive process design and engineering system is foreseeable.

8.2.1 Statistical Process Control

Statistical process control (SPC) is a key tactic in improving product quality. Future research in this area would go well beyond the approaches to SPC currently used. The objective would be to determine the true underlying probability distribution of the process variables and subsequently draw inferences about process status and product quality on this more accurate distribution. Unlike current approaches that implicitly assume normal distributions for the process variables or their principal components (e.g., MacGregor *et al.*, 1994), this approach would employ mixtures of normals to represent arbitrary pdfs and also explicitly incorporate known process relations. The results sought would be methods for SPC that are more sensitive and thus, would allow potential process faults to be averted. Such a tool might be capable of providing earlier warning to process faults and consequently reduce the costs and increase the safety of process operation.

8.2.2 Robust Adaptive Control

Robust adaptive process control can be posed as a problem of probabilistic inference. When represented probabilistically, the parameters of the controller are updated by applying Bayes' Theorem. A future project in this area would derive a formal statement of the probabilistic adaptive control problem, the form of the associated probability distributions, and a fast real-time solution methodology. The resultant controller should be robust to uncertainty in the parameters of the model and the controller.

8.2.3 Fault Diagnosis

Fault diagnosis attempts to identify root causes of process faults. Probabilistic approaches have successfully accomplished this as demonstrated by Rojas-Guzmán (1995), using Bayesian belief networks for discrete-variate problems. However, improvements can be made by using continuous as well as discrete probability distributions to represent process knowledge. The result would be a diagnosis problem that, in general, could require

greater computation than the methods proposed in this thesis, but the problem should still be tractable.

8.2.4 Computational Methods

The research in this thesis focused on small-scale problems in order to more easily implement and interpret the behavior of the methodology. It still remains to determine the performance of the proposed methods to large-scale industrial processes. Research to address these implementation issues and to provide an engineer with reliable, easy-to-use and efficient tools to solve problems in systems engineering is necessary.

8.2.5 Integrated Environment for Systems Engineering

Finally, the development of an intelligent simulation, design and control package that applies the advances we achieve in computational learning and probabilistic inference is foreseeable. Such a package should be valuable in delivering the capabilities of this research to engineers in industry, as well as being useful in the education of and the research conducted by engineering students in academia.

References

1. Abraham, B. and Chuang, A. (1993), "Expectation-Maximization algorithms and the estimation of time series models in the presence of outliers." *Journal of Time Series Analysis* **14**(3): 221.

2. Abu-Mostafa, Y. S. (1990), "Learning from hints in neural networks," *Journal of Complexity* **6**: 192-198.

3. Akaike, H. (1970), "Statistical predictor identification," *Ann. Inst. Stat. Math.* **22**: 203-217.

4. Akaike, H. (1974), "A new look at the statistical model identification," *IEEE Trans. Auto. Control* **19**: 716-723.

5. Barron, A. R., (1994), "Approximation and estimation bounds for artificial neural networks", *Machine Learning* **14**(1):115

6. Baum, E. B. and Haussler, D. (1989), "What size net gives valid generalization?" *Neural Computation* **1**: 151-160.

7. Berger, J. O. (1985), *Statistical Decision Theory and Bayesian Analysis*, New York: McGraw-Hill.

8. Bernardo, J. M. and Smith, A. F. M. (1994), *Bayesian Theory*, New York: Wiley.

9. Bhat, N. V. and McAvoy, T. J. (1992), "Determining model structure for neural models by network stripping," *Computers in Chemical Engineering* **16** (4): 271-281.

10. Bhat, N. V., Minderman, P. A., McAvoy, T. J. and Wang, N. S. (1990), "Modeling chemical process systems via neural computation," *IEEE Control Sys. Mag.* **10**: 24.

11. Bishop, C. (1991), "Improving the generalization properties of radial basis function neural networks," *Neural Computation* **3**: 579.

12. Bishop, C. (1992), "Exact calculation of the Hessian matrix for the multilayer perceptron," *Neural Computation* **4**: 494.

13. Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M. K. (1987), "Occam's razor," *Information Processing Letters* **24**: 377-380.

14. Box, G. E. P., Hunter, W. G. and Hunter, J. S. (1978), *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*, New York: Wiley.

15. Box, G. E. P. and Tiao, G. C. (1973), *Bayesian Inference in Statistical Analysis*, Reading, Massachusetts: Addison-Wesley.

16. Buja, A., Hastie, T. and Tibshirani, R. (1989), "Linear smoothers and additive models" (with discussion), *The Annals of Statistics*, **17**: 453.

17. Buntine, W. and Stirling, D. (1991), "Interactive induction," *Machine Intelligence 12*, eds. J.E. Hayes, D. Michie, and E. Tyugu, Oxford: Clarendon Press.

18. Buntine, W. L. (1991), "Myths and legends in learning classification rules," *Machine Learning: Proc. of 8th International Workshop (ML91).*

19. Buntine, W. L. and Weigend, A. S. (1991), "Bayesian back-propagation," *Complex Systems* **5**: 603.

20. Buntine, W. L. and Weigend, A. S. (1991), "Computing second derivatives in feed-forward networks: A review," *IEEE Transactions on Neural Networks.*

21. Burman, P. A. (1989), "A comparative study of ordinary cross validation, v-fold cross validation and the repeated learning-testing methods," *Biometrika*, **76**: 503-514.

22. Casella, G. and George, E. I. (1992), "Explaining the Gibbs sampler," *American Statistician*, (**46**) 3: 167.

23. Celmins, A. (1983), "Least squares optimization with implicit equations," appears in Fiacco, A. V. (ed.), *Mathematical Programming with Data Perturbations II*, 131-152, New York: Marcel Dekker.

24. Chatterjee, S. and Hadi, A. S. (1988), *Sensitivity Analysis in Linear Regression*, New York: Wiley.

25. Chen, C. and Liu, L-M. (1993), "Joint estimation of model parameters and outlier effects in time series." *Journal of the American Statistical Association.* **88** (421): 284.

26. Cherkassky, V. and Lari-Najafi, H. (1991), "Constrained topological mapping for nonparametric regression analysis," *Neural Networks* **4**: 27-40.

27. Choi, B. (1992), *ARMA Model Identification*, New York: Springer-Verlag.

28. Cohn, D. and Tesauro, G. (1992), "How tight are the Vapnik-Chervonenkis bounds?" *Neural Computation* **4**: 249-269.

29. Cook, R. D. and Weisberg, S. (1982), *Residuals and Influence in Regression*, New York: Chapman and Hall.

30. Cox, R. T. (1946), "Probability, frequency, and reasonable expectation," *Am. Jour. Phys.* **14**: 1-13.

31. Cox, R. T. (1961), *The Algebra of Probable Inference*. Baltimore: Johns Hopkins Press.

32. Craven, P. and Wahba, G. (1979), "Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross validation," *Numerische Mathematik*, **31**: 317.

33. Cybenko, G. (1989), "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals Sys.* **2**: 303.

34. Davison, A. C. and Hall, P. (1992), "On the bias and variability of bootstrap and cross validation estimates of error rate in discrimination problems." *Biometrika*, **79**: 279.

35. Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977), "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist. Soc. B* , **39**(1): 1-39.

36. Dente, J. and Vilela Mendes, R. (1992), "Learning from examples and generalization," *Complex Systems* **6**: 301-314.

37. Di Massimo, C., Montague, G. A., Willis, M. J., Tham, M. T. and Morris, A. J. (1992), "Towards improved penicillin fermentation via artificial neural networks," *Computers in Chemical Engineering* **16**(4): 283-291.

38. Duan, N. and Li, K-C. (1991), "Slicing regression: a link-free regression method," *Annals of Statistics* **19**: 505.

39. Efron, B. (1982), *The Jackknife, The Bootstrap, and Other Resampling Plans*, Philadelphia: SIAM.

40. Eubank, R. L. (1988), *Spline Smoothing and Non-parametric Regression*, New York: Marcel Dekker.

41. Eubank, R. L. (1984), "The hat matrix for smoothing splines," *Statistics and Probability Letters*, **2**: 9-14.

42. Fahlman, S. E. and Lebiere, C. (1990), "The cascade-correlation learning architecture," *Neural Information Processing Systems*.

43. Fan, J. and Marron, J. S. (1992), "Best possible constant for bandwidth selection," *Annals of Statistics* **20**(4): 2057.

44. Fiacco, A. V. (1983), *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*, New York: Academic Press.

45. Fiacco, A. V. and Kortanek, K. O. (1983), *Semi-Infinite Programming and Applications*, Berlin: Springer-Verlag.

46. Freedman, R. S. and Stuzin, G. J. (1991), "A knowledge-based methodology for tuning analytical models," *IEEE Transactions on Systems, Man, and Cybernetics* **21**(2): 347-358.

47. Friedman, J. and Stuetzle, W. (1981), "Projection pursuit regression," *Journal of the American Statistical Association* **76**: 817.

48. Friedman, J. H. (1991), "Multivariate adaptive regression splines" (with discussion), *The Annals of Statistics*, **19**: 1-141.

49. Gallant, A. R. (1987), *Nonlinear Statistical Models*, New York: Wiley.

50. Gallant, A. R. and Golub, G. H. (1984), "Imposing curvature restrictions on flexible functional forms," *Journal of Econometrics* **26**: 295-321.

51. Gatsonis, C., Hodges, J. S., Kass, R. E. and Singpurwalla, N. D. (eds.) (1991), *Case Studies in Bayesian Statistics*, New York: Springer-Verlag.

52. Geman, S., Bienenstock, E. and Doursat, R. (1992), "Neural networks and the bias/variance dilemma," *Neural Computation* **4**: 1-58.

53. Golub, G. H. and Pereyra, V. (1973), "The differentiation of pseudo-inverses and nonlinear least squares problems whose variables are separate," *SIAM Journal of Numerical Analysis* **10**(2): 413-432.

54. Gruber, M. H. J. (1990), *Regression Estimators: A Comparative Study*, New York: Academic Press.

55. Gu, C. and Wahba, G. (1991), "Minimizing GCV/GML scores with multiple smoothing parameters via the Newton method," *SIAM Journal on Scientific and Statistical Computing*, **12**: 383.

56. Gu, C. and Wahba, G. (1993), "Semiparametric analysis of variance with tensor product thin plate splines," *J Roy Stat Soc, Ser B* **55**(2): 353.

57. Gull, S. F. (1989), "Developments in maximum entropy data analysis," J. Skilling (ed.), *Maximum Entropy and Bayesian Methods*, 53-71, Boston: Kluwer Academic Publishers.

58. Gustaffson, T. K. (1989), "Dynamic modelling and reaction invariant control of pH," *Chemical Engineering Science* **38**: 389.

59. Hall, P., and Marron, J. S. (1991), "Local minima in cross validation functions," *Journal of the Royal Statistical Society, Series B*, **53**: 245.

60. Hall, P., Sheather, S. J., Jones, M. C. (1991), "On optimal data-based bandwidth selection in kernel density estimation," *Biometrika* **78**(2): 263.

61. Hall, R. C. and Seborg, D. E. (1989), "Modelling and self-tuning control of a multivariable pH neutralization process. Part I: Modelling and multiloop control," in *Proc. 1989 American Control Conference, v. 2, Pittsburgh.*

62. Härdle, W. (1990), *Applied Nonparametric Regression*, Cambridge: Cambridge University Press.

63. Härdle, W. (1991), *Smoothing Techniques: With Implementation in S*, New York: Springer-Verlag.

64. Hastie, T. J. and Tibshirani, S. (1990), *Generalized Additive Models*, London: Chapman and Hall.

65. Haussler, D. (1988), "Quantifying Inductive Bias: AI Learning algorithms and Valiant's learning framework," *Artificial Intelligence* **36**: 177-221.

66. Hornik, K., Stinchcombe, M. and White, H. (1989), "Multi-layer feedforward networks are universal approximators," *Neural Networks* **2**: 359.

67. Hyun, J. C., Graessley, W. W. and Bankoff, S. G. (1976), "Continuous polymerization of vinyl acetate - I: Kinetic modeling," *Chemical Engineering Science* **31**: 945-952.

68. Jacobs, R. A. and Jordan, M. I. (1991), "A competitive modular connectionist architecture," *Advances in Neural Information Systems, 3*; Lippman, R. P., *et al.* (eds.), San Mateo, California: Morgan Kaufmann.

69. Jacobs, R. A., Jordan, M. I. and Barto, A. G. (1991), "Task decomposition through competition in a modular connectionist architecture: the what and where vision tasks," *Cognitive Science* **15**: 219.

70. Jaynes, E. T. (1993), *Probability Theory: The Logic Of Science*, unpublished manuscript, St. Louis, Missouri: Washington University.

71. Jaynes, E. T. (1983), *E. T. Jaynes: Papers on Probability, Statistics, and Statistical Physics.* Boston: Kluwer.

72. Joerding, W. H. and Meador, J. L. (1991), "Encoding *a priori* information in feedforward networks," *Neural Networks* **4**: 847.

73. Johansen, T. A. (1994), *Operating Regime Based Process Modeling and Identification.* Ph. D. Thesis, Department of Engineering Cybernetics, The Norwegian Institute of Technology – University of Trondheim.

74. Johansen, T. A. and Foss, B. A. (1992a), "Representing and learning unmodelled dynamics with neural network memories," *Proc. Amer. Control Conf.*, 3703.

75. Johansen, T. A. and Foss, B. A. (1992b), "Nonlinear local model representation for adaptive systems," *IEEE Singapore Intl. Conf. on Intell. Control & Instr.*

76. Johnston, L. P. M. and Kramer, M. A. (1994), "Probability density estimation using elliptical basis functions," *AIChE J* **40**(10): 1637.

77. Johnston, L. P. M. and Kramer, M. A. (1995), "Maximum likelihood data rectification: Steady-state systems," *AIChE J* **41**(11): 2415-2426.

78. Johnston, L. P. M., Thompson, M. L. and Kramer, M. A., (1996), "An efficient algorithm for data rectification," in preparation.

79. Jordan, M. I. (1992), "Constrained supervised learning," *Jour. Math. Psych.* **36**: 396.

80. Jordan, M. I. and Jacobs, R. A. (1992), "Hierarchies of adaptive experts," *Neural Information Systems, 4*; Moody, J. *et al.* (eds.), San Mateo, California: Morgan Kaufmann.

81. Jordan, M. I. and Rumelhart, D. E. (1992), "Forward models: Supervised learning with a distal teacher," *Cognitive Science* **16**: 307.

82. Kapur, J. N. and Kesavan, H. K. (1992), *Entropy Optimization Principles with Applications*, Boston: Academic Press.

83. Kramer, M. A., Thompson, M. L. and Bhagat, P. M. (1992), "Embedding theoretical models in neural networks," *Proc. Amer. Control Conf., Chicago.*

84. Krishnapuram, R. and Joonwhoan, L. (1992), "Fuzzy-set-based hierarchical networks for information fusion in computer vision," *Neural Networks* **5**: 335-350.

85. Lange, K. (1995), "A gradient algorithm locally equivalent to the EM algorithm," *J. R. Statist. Soc. B* **57**(2): 425.

86. Lee, D. K. C. (1990), "Cross validation in semiparametric models: Some Monte Carlo results," *Journal Statistical Computation and Simulation*, **37**: 171.

87. Leonard, J. A. and Kramer, M. A. (1990), "Improvement of the backpropagation algorithm for training neural networks," *Comp. Chem. Engng.* **14**: 337.

88. Levin, E., Tishby, N. and Solla, S. A. (1989), "A statistical approach to learning and generalization," *Colt '89: Proc. of 2nd Ann. Wkshp. on Computational Learning Theory*: 245-260, San Mateo, California: Morgan Kaufman.

89. Lin, C-T. and Lee, C. S. G. (1991). "Neural net-based fuzzy logic control and decision system," *IEEE Transactions on Computers* **40**(12): 1320-1336.

90. Little, R. J. A. (1992), "Regression with missing X's: A review," *J. Amer. Stat. Assoc.*, **87** (420): 1227.

91. Little, R. J. A. and Rubin, D. B. (1987), *Statistical Analysis with Missing Data*, New York: John Wiley.

92. Ljung, G. M. (1993), "On outlier detection in time series," *J. R. Statist. Soc. B*, **55** (2): 559-567.

93. Ljung, L. (1987), *System Identification: Theory for the User*, Englewood Cliffs, NJ: Prentice-Hall.

94. Lu, H. and Mathis, F. H. (1992), "Surface approximation by spline smoothing and generalized cross validation," *Mathematics and Computers in Simulation*, **34**: 541.

95. Lukas, M. A. (1993), "Asymptotic optimality of generalized cross validation for choosing the regularization parameter," *Numerische Mathematik*, **66**: 41.

96. MacKay, D. J. C. (1992a), "Bayesian interpolation," *Neural Computation* **4**: 415-447.

97. MacKay, D. J. C. (1992b), "A practical Bayesian framework for backpropagation networks," *Neural Computation* **4**: 448-472.

98. MacKay, D. J. C. (1992c), "The evidence framework applied to classification networks," *Neural Computation* **4**: 720.

99. Mardia, K. V., Kent, J. T. and Bibby, J. M. (1979), *Multivariate Analysis*, London: Academic Press.

100. Mavrovouniotis, M. L. and Chang, S. (1992), "Hierarchical neural networks," *Computers in Chemical Engineering* **16**(4): 347-369.

101. McCullagh, P. and Nelder, J. A. (1989), *Generalized Linear Models*, London: Chapman and Hall.

102. Moody, J. and Darken, C. J. (1989), "Fast learning of locally-tuned processing units," *Neural Computation* **1**: 281.

103. Moody, J. E. (1992), "The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems," appears in Moody, J. E., Hanson, S. J., and Lippmann, R. P. (eds.), *Advances in Neural Information Processing Systems 4*, San Mateo, California: Morgan Kaufmann.

104. Mueller, P., Erkanli, A., and West, M. (1992), "Bayesian curve fitting using multivariate normal mixtures," Technical Report 92-A09, Institute of Statistics and Decision Sciences, Duke University.

105. Murray-Smith, R. (1994), *A Local Model Network Approach to Nonlinear Modelling*, Ph.D. Thesis, The University of Strathclyde, Glasgow.

106. Namatame, A., and Tsukamoto, Y. (1991), "Flash learning for a multi-layer network," INNS/IEEE, ed. *Int'l Joint Conf. on Neural Networks, '91, Seattle II*: 53-57.

107. Neal, R. M. (1993), "Probabilistic inference using Markov chain Monte Carlo methods," Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto.

108. Neal, R. M. (1995), *Bayesian Learning Networks*, Ph. D. Thesis, Department of Computer Science, University of Toronto.

109. O'Sullivan, F. and Wahba, G. (1985), "A cross-validated Bayesian retrieval algorithm for nonlinear remote sensing experiments," *Journal Computational Physics*, **59**: 441.

110. Pearl, J., (1988), *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, San Mateo, California: Morgan Kaufmann Publishers.

111. Pearlmutter, B. A. (1994), "Fast exact multiplication by the Hessian," *Neural Computation* **6**(1): 147.

112. Plutowski, M., Sakata, S. and White, H. (1994), "Cross-validation estimates IMSE," appears in Moody, J. E., Hanson, S. J., and Lippmann, R. P. (eds.), *Advances in Neural Information Processing Systems 6*, San Mateo, California: Morgan Kaufmann, 391.

113. Poggio, T. and Girosi, F. (1990a), "Networks for approximation and learning," *Proc. IEEE* **78**(9): 1481.

114. Poggio, T. and Girosi, F. (1990b), "Regularization algorithms for learning that are equivalent to multilayer networks," *Science* **247**: 978.

115. Pollard, J. F., Brousard, M. R., Garrison, D. B. and San K. Y. (1992), "Process identification using neural networks," *Comp. Chem Engng.* **16**(4): 253.

116. Pólya, G. (1954), *Mathematics and Plausible Reasoning*, 2 Vols., Princeton, New Jersey: Princeton University Press.

117. Psichogios, D. C. and Ungar, L. H. (1991), "A hybrid neural network-first principles approach to process modeling," *AIChE Journal* **38**(10): 1499-1511.

118. Rao, R. B., Lu, S. C-Y. and Stepp R. E. (1991), "Knowledge-based equation discovery in engineering domains," *Machine Learning: Proc. of the 8th International Workshop*, 645-649.

119. Rivest, R.L. and Sloan, R. (1988), "Learning complicated concepts reliably and usefully," *Seventh National Conference on Artificial Intelligence, Saint Paul, Minnesota*: 635-640.

120. Rojas-Guzmán, C. (1995), *An Evolutionary Programming Approach to Probabilistic Model-Based Fault Diagnosis of Chemical Processes*, Ph. D. Thesis, Department of Chemical Engineering, Massachusetts Institute of Technology.

121. Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986), "Learning internal representations by error propagation," in *Parallel Distributed Processing*, eds. Rumelhart, D. E.. and McClelland, J. L., Cambridge, Massachusetts: MIT Press, 318.

122. Sanger, T. D. (1991), "Optimal hidden units for two-layer nonlinear feedforward neural networks," *International Journal of Pattern Recognition and Artificial Intelligence* **5**(4): 545-561.

123. Schick, I. C. and Mitter, S. K. (1994), "Robust recursive estimation in the presence of heavy-tailed observation noise," *Annals of Statistics*, **22**(2): 1045-1080.

124. Schumaker, L. L. and Utreras, F. I. (1990), "On generalized cross validation for tensor smoothing splines," *SIAM Journal on Scientific and Statistical Computing*, **11**: 713.

125. Sclove, S. L. (1993), "Small-sample and large-sample statistical model selection criteria." Appears in Cheeseman, P. and Oldford, R. W. (eds.), *Selecting Models from Data: Artificial Intelligence and Statistics IV*, New York: Springer-Verlag, 31-39.

126. Shao, J. (1993), "Linear model selection by cross validation." *Journal of the American Statistical Association*, **88**: 486.

127. Sibisi, S. (1989), "Regularization and inverse problems," *Maximum Entropy and Bayesian Methods*, J. Skilling, ed., 389-396, Boston: Kluwer Academic Publishers.

128. Silverman, B. W. (1986), *Density Estimation for Statistics and Data Analysis*, New York: Chapman and Hall.

129. Sin, S-K. and DeFigeiredo, R. J. P. (1991), "An incremental fine adjustment algorithm for the design of optimal interpolating neural networks," *International Journal of Pattern Recognition and Artificial Intelligence* **5**(4): 563-579.

130. Skilling, J. (1989), "Classic maximum entropy," J. Skilling (ed.), *Maximum Entropy and Bayesian Methods*, 45-52, Boston: Kluwer Academic Publishers.

131. Skilling, J. and Gull, S. F. (1987), "Prior knowledge must be used," C. R. Smith and G. J. Erickson (eds.), *Maximum-Entropy and Bayesian Spectral Analysis and Estimation Problems*, 161-172, Boston: Kluwer Academic Publishers.

132. Specht, D. F. (1990), "Probabilistic neural networks," *Neural Networks* **3**: 109-118.

133. Spiegelhalter, D., Thomas, A., Best, N. and Gilks, W. (1995) "BUGS: Bayesian inference Using Gibbs Sampling, Version 0.30a," Cambridge, UK: MRC Biostatistics Unit.

134. Stein, M. L. (1990), "A comparison of generalized cross validation and modified maximum likelihood for estimating the parameters of a stochastic process," *The Annals of Statistics*, **18**: 1139.

135. Stone, M. (1974), "Cross validatory choice and assessment of statistical predictors," *Journal of the Royal Statistical Society, Series B*, **35**: 111.

136. Su, H.-T., Bhat, N., Minderman, P. A. and McAvoy, T. J. (1992) "Integrating neural networks with first principles models for dynamic modeling," *3rd IFAC Symp. on Dynamics and Control of Chemical Reactors, Distillation Columns, and Batch Processes (DYCORD+).*

137. Tanaka, Y., Fukushima, M. and Ibaraki, T. (1988), "A comparative study of several semi-infinite nonlinear programming algorithms," *European J. Oper. Res.* **36**: 92.

138. Tcheng, D. K., Lambert, B. L., Lu, S. C-Y. and Rendell L. A. (1991), "AIMS: an adaptive interactive modeling system for supporting engineering decision making," *Machine Learning: Proc. of the 8th International Workshop*, 645-649.

139. Templeman, A.B. and Xingsi, L. (1989), "Maximum entropy and constrained optimization," J. Skilling (ed.), *Maximum Entropy and Bayesian Methods*, 447-454, Boston: Kluwer Academic Publishers.

140. Thompson, M. L. (1984), "System analysis, control and optimization of the fed-batch penicillin fermentation," S.M. Thesis, Department of Chemical Engineering, Massachusetts Institute of Technology.

141. Thompson, M. L. and Kramer, M. A. (1994), "Modeling chemical processes using prior knowledge and neural networks," *AIChE Journal*, **40**: 1328-1340.

142. Tishby, N., Levin, E. and Solla, S. A. (1989), "Consistent inference of probabilities in layered networks: Prediction and generalization," INNS/IEEE, ed. *Proc. of Int'l Joint Conf. on Neural Networks, Wash.DC II*: 403-409.

143. Titterington, A. F. M., Smith, U. E. and Makov, D. M. (1985), *Statistical Analysis of Finite Mixture Distributions*, New York: Wiley.

144. Utgoff, P. E. (1986), *Machine Learning of Inductive Bias*, Boston: Kluwer Academic Publishers.

145. Van Es, B. (1992), "Asymptotics for least squares cross validation bandwidths in nonsmooth cases," *The Annals of Statistics*, **20**: 1647.

146. Wada, Y., and Kawato, M. (1992), "A new information criterion combined with cross validation method to estimate generalization capability." *Systems and Computers in Japan*, **23**: 92.

147. Wahba, G. (1990), *Spline Models for Observational Data*, Philadelphia: SIAM.

148. Wei, W. W. S. (1990), *Time Series Analysis: Univariate and Multivariate Methods*, New York: Addison-Wesley.

149. Weiss, S. M. and Kulikowski, C. A. (1991), *Computer Systems that Learn*, San Mateo, California: Morgan Kaufmann.

150. White, H. (1989), "Learning in artificial neural networks: A statistical perspective," *Neural Computation* **1**: 425-464.

151. Wolpert, D. H. (1990a), "A mathematical theory of generalization: Part I," *Complex Systems* **4**: 151-200.

152. Wolpert, D. H. (1990b), "A mathematical theory of generalization: Part II," *Complex Systems* **4**: 201-249.

153. Wolpert, D. H. (1990c), "Constructing a generalizer superior to NetTalk via a mathematical theory of generalization," *Neural Networks*, **3**: 445-452.

154. Wolpert, D. H. (1990d), "The relationship between Occam's razor and convergent guessing," *Complex Systems* **4**: 319-368.

155. Wolpert, D. H. (1992), "On the connection between in-sample testing and generalization error," *Complex Systems* **6**: 47-94.

156. Zhang, P. (1993a), "On the choice of penalty term in generalized FPE criterion." appears in Cheeseman, P. and Oldford, R. W. (eds.), *Selecting Models from Data: Artificial Intelligence and Statistics IV*, 41-49, New York: Springer-Verlag.

157. Zhang, P. (1993b), "Model selection via multifold cross validation." *Annals of Statistics*, **21**: 299.

Appendix A

Notation for Chapter 3

Scalar Variables	
c	residual, $c = y - \hat{y}$
D	data, $D \equiv \{ \mathbf{x}_k, y_k : \mathbf{x}_k \in \Re^n, y_k \in \Re;\ k = 1, \ldots, N \}$
H	set of candidate models
h_{kk}	diagonal element of hat matrix $\mathbf{H}$
h_{avg}	average of the h_{kk} = trace($\mathbf{H}$)/N
K	index of candidate models; number of hidden nodes in 3-BPN
M	subset of candidate models in H
N	number of data points
n_s	number of data points in the s^{th} subset in SCV
p	number of model parameters
p_K	number of parameters in a linear model $f_K(\mathbf{x};\mathbf{w}_K)$
p_{eff}	effective number of parameters in a nonlinear model $f_K(\mathbf{x};\mathbf{w}_K)$
q	number of independent variables
u_j	input into the j^{th} hidden-layer node of a 3-BPN
X	random variable (independent variable)
Y	random variable (dependent variable)
y	data point dependent variable (realization of Y)
$\hat{y}$	model prediction $f_K(\mathbf{x};\mathbf{w})$
δy_k	perturbation of the k^{th} data point
Δy_k	total displacement of the k^{th} data point
z	zero-influence data point = leave-one-out prediction for y
α_{ij}	input weight from i^{th} input to j^{th} hidden-layer node of a 3-BPN
β_j	output weight of a 3-BPN
ε	zero-mean additive noise in y
η_{kk}	diagonal element of effective hat matrix $\hat{\mathbf{H}}$
σ^2	variance of ε

Vectors, Matrices, and Tensors	
$\mathbf{A}(\mathbf{X};\alpha)$	matrix of model basis functions evaluated at data points (e.g., hidden layer node activations of a 3-BPN) ($N\times(K+1)$)
$\mathbf{B}$	$\nabla\mathbf{f}(\mathbf{X};\mathbf{w}^*)$, the Jacobian of $\mathbf{f}$ given optimal parameters $\mathbf{w}^*$ ($N\times p$)
$\mathbf{c}$	residuals ($\mathbf{y}$ - $\mathbf{f}$) ($N\times 1$)
$\mathbf{D}(\nabla\mathbf{f}^T)$	Fréchet derivative of $\nabla\mathbf{f}^T$ ($p\times p\times N$)
∇F	gradient of objective function $F(\mathbf{w})$ ($p\times 1$)
$\mathbf{f}$	model predictions $\mathbf{f}(\mathbf{X};\mathbf{w})$ ($N\times 1$)
$\nabla\mathbf{f}$	Jacobian of $\mathbf{f}$ with respect to $\mathbf{w}$ ($N\times p$)
$\mathbf{H}$	hat (influence) matrix for linear model ($N\times N$)
$\hat{\mathbf{H}}$	effective hat (influence) matrix for nonlinear model ($N\times N$)
$\mathbf{M}(\delta\mathbf{y})$	Hessian with respect to $\mathbf{w}$ of F evaluated at $\delta\mathbf{y}$ ($p\times p$)
$\mathbf{N}(\delta\mathbf{y})$	negative Jacobian of ∇F with respect to $\delta\mathbf{y}$ evaluated at $\delta\mathbf{y}$ ($p\times N$)
$\mathbf{Q}(\alpha)$	Moore-Penrose generalized inverse of $\mathbf{A}(\mathbf{X};\alpha)$ ($(K+1)\times N$)
$\mathbf{w}$	model parameters ($p\times 1$)
$\mathbf{X}$	data design matrix of independent variables ($N\times p$)
$\mathbf{x}$	independent variables (realization of X) ($n\times 1$)
$\mathbf{y}$	dependent variables in data ($N\times 1$)
$\delta\mathbf{y}$	data perturbations ($N\times 1$)
α	subset of $\mathbf{w}_K$ with respect to which $\mathbf{f}_K(\mathbf{x};\mathbf{w}_K)$ is nonlinear $[K(n+1)]\times 1$
β	subset of $\mathbf{w}_K$ with respect to which $\mathbf{f}_K(\mathbf{x};\mathbf{w}_K)$ is linear $(K+1)\times 1$

Functions	
$a(\mathbf{x})$	sigmoid function
$a_i(\mathbf{x})$	i^{th} basis function of model (e.g., sigmoid activation function for i^{th} hidden layer node in 3-BPN)
$d(K)$	model degrees of freedom (equals p_{eff} for nonlinear model)
$F(\mathbf{w}_K;f_K,D)$, $F(\mathbf{w})$	objective function used in parameter estimation
$f(\mathbf{x})$	true (unknown) model
$f_K(\mathbf{x};\mathbf{w}_K)$	candidate model
$J(f_K;\mathbf{w}_K)$	prediction accuracy metric for model selection
$N(0,\sigma^2)$	normal probability distribution with mean 0 and variance σ^2
$p(X)$	marginal probability distribution of X
$p(X,Y)$	joint probability distribution of X and Y
$p(Y\|X)$	conditional probability distribution of Y given X
trace($\mathbf{H}$)	sum of the diagonal elements of $\mathbf{H}$
$\mathbf{w}_h^*(\mathbf{y})$	vector function relating optimal model parameters to data
$\mathbf{w}(\delta\mathbf{y})$	vector function relating model parameters to data perturbations
$\Phi(K)$	penalty function in complexity-penalized criteria

Subscripts	
avg	average
eff	effective (estimated)
K	index for candidate models
i,j	generic index
k	index for data points, k=1,...,N
$[k]$	leave out k^{th} data point in estimating the parameters
s	index for subsets used in S-fold cross validation
T	independent test set
Superscripts	
T	transpose
$*$	optimal
(i)	i^{th} iteration of Newton's method
-1	matrix inverse
Special	
$\Re$	the set of real numbers
Δ	change or displacement
∇	partial differentiation with respect to a vector
$\sim$	distributed according to
Abbreviations	
3-BPN	three-layer back-propagation neural network
AIC	Akaike's information criterion
ASE	averaged squared error
CV	cross validation
FCV	fast cross validation
FPE	final prediction error
GCV	generalized cross validation
GPE	generalized prediction error
MARS	multivariate adaptive regression splines
MISE	mean integrated squared error
PMSE	prediction mean squared error
RLT	repeated learning testing
SCV	S-fold cross validation
SLMP	strictly local minimizing point

Appendix B

Useful Formulas for Normal Distributions

This appendix contains formulas that are useful in manipulating multivariate normal distributions and finite mixtures of normals. They were used in deriving the EM formulas of Chapter 6. Many more such useful identities and theorems about the multivariate normal distribution and linear algebra (matrix manipulation) are collected in Mardia, *et al.*, (1979), which is a good reference book on multivariate statistical analysis.

In what follows, let a multivariate normal (gaussian) pdf of $x \in \Re^n$ with mean μ and covariance matrix Σ be represented by the following expression:

$$\phi(x;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^n|\Sigma|}} \exp\left[-\tfrac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right].$$

B.1. Product of Two Normal Distributions

The product of two normal distributions in x is proportional to a single normal in x:

$$\phi(x;\mu_1,\Sigma_1)\phi(x;\mu_2,\Sigma_2) = \phi(\mu_1;\mu_2,\Sigma_1+\Sigma_2)\phi(x;u,S) = \phi(\mu_2;\mu_1,\Sigma_1+\Sigma_2)\phi(x;u,S),$$

where $S \equiv \left(\Sigma_1^{-1}+\Sigma_2^{-1}\right)^{-1}$, and $u \equiv S\left(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2\right)$.

B.2. Convolution of Two Normal Distributions

The convolution of two normal distributions in x equals a normal distribution:

$$\begin{aligned}\int \phi(x-y;\mu_1,\Sigma_1)\phi(x;\mu_2,\Sigma_2)dx &= \int \phi(y;\mu_1+\mu_2,\Sigma_1+\Sigma_2)\phi(x;u,S)dx \\ &= \phi(y;\mu_1+\mu_2,\Sigma_1+\Sigma_2),\end{aligned}$$

where $S \equiv \left(\Sigma_1^{-1}+\Sigma_2^{-1}\right)^{-1}$, and $u \equiv S\left(\Sigma_1^{-1}\mu_1+\Sigma_2^{-1}\mu_2\right)$.

B.3. Finite Mixtures of Normal Distributions

A pdf $p(x|H)$ representable as a finite mixture distribution with normal components $\phi(x;\mu_j,\Sigma_j)$ (i.e., $p(x|H)=\sum_{j=1}^{K} q_j\phi(x;\mu_j,\Sigma_j)$) and binary indicator variables z (i.e., z_j=0 or 1, and one and only one z_j=1 $\forall\, j$=1,2,...,K) has a joint pdf $p(x,z|H)$ proportional to a single normal distribution in x:

$$p(x,z|H)=\prod_{j=1}^{K}\left[q_j\phi(x;\mu_j,\Sigma_j)\right]^{z_j}=P(z|H)p(x|z,H)=\left(\prod_{j=1}^{K}q_j^{z_j}\right)\phi(x;u(z),S(z));$$

where

$$P(z|H)=\prod_{j=1}^{K}q_j^{z_j};\quad p(x|z,H)=\phi(x;u(z),S(z));\quad S(z)\equiv\left[\sum_{j=1}^{K}z_j\Sigma_j^{-1}\right]^{-1};\quad u(z)\equiv S(z)\sum_{j=1}^{K}z_j\Sigma_j^{-1}\mu_j.$$

B.4. Singular Normal Distributions

If covariance matrix Σ is singular with rank $r<n$, then $\phi(x;\mu,\Sigma)$ is given as

$$\phi(x;\mu,\Sigma)=\frac{1}{\sqrt{(2\pi)^r\prod_{k=1}^{r}\lambda_k}}\exp\left[-\tfrac{1}{2}(x-\mu)^{\mathsf{T}}\Sigma^{-}(x-\mu)\right],$$

where Σ^{-} is the generalized inverse of Σ, and the λ_k are the first r non-zero eigenvalues of Σ. The generalized inverse of a matrix can be computed from the singular value decomposition of the matrix: For matrix $\mathbf{A}$ of rank r with singular value decomposition $\mathbf{A}=\mathbf{USV}^{\mathsf{T}}$, where $\mathbf{S}$ is an $r\times r$ diagonal matrix with elements $s_k=(\lambda_k)^{1/2}$, the generalized inverse is $\mathbf{A}^{-}=\mathbf{VS}^{-1}\mathbf{U}^{\mathsf{T}}$ (Mardia, *et al.*, 1979).

B.5. Conditional Distribution of a Normal Distribution

If x has pdf $p(x|\theta,H)=\phi(x;\mu,\Sigma)$, and $x=[x_1^T\ x_2^T]^T$ – i.e., x is partitioned into two subvectors x_1 and x_2 – then the conditional pdf $p(x_2|x_1,\theta,H)$ is also a multivariate normal distribution: $p(x_2|x_1,\theta,H) = \phi(x_2;u(x_1),S)$. The mean and covariance matrix of the conditional distribution are as follows:

$$u(x_1) \equiv \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(x_1 - \mu_1),$$

$$S \equiv \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12},$$

where μ has been partitioned into $\mu_1=E[x_1|H]$ and $\mu_2=E[x_2|H]$; and Σ has been partitioned into $\Sigma_{11}=E[(x_1-\mu_1)^2|H]$, $\Sigma_{22}=E[(x_2-\mu_2)^2|H]$, and $\Sigma_{12}=\Sigma_{21}=E[(x_1-\mu_1)(x_2-\mu_2)|H]$ (Mardia, *et al.*, 1979).

B.6. Log-Likelihood for a Sample of Normally Distributed Covariates

If $X=\{x_i\}$, where $x_i \in \Re^n$; $i=1,2,...,N$; and the x_i are iid from $p(x|\theta,H)=\phi(x;\mu,\Sigma)$, then the log-likelihood $\log[p(X|\theta,H)]$ can be expressed in terms of the sufficient statistics $m = \frac{1}{N}\sum_{i=1}^{N} x_i$ and $S = \frac{1}{N}\sum_{i=1}^{N}(x_i - m)(x_i - m)^T$ as follows:

$$\log[p(X|\theta,H)] = -\frac{N}{2}\left[\mathrm{tr}(\Sigma^{-1}S) + (m-\mu)^T\Sigma^{-1}(m-\mu) + \log|2\pi\Sigma|\right],$$

where trA is defined as the trace of the square matrix A, which is the sum of the diagonal elements.

B.7. Inverse of the Sum of Two Matrices

The following matrix identity is useful in deriving the expressions above:

For square, invertible matrices A and B:

$$(A + B)^{-1} = A^{-1} - A^{-1}(A^{-1} + B^{-1})^{-1}A^{-1};$$

thus,

$$(A^{-1} + B^{-1})^{-1} = A - A(A + B)^{-1}A.$$

Appendix C

Function-Oriented Modeling Toolkit for Matlab

This is the Matlab, vers. 4.0, source code used to implement the function-oriented hybrid network methodology.

C.1. Synthesis

```
function [m,shape,w,cntrl]=
make_hyb3(x,y,m,shape,w0,cntrl,defmod,outmod,dout,a1,a2,a3,a4,a5,a6)
%  MAKE_HYB3
%  Initial call:
%   [m,shape,w]=make_hyb3(x,y,h,p,[],iters,'defmod','outmod','dout',a1,..a6)
%  Subsequent calls:
%  [m,shape,w]=make_hyb3(x,y,m,shape,w,iters, 'defmod','outmod','dout',a1,..a6)
%
%  Function for training hybrid model-based nets, given training data
%  x[ntrain,nin] and targets y[ntrain, nout]; number of hidden units
%  h and the unit overlap p; starting weights w0[h,nz] which can be
%  set to [] on initial call. iters can be replaced by cntrl vector
%  (see make_bpn), which can appear also as an optional output.  On
%  repeated calls, matrices m and shape replace scalars h and p.
%  a1 through a6 are optional arguments passed directly to user functions.
%
%  User functions:
%    'defmod' = name of function containing default model.
%               If there is no default model, replace this argument with
%               nz, the number of unconstrained variables.
%    'outmod' = name of function defining output equations (required)
%    'dout' = function for derivatives of output equations (optional).
%  See manual for details of user functions.
%
%  Outputs:
%     w =  RBF second layer weights [h,nz]
%     m = cluster centers
%     shape = RBF unit shapes
%     cntrl = Returned from optimization routine; see make_bpn
%
%   The global variable ELLIPSE = 'on' or 'off' specifies whether
%   or not elliptical units are used by MAKE_HYB.

%       Copyright (c) 1992 by Process Science, Inc.
%       Mark A. Kramer 6-15-92.
%       Revisions by Michael L. Thompson 10-17-92:
% * Allow w0 to be matrix even if m&shape are scalars (i.e. h&p are given).
% * Allow w0 to be [] even if m&shape matrices given (i.e. compute h from m).
%
[ntrain,nin] = size(x);
if(~exist('dout')) dout=[]; end;
if(isempty(defmod))
       error('If no default model then include the number of unconstrained variables in
call');
end
%  Run default model on training set
if(isstr(defmod))
  str=[defmod,'(x'];
  for i=1:nargin - 9, str = [str,',a',num2str(i)]; end
  str = [str,')'];
  zdef = eval(str); [ntrain,nz]=size(zdef);
```

```
else
  nz = defmod; zdef = zeros(ntrain,nz);
end
%  Initialize weights and set RBF centers and shapes if required
if(isempty(w0))
  rand('uniform');
  if(length(m) > 1 | length(shape) > 1)
%    error('if w0=[] then m and shape should be scalars, h and p')
       [h,nin] = size(m);
  else
       h=m; p=shape;
       m = cluster(x,h);
       if(~exist('ELLIPSE')) ELLIPSE = 'off'; end
       if(strcmp(ELLIPSE,'on') & nin ~= 1)
              shape = ep_heur(m,p,x);
       else
              shape = p_heur(m,p);
       end
  end
  if(isstr(defmod)) nvar = h*nz; else, nvar=(h+1)*nz; end
%**** Reduced magnitude of initial weights ********** MLT-10-23-92 ****
  w0 = (2*rand(nvar,1)-ones(nvar,1))*0.1/h;
else
  if(length(m)==1 | length(shape)==1)
%    error('if w given as matrix then matrices m and shape are required')
       h=m; p=shape;
       if(isstr(defmod)) nvar = h*nz; else, nvar=(h+1)*nz; end
       [nvw,dum] = size(w0);
       if nvw*dum ~= nvar,
              error('Initial w matrix inconsistent with h & nz.')
       end
       m = cluster(x,h);
       if(~exist('ELLIPSE')) ELLIPSE = 'off'; end
       if(strcmp(ELLIPSE,'on') & nin ~= 1)
              shape = ep_heur(m,p,x);
       else
              shape = p_heur(m,p);
       end
  else
       [h,nin] = size(m);
       if(isstr(defmod)) nvar = h*nz; else, nvar=(h+1)*nz; end
  end
end
ah = layer1(m,shape,x);
if(~isstr(defmod)) ah = [ah,ones(ntrain,1)]; h = h+1; end
%
%  Call train, with or without gradients
%
if(length(dout)==0)
  str='train(w0,''f_hyb'',[],cntrl,ah,zdef,y,outmod,[]';
else
  str='train(w0,''f_hyb'',''df_hyb3'',cntrl,ah,zdef,y,outmod,dout';
end
for i=1:nargin - 9, str = [str,',a',num2str(i)]; end
str = [str, ')'];
[w,cntrl]=eval(str);
w = reshape(w,h,nz);


function zdef=f_def(xi,xt,vex,params,sclf);
% function zdef=f_def(xi,xt,vex,params,sclf);
%
% Compute the default model for the fed-batch fermentation hybrid model.
% xi = [S0, X0, P0] ** scaled **
% vex = [D, Sf, dt] (not used)
% params = (mumax,Kx,qpmax,Kp,Ki,Yxs,Yps,mx,K)'
% sclf = scale factors: row 1 = means, row 2 = std devs.
%        cols 1:3 are for inputs xi; cols 4:6 for outputs y (see "f_out.m").
% ** This routine expects xi(:,1), i.e. scaled S, to equal mcv of log(S). ***
% zdef = [mu(t), sigma(t), qp(t)]
%

mumax=params(1);
Kx =params(2);
```

```
qpmax=params(3);
Kp =params(4);
Ki =params(5);
Yxs=params(6);
Yps=params(7);
mx =params(8);
K  =params(9);

logS = postprocess(xi(:,1),sclf(1,1),sclf(2,1));
S = exp(logS);
%plot((1:length(S)),S,'o')
% No cell lysis, alternative growth rate correlation and different parameters
% from actual process (see "ferm_deriv.m").
%
% Monod kinetics: specific growth rate.
% As a result: 1) High biomass conc. does not inhibit growth (overestimates).
%              2) Underestimates growth rate at low biomass conc.
mu   =mumax*S./(20*Kx + S);

% Specific product formation rate.
% Effective parameters are qpmax=1.1qpmax, Kp=0.9Kp, Ki=2Ki.
% As a result: 1) max product formation rate is 10% higher
%              2) substrate has much less of an inhibitory effect on production.
qp   =(1.1*qpmax)*S./((0.9*Kp) + S.*(1 + S/(2*Ki)));

% Specific substrate consumption rate.
% Effective parameters are Yxs=0.9Yxs, Yps=0.9Yps, mx=1.1mx.
% As a result: 1) spec. consumption rate is greater than actual process.
sigma=mu/(0.9*Yxs) + qp/(0.9*Yps) + (1.1*mx);

%plot((1:length(mu)),mu,'o',(1:length(sigma)),sigma,'+',(1:length(qp)),qp,'*')

zdef(:,1)=mu;
zdef(:,2)=sigma;
zdef(:,3)=qp;


function y=f_out(z,xt,vex,params,sclf);
% function y=f_out(z,xt,vex,params,sclf);
%
% Computes the outputs for the hybrid model of the fed-batch fermentation
% by analytically integrating the component balances under the assumption of
% constant mu, sigma, qp, F and Sf.
% *** Output quantities y are scaled ***
%
% z = [mu(t),sigma(t),qp(t)] [=] ntrain x 3.
% xt = inputs [S(t) X(t) P(t)] [=] ntrain x 3. ** unscaled **
% vex = [D(t) Sf(t) dt] [=] ntrain x 3.
% params = (mumax,Kx,qpmax,Kp,Ki,Yxs,Yps,mx,K,clm)'
% sclf = scale factors: row 1 = means, row 2 = std devs.
%        cols. 1:3 for inputs xi (see "f_def.m"); cols. 4:6 for outputs y.
% ** This routine outputs y(:,1), i.e. scaled S,as mcv of log(S). ***
% y = [S(t+dt),X(t+dt),P(t+dt)] [=] ntrain x 3. ** scaled **
%

[ndata,nz]=size(z);

K  = params(9);

S0 = xt(1:ndata,1);
X0 = xt(1:ndata,2);
P0 = xt(1:ndata,3);

D  = vex(1:ndata,1);
Sf = vex(1:ndata,2);

dt = vex(1:ndata,3);

mu   =z(:,1);
sigma=z(:,2);
qp   =z(:,3);
ind=find(mu==nan);
if length(ind)~=0,
       mu(ind)=zeros(length(mu(ind)),1);
```

```
        disp('WARNING: Bad mu in f_out2.m');
end
ind=find(sigma==nan);
if length(ind)~=0,
        sigma(ind)=zeros(length(sigma(ind)),1);
        disp('WARNING: Bad sigma in f_out2.m');
end
ind=find(qp==nan);
if length(ind)~=0,
        qp(ind)=zeros(length(qp(ind)),1);
        disp('WARNING: Bad qp in f_out2.m');
end

expmut = exp(-mu.*dt);
expmuKt = exp(-(mu+K).*dt);

beta= sigma./mu;
csi = qp./(mu+K);

vt = 1 + D.*dt;

X = X0./(expmut.*vt);
S = Sf - (Sf - S0)./vt -  beta.*X.*(1 - expmut);
P = X.*((P0./X0 - csi).*expmuKt + csi);

% Pins S to 1.0e-5 g/l if S drops below zero.
ind = find(S<0);
one = ones(length(S(ind)),1);
%S(ind)=1.0e-5*one;
S(ind)=1.0e-5*exp(S(ind));

% Pins P to 0.0 mg/l if P drops below zero.
ind = find(P<0);
P(ind)=zeros(length(P(ind)),1);

ind=find(S==nan);
if length(ind)~=0,
        S(ind)=1.0e-5*ones(length(S(ind)),1);
        disp('WARNING: Bad S in f_out2.m');
else
%       plot((1:length(S)),S,'o',(1:length(X)),X,'+',(1:length(P)),P,'*')
end

ind = find(S==0);
one = ones(length(S(ind)),1);
S(ind)=eps*one;

%plot((1:length(mu)),mu,'o',(1:length(sigma)),sigma,'+',(1:length(qp)),qp,'*')


logS = log(S);
y = preprocess([logS X P],sclf(1,4:6),sclf(2,4:6));


function g=f_dout(k,z,xt,vex,params,sclf)
% function g=f_dout(k,z,xt,vex,params,sclf)
%
% Computes the gradients of the output model for the fed-batch
% penicillin fermentation, i.e. computes the partials
% of the outputs y w.r.t. the intermediates z.
% (Called by df_hyb.m.)
%
% k = index of the current data point.
% z = row vector of intermediates at the kth data point,
%     [muk sigmak qpk] [=] (1 x nz) = (1 x 3).
% xt = [S X P] [=] (ntrain x 3).* unscaled *
% vex= [F Sf dt] [=] (ntrain x 3).
% params = model parameters
% sclf = scale factors for inputs and outputs.
%
% g = an [nout x nz] = (3 x 3) vector of partials.

K = params(9);
D = vex(1);
```

```
Sf= vex(2);
dt= vex(3);

S0 = xt(k,1);
X0 = xt(k,2);
P0 = xt(k,3);

% Compute outputs.
y = postprocess(f_out(z,xt(k,:),vex(k,:),params,sclf),sclf(1,4:6),sclf(2,4:6));
y(1)=exp(y(1));
S = y(1);
X = y(2);
P = y(3);

dXdz = zeros(1, );
dXdz(1) = dt*X;

gamma = Sf - (Sf - S0)/(1 + D*dt);
beta  = z(2)/z(1);
alpha = 1 - exp(-z(1)*dt);

dSdz = zeros(1,3);
dSdz(1,1) = beta*X*(alpha/z(1) - dt);
dSdz(1,2) = -X*alpha/z(1);

dPdz = zeros(1,3);
csi = z(3)/(z(1) + K);
dcsidz1 = -csi/(z(1)+K);
dcsidz3 = 1/(z(1)+K);
expz1Kt = exp(-(z(1)+K)*dt);
dPdz(1) = dt*P + X*(-expz1Kt*(dcsidz1 + dt*(P0/X0-csi))+ dcsidz1);
dPdz(3) = X*dcsidz3*(1-expz1Kt);

g = [dSdz/sclf(2,4)/S; dXdz/sclf(2,5); dPdz/sclf(2,6)];


function [scaleddata,means,stds] = mcvscale(data,tolerance)
%       function [scaleddata,means,stds] = mcvscale(data,tolerance)
%
%       Mean-center and variance scale the data.  Return the scaleddata
%       and the means and standard deviations of the original data.
%       Each column of the  scaled data will have mean = 0 and variance = 1 (unless
%       all rows of the column have the same value, in which case the variance
%       of the column will be zero).
%
%       If the standard deviation of a column of the data
%       is less than the tolerance, that column is treated
%       as a column of the same value and its scaled equivalent is
%       a column of zeros.
%

[rows,columns] = size(data);
means = mean(data);
stds = std(data);
scaleddata = zeros(rows,columns);
v1 = ones(rows,1);
flg = (stds > tolerance);
scaleddata(:,flg) = (data(:,flg) - (v1*means(flg)))./(v1*stds(flg));


function xp = preprocess(x,xmeans,xstds)
%
% PREPROCESS -- function xp = preprocess(x,xmeans,xstds)
%
%   Preprocesses inputs to mnet.  Takes means and standard deviations
%   of the mean-centered variance-scaled training data and scales the new
%   data appropriately prior to introducing them to net.
%
[ndata,cols] = size(x);
v1 = ones(ndata,1);
xp = (x - v1*xmeans);
iflg = xstds > 0;
xp(:,iflg) = xp(:,iflg)./(v1*xstds(iflg));
```

```
function y = postprocess(yp,ymeans,ystds)
%
% POSTPROCESS -- function y = postprocess(yp,ymeans,ystds)
%
%   Postprocesses outputs from an mnet.  Takes means and standard deviations
%   of the mean-centered variance-scaled training data and scales the new
%   data appropriately prior to outputting them from net.
%
[ndata,cols] = size(yp);
v1 = ones(ndata,1);
y = yp.*(v1*ystds) + v1*ymeans;
```

C.1.1 Nonparametric Density Estimation

```
function m = cluster(x,h,iprint)
%CLUSTER
%m = cluster(x,h)
%
%   CLUSTER returns the matrix m[h,nin] of h cluster
%   centers for the data matrix x[ntrain,nin].  The k-means
%   clustering algorithm is used.
%
%   m=cluster(x,h,1) plots the movement of the cluster
%   centers if nin=2.
%

%   Copyright (c) 1992 by Process Science, Inc.
%   Mark A. Kramer 6-15-92.
[ntrain,nin] = size(x);
if(nargin==3)
   if(nin~=2)
      iprint=0;
   else
      plot(x(:,1),x(:,2),'+')
      hold on
   end
else
   iprint=0;
end
%
%  choose h random points for initial cluster centers
%
m = x(unique_r(h,1,ntrain),:);
if(iprint==1); plot(m(:,1),m(:,2),'g*'); end
%
%  main loop
%
  rel_change = 99;
  while(rel_change > 0.0001)
    dist = dist2(x,m);
    if(h == 1)
       b=ones(1,ntrain);
    else
       b = ~(dist - ones(h,1)*min(dist));
    end
    ncluster = sum(b');
    if(~any(ncluster == 0))
%
%  locate centroid of each cluster (b*x divided elementwise by ncluster)
%
      mnew=(b*x)./(ones(nin,1)*ncluster)';
      rel_change = sum(sum((m-mnew).^2))/sum(sum(m.^2));
         if(iprint==1); plot(mnew(:,1),mnew(:,2),'go'); end
      m = mnew;
    else
       break;
    end
  end
  if(any(ncluster == 0))
     disp('ncluster = 0...restarting clustering algorithm')
```

```
    hold off
    m = cluster(x,h);
  end
end
if(iprint==1); hold off; end


function shape = ep_heur(m,p,x)
%EP_HEUR
%shape = ep_heur(m,p,x)
%
%  Elliptical version of p-nearest neighbor heuristic.  Given hidden
%  unit locations m[h,nin], overlap parameter p, and data x[k,nin],
%  EP_HEUR establishes ellipses by local adaption to the covariance
%  structure of the x data, and sizes the ellipses so that there is
%  the desired overlap.  Output is returned as matrix shape[h*nin,nin]
%  which contains the inverse covariance matrices of the hidden units.
%

%       Copyright (c) 1992 by Process Science, Inc.
%       Mark A. Kramer 6-15-92.
[h,nin]  = size(m);
[nt,nin] = size(x);
sigma    = p_heur(m,p);
a        = layer1(m,sigma,x);
shape    = [];
for i=1:h
    wcov = wtcov(a(:,i),x);
%*** M.L.Thompson - 5/9/93 ***
    if rank(wcov)<nin,
        wcov = eye(size(wcov));
    end
    shape = [shape; inv(wcov)];
end
for i=1:h
    start   = (i-1)*nin+1;
    finish  = start+nin-1;
    shape_i = shape(start:finish,:);
    sqdist  = mahal2(ones(h,1)*m(i,:),m,shape_i);
%*** M.L.Thompson - 5/9/93 ***
    sigmasq = (sum(sqdist) - sqdist(i))/(h-1);
    shape(start:finish,:) = shape_i/sigmasq;
end


function sigma = p_heur(m,p)
%P_HEUR
%sigma = p_heur(m,p)
%
%  P_HEUR returns radial basis function widths sigma[h,1]
%  given m[h,nin] the cluster centers and the overlap parameter p.
%  If p=1 then sigma(i) will be the distance from center i to its
%  nearest neighbor.  If p>1 then sigma(i) is the root mean
%  square distance from center i to its p nearest neighbors.
%

%   Copyright (c) 1992 by Process Science, Inc.
%   Mark A. Kramer 6-15-92.
[h,nin] = size(m);
dist    = dist2(m,m);
dist    = sort(dist);
dist    = dist(2:p+1,1:h);
if p>1,
    sigma = (sqrt(mean(dist)))';
else
    sigma = (sqrt(dist))';
end
```

C.1.2 Parameter Estimation

```
function [w,cntrl] = train(w,func,dfunc,cntrl,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12)
%TRAIN
%[w,cntrl]=train(w0,func,dfunc,cntrl,a1,a2,...a12)
```

```
%
%  TRAIN contains the algorithms for training neural networks,
%  as invoked by make_bpn. The initial values of the weights are
%  given in w0[nvars].  func and dfunc are the functions supplying
%  the objective function and its derivatives, respectively, with
%  dfunc optional, but suggested.  cntrl[6] is a control vector
%  described in make_bpn.  a1 through a12 are user-defined arguments
%  passed directly to func and dfunc

%       Copyright (c) 1992 by Process Science, Inc.
%       Mark A. Kramer 6-15-92.
%       With revisions by M.L.Thompson 1-21-93
%*** MLT: Revisions:
%***      + Fixed bug that caused "nextw" not to be the weights corresponding
%***        to "f" (= "fmin") in all possible exit scenarios.
%***        (Previously, sometimes they didn't match after exiting if the
%***        "...insensitivity in numerical derivatives..." message had appeared.)
%***      + Removed manipulation of function evaluation counter upon recursion.
%***      + Minor efficiency enhancements (removal of unnecessary tests,
%***        assignments, etc.).
w = w(:);
%  TERMINATION TOLERANCES...user can adjust if necessary
gtol = 5.e-8;      % gradient criterion, default 5.e-4
ftol = 5.e-5;      % function progress criterion, default 5.e-5
steptol = 5.e-6;  % steplength criterion, default 5.e-6
%
nvars=length(w);
np = length(cntrl);
default = [1000 1 1 0 0 0];
if(nvars<100), default(3) = 3; end
cntrl(np+1:6)=default(np+1:6);
grad_reset = min(50,ceil(0.7*nvars));
if(cntrl(3) < 3), cg=1; else, h=eye(nvars); cg=0; end
%*** MLT: Eliminated "startiter" and manipulation of "cntrl(4)" (throughout).
%*** MLT: Added "paramstr".
paramstr='';
for i=1:nargin - 4, paramstr = [paramstr,',a',num2str(i)]; end
str=[func, '(w',paramstr, ')'];
if(nargin < 4) dfunc=[]; end
if(length(dfunc)>0)
    str2 = [dfunc, '(w',paramstr, ')'];
end
f = eval(str); func0 = f; df=zeros(nvars,1); nextw=w;
if(cntrl(2)>0),
    disp('')
    disp('GRADIENTS FUNCTIONS  OBJECTIVE    METHOD')
end
dwmin=1.e-4; dwmax=0.1; direc = nextw; sl = 0.01; msgcnt = 0;
count=0; method=0; loop=0; noprog=0; msg=[]; fold=0;
%*** MLT: Replaced "startiter+cntrl(4)" with simply "cntrl(4)".
if(cntrl(2)>0), disp([sprintf('%6.0f %8.0f %12.4g   ',max(0,cntrl(5)-1), cntrl(4),f)]),
end;
cntrl(2) = abs(cntrl(2));
while(loop~=2)
    w(:)=nextw;
    if(length(dfunc)>0)
        cntrl(5)=cntrl(5)+1;
        df = eval(str2);
    else
        prevf=f;
        dw = sign(direc+eps*ones(nvars,1)).*max(dwmin,min(dwmax, 0.01*abs(sl*direc)));
        for k=1:nvars,
        w(k) = w(k) + dw(k);
            f = eval(str);
            df(k) = (f-prevf)/dw(k); w(k) = w(k) - dw(k);
        end
        f = prevf; cntrl(4)=cntrl(4)+nvars;
    end
    if(loop==0)
        direc=-df; prevdls = df'*direc;
        sl = max(0.01, min([1,2*abs(f/prevdls)]));
        meth = 'steepest descent';
%*** MLT: Moved "loop=1" here, because unnecessary to assign if already equals 1.
        loop = 1;
```

```
    else
        count = count+1;
        if(cg & count>=grad_reset)
            count = 0; method = cntrl(3); cntrl(3)=0;
        end
        if(cntrl(3)==0)
            direc=-df; meth = 'steepest descent ';
        elseif(cntrl(3)==1)
            gg = prevdf'*prevdf; dgg = df'*df;
            gam = dgg/gg; direc = -df + gam*direc;
            meth = 'FR conj grad';
        elseif(cntrl(3)==2)
            gg = prevdf'*prevdf; dgg = (df - prevdf)'*df;
            gam = dgg/gg; direc = -df + gam*direc;
            meth = 'PR conj grad';
        elseif(cntrl(3)==3)
            delw = nextw-prevw; deldf = df-prevdf;
            fa = h*delw;
            h = h - fa*fa'/(delw'*fa) + deldf*deldf'/(deldf'*delw);
            direc=-h\df; meth = 'Broyden';
        elseif(cntrl(3)==4)
            delw = nextw-prevw; deldf = df-prevdf;
            fa = h*deldf;
            h = h + delw*delw'/(delw'*deldf) - fa*fa'/(deldf'*fa);
            direc=-h*df; meth = 'Davidon';
        end
    end
    if(method~=0), cntrl(3) = method; method = 0; end
%*** MLT: Moved "loop=1" from here to above, where "loop==0" tests TRUE.
    prevw=nextw; func0 = f;
    prevdf=df; prevdls=df'*direc;
    if(prevdls>=0)
        w(:)=prevw;
%*** MLT: Removed manipulation of func. evaluation counter.
        str3=['train(w,func,dfunc,cntrl',paramstr, ')'];
%*** MLT: Added "recursing" message.
        disp('...recursing...')
        [w,cntrl] = eval(str3);
        return; %*** EXITS FUNCTION ***
    end

%*** MLT: At this point, "nextw" equals "prevw".

%*** MLT: Changed "prevdls<gtol" to "prevdls>-gtol"
%***      because "prevdls" < 0. "prevdls" >=0 causes recursion above.
    if(max(abs(direc))<gtol & prevdls>-gtol)
        if(cntrl(2)>0),msg='Training terminated: gradient criterion'; end
        loop=2;
%*** MLT: Removed "nextw=prevw" because (after revisions) unnecessary.
    elseif(cntrl(4)>=cntrl(1))
        if(cntrl(2)>0),msg='Maximum number of function calls exceeded'; end
        loop=2;
%*** MLT: Removed "nextw=prevw" because (after revisions) unnecessary.
    else  % no convergence yet, do line search
%*** MLT: Replaced "elsif(loop==1)" with "else" because "loop" always equals 1 at this
point.
%*** MLT: Moved reassignment of "nextw" to here.
        nextw=nextw+sl*direc;
        wvec = [0 sl]; w(:)=nextw; fvec = [func0 eval(str)];
        cntrl(4)=cntrl(4)+1; go=2;
%*** MLT: Removed "factor=1" because "factor" is never used.
%*** MLT: Removed "i=2" because no longer necessary (see below).
        while(go > 0)
            slold = sl; sl = nextstep(prevdls,wvec,fvec);
%*** MLT: Replaced "fmin=..." with "[fmin,imin]=...".
            if(abs((slold-sl)/(sl+eps))<0.001),
                [fmin,imin]=min(fvec);
                break; %*** EXITS while-loop ***
            end
            wvec=[wvec sl]; if(length(wvec) > 5) go=0; end
            w(:) = prevw+sl*direc; fnew = eval(str); cntrl(4)=cntrl(4)+1;
%*** MLT: Removed "nextw=w(:)" because (after revisions) unnecessary.
            fvec=[fvec fnew];
%*** MLT: Replaced "i" with "isort".
```

```
            [wvec,isort]= sort(wvec); fvec = fvec(isort);
%*** MLT: Replaced "i" with "imin" (throughout).
            [fmin,imin]= min(fvec);
            if((imin~=1&imin~=length(wvec))), go=go-1; end
            if(go~= 0 & abs(sl) < steptol)
                go=0;loop=2;
                if(cntrl(2)>0),msg='Training terminated: steplength criteria'; end
            end
        end  % while (go > 0)

        sl=wvec(imin); f=fmin;
%*** MLT: Added following line to guarantee "nextw" corresponds to "f" (= "fmin")
%***      regardless of how the line search exited (because "imin" &
%***      "fmin" match, as well as "fvec" & "wvec," hence, "sl" & "f" match).
        nextw = prevw + sl*direc;

%*** MLT: At this point, "nextw" corresponds to "f."

        if(imin==1)
            sl=0.01;
            if(length(dfunc)==0 & cntrl(2)>0 & msgcnt==0)
%*** MLT: Changed "sensitivity" to "insensitivity." (I.e., obj. func. is
%***      insensitive to changes in weights.)
                disp('Warning: insensitivity in numerical derivatives detected.')
                disp('If training terminates shortly then restarting')
                disp('from termination point may reduce objective further.')
            end
            msgcnt=1;
        end
        if(abs((f-fold)/f)<ftol)
            prog=prog+1;
            if(prog>5)
                loop=2;msg='Training terminated: no more progress';
            end
        else, prog=0; end
        fold=f;
        if(cntrl(2)>0)
%*** MLT: Replaced "startiter+cntrl(4)" with simply "cntrl(4)."
            disp([sprintf('%6.0f %8.0f %12.4g   ',cntrl(5),cntrl(4),f) meth]);
        end
    end
end
w(:)=nextw;
%*** MLT: Removed manipulation of func. evaluation counter "cntrl(4)."
cntrl(6) = eval(str); disp(msg)
```

C.2. Prediction

```
function y_pred = run_hyb(x,m,shape,w,defmod,outmod,a1,a2,a3,a4,a5,a6)
%RUN_HYB
%y_pred = run_hyb(x,m,shape,w,'defmod','outmod',a1,a2,...,a6)
%
%  This function runs a hybrid network.  Inputs are the
%  cases to be run x[k,nin], the parameters defining
%  the hybrid network: m[h,nin], shape[h,1] or shape[h*nin,nin],
%  and weights w[h,nz], and the user functions defmod and
%  outmod (see manual for details).  The output are the
%  network predictions y_pred[k,nout].

%       Copyright (c) 1992 by Process Science, Inc.
%       Mark A. Kramer 6-15-92.
[k,nin] = size(x);
[h,nin] = size(m);
[hh,nz] = size(w);
ah = layer1(m,shape,x);
if(hh==h+1)
   ah = [ah,ones(k,1)];
elseif (hh~=h)
  error('incommensurate dimensions')
end
```

```
if(length(defmod)>0)
  str=[defmod,'(x'];
  for i=1:nargin - 6, str = [str,',a',num2str(i)]; end
  str = [str,')'];
  zdef = eval(str); [k,nz]=size(zdef);
else
  zdef = zeros(k,nz);
end
z = ah*w + zdef;
str=[outmod,'(z'];
for i=1:nargin - 6, str = [str,',a',num2str(i)]; end
str = [str,')'];
y_pred = eval(str);
```

C.3. Validation

C.3.1 Cross Validation

```
function [tse,press,sq_errs] = cv_rbf(x,y,h,p,s)
%CV_RBF
%[tse,press] = cv_rbf(x,y,h,p,s)
%[tse,press,sq_errs] = cv_rbf(x,y,h,p,s)
%
%   CV_RBF cross validates radial basis function networks.
%   Inputs are training inputs x[ntrain,nin], training targets
%   y[ntrain,nout], h[nh] vector of number of rbf units,
%   p[np] vector of overlap parameters, and s the number of
%   partitions in s-fold cross validation.
%
%   The outputs are the mean training squared errors tse[nh,np],
%   and the mean prediction squared errors press[nh,np] for each
%   feasible architecture.  The sq_errs[ntrain,nout] are the
%   squared error of each output for each training example
%   for the LAST combination of h and p only.
%
%   CV_RBF can be used with spherical or elliptical units,
%   using the global variable ELLIPSE (see make_rbf).
%

%       Copyright (c) 1992 by Process Science, Inc.
%       Mark A. Kramer 6-15-92.
[ntrain,nin] = size(x);
[ntrain,nout] = size(y);
np = length(p);
nh = length(h);
press = zeros(nh,np);
tse = zeros(nh,np);
for ip=1:np
for ih=1:nh
if(p(ip) < h(ih))
%
%  feasible architecture; S-fold cross validation procedure
%
disp(['h =',int2str(h(ih)),' p = ',int2str(p(ip))])
a = shuffle(ntrain);
sq_errs = zeros(ntrain,nout);
ntest = floor(ntrain/s)*ones(1,s);
for i=1:rem(ntrain,ntest)
  ntest(i) = ntest(i) + 1;
end
if(any(ntest > ntrain-h(ih)))
   error('h too big for training subset. Increase s or decrease h.');
end
for i=1:s
  string = ['  CV_iteration = ',int2str(i),' of ',int2str(s)];
  disp(string)
  if(i==1)
     start_test = 1;
     end_test = ntest(1);
  else
```

```
    start_test = start_test + ntest(i-1);
    end_test = end_test + ntest(i);
  end
  x_train = zeros(ntrain-ntest(i),nin);
  y_train = zeros(ntrain-ntest(i),nout);
  x_test = zeros(ntest(i),nin);
  y_test = zeros(ntest(i),nout);
%
%  Set up training and testing subsets
%
  itrain = 0;
  itest = 0;
  for j=1:ntrain
    if(j<start_test | j>end_test)
        itrain=itrain+1;
        x_train(itrain,:) = x(a(j),:);
        y_train(itrain,:) = y(a(j),:);
    else
        itest = itest+1;
        x_test(itest,:) = x(a(j),:);
        y_test(itest,:) = y(a(j),:);
    end
  end
  [m,sigma,w] = make_rbf(x_train,y_train,h(ih),p(ip));
  y_pred = run_rbf(x_test,m,sigma,w);
  for j=1:ntest(i)
    if(nout==1)
        sq_errs(a(start_test+j-1)) = (y_test(j)-y_pred(j))^2;
     else
        sq_errs(a(start_test+j-1),:) = (y_test(j,:)-y_pred(j,:)).^2;
     end
  end
  y_pred = run_rbf(x_train,m,sigma,w);
  tse(ih,ip) = tse(ih,ip) + mean(mean((y_pred-y_train).^2));
  clear x_train;
  clear y_train;
  clear x_test;
  clear y_test;
end
press(ih,ip) = mean(mean(sq_errs));
tse(ih,ip) = tse(ih,ip)/s;
end
end
end
end
end
end
```

C.3.2 Fast Cross Validation (FCV)

```
function   [w,yp,hkk,criteria,tol_factor] =
fcv_ls(x,y,fmodel,fgrad,frechetc,w0,printout,lv1,p1,p2,p3);
% function [w,yp,hkk,criteria,tol_factor] =
fcv_ls(x,y,fmodel,fgrad,frechetc,w0,printout,lv1,p1,p2,p3);
%
% Estimate the optimal parameters of a nonlinear model using data x & y.
% Return the parameters, the diagonal of the effective hat matrix, and
% a criteria vector.  The criteria vector has the following contents:
% criteria(1) = peff;
% criteria(2) = FCV;
% criteria(3) = GCV;
% criteria(4) = Moody's GPE;
% criteria(5) = Larsen's generalized FPE;
% criteria(6) = Linear GCV;
%
% Uses unconstrained least-squares objective function: F(w)=(1/2/N)*|c|^2,
% where w  = parameters, N=no. of data points, c = residual = y - yp,
%       yp = predictions, and y=data values for dependent variable.
% Uses Newton's method (i.e. requires inverse Hessian).
% Uses Numerical Recipes in C routines (of course, rewritten
% in Matlab language) to perform the line minimization.
% These routines include: linmin, mnbrakt and brent.
%
```

```
% Model is function fmodel; gradient w.r.t. parameters is function fgrad; and
% function frechetc is the Frechet derivative of dfdw times c.
% [yp,q1,q2,q3] = feval(   fmodel,x,w,p1,p2,p3);
% dfdw          = feval(    fgrad,x,w,yp,p1,p2,p3,q1,q2,q3);
% frchtc        = feval( frechetc,x,w,yp,dfdw,p1,p2,p3,q1,q2,q3).
% Optional parameters p1,p2, and p3 are used in fmodel and fgrad.
% Returned values q1, q2, and q3 are mandatory for fmodel.
%
% Initial parameters (w0) must be supplied; but print flag (printout) is optional.
%
% MLT: 11/94

fevlsmax = 1700;
thresh = 3e-7;
FTOL   = (5e-4)*10;
tol_factor = 100;

[n,nx] = size(x);
[n,ny] = size(y);
if ny~=1,
    error('y-data must be a column vector');
end

if nargin<7,
    printout = 1;
end

stop_time = 0;
fevls     = 1;
w         = w0;

% Compute objective function.
[Fobj,yp,q1,q2,q3] = objf(w,x,y,fmodel,p1,p2,p3,tol_factor);
h = p1;
cntrl = [1000 1];
%[w,cntrl] = trainmt2(w,'objf',[],cntrl,x,y,fmodel,fgrad,p1,p2,p3,tol_factor);
[w_all,cntrl]  = make_bpn(x,y,[nx h 1],[0 1 0],[w;q2],cntrl);
w = w_all(1:(h*(nx+1)));
% Compute objective function.
[Fobj,yp,q1,q2,q3] = objf(w,x,y,fmodel,p1,p2,p3,tol_factor);

% Compute the gradient of the objective function.
dfdw      = feval(fgrad,x,w,y,yp,p1,p2,p3,tol_factor,q1,q2,q3);
gradient  = -dfdw'*(y-yp)/n;
normgrad  = norm(gradient,2);

if printout,
    fprintf(1,  '\n%8s%16s%16s%16s\n','fevals','(1/2/N)*|c|^2','|grad|','|dw|');
    fprintf(1,    '%8s%16s%16s%16s\n','======','============','======','====');
    fprintf(1,'%8d%16.4e%16.4e%16s\n',  fevls , Fobj          ,normgrad,'----');
end

% Check stopping criteria.
gradflat     = normgrad<thresh;
stop_time    = gradflat;

% *** NEWTON'S METHOD ***
dfn   = length(w);
fnext = 2*dfn;
do_numer = 0;
while ~stop_time,
    % Save previous parameters and obj. func.
    prevw = w;
    Fobj0 = Fobj;

    % Compute the inverse Hessian.
    if ~do_numer,
        if 1,
            hess   = (1/n)*dfdw'*dfdw;
        else
            frchtc = feval(frechetc,x,w,y,yp,dfdw,p1,p2,p3,q1,q2,q3); % (Frechet deriv. of
dfdw')*c
            hess   = (1/n)*(dfdw'*dfdw - frchtc);
            hess   = (hess + hess')/2;
```

```
        end
    else
        fnext  = fevls + dfn;
        hess   = numrhess(w,x,y,fmodel,fgrad,p1,p2,p3,tol_factor);
    end
    ihess  = myinv(hess,max(size(hess))*norm(hess)*eps*tol_factor);

    % Compute direction of line search.
    tau = -ihess*gradient;

    % Perform line search.
    [w,tau,Fobj,ilin,yp,q1,q2,q3] =
linmin(prevw,tau,'f1d',p1,p2,p3,fmodel,x,y,prevw,tau,tol_factor);
    normtau   = norm(tau,2);

    % Compute objective function.
    [Fobj,yp,q1,q2,q3] = objf(w,x,y,fmodel,p1,p2,p3,tol_factor);
    fevls     = fevls + ilin + 1;

    % Compute the gradient of the objective function.
    dfdw      = feval(fgrad,x,w,y,yp,p1,p2,p3,tol_factor,q1,q2,q3);
    gradient  = -dfdw'*(y-yp)/n;
    normgrad  = norm(gradient,2);

    if printout,
        fprintf(1,'%8d%16.4e%16.4e%16.4e\n',fevls,Fobj,normgrad,normtau);
    end

    % Check stopping criteria.
    toomanyfevls = fevls>fevlsmax;
    fobjflat     = abs(Fobj0-Fobj)<FTOL*abs(Fobj);
    gradflat     = normgrad<thresh;
    stop_time    = fobjflat | gradflat | toomanyfevls;

    if stop_time & ~do_numer,
        stop_time = 0;
        do_numer   = 1;
    end

if 0,
    if stop_time & ~toomanyfevls & tol_factor>1,
        tol_factor = tol_factor/10;
        if printout,
            fprintf(1,'\ntol_factor=%e; rank(hess)=%d\n',tol_factor,rank(hess));
        end
        [Fobj,yp,q1,q2,q3] = objf(w,x,y,fmodel,p1,p2,p3,tol_factor);
      if 0,
        while (Fobj>=Fobj0) & tol_factor>1,
            tol_factor = tol_factor/10;
            [Fobj,yp,q1,q2,q3] = objf(w,x,y,fmodel,p1,p2,p3,tol_factor);
        end
      end
        stop_time = Fobj>=Fobj0;
        if stop_time,
            tol_factor = tol_factor*10;
            [Fobj,yp,q1,q2,q3] = objf(w,x,y,fmodel,p1,p2,p3,tol_factor);
        end
    end
end
end

if printout,
    if fevls>fevlsmax,
        fprintf(1,'\n*** Exceeded max evaluations (%d) *** Fobj =
%12.4e\n',fevlsmax,Fobj);
    else
        fprintf(1,'\n*** Convergence achieved *** Fobj = %12.4e\n',Fobj);
    end
end

if ~lv1, % If not leave-one-out training return hkk and criteria.

  % Compute the hat matrix and take its diagonal.
  if isempty(ihess) | cond(hess)>1e4,
```

```
        if 0,
            dwdy  = pinv(dfdw);
        else
            hess   = numrhess(w,x,y,fmodel,fgrad,p1,p2,p3,tol_factor);
            ihess  = myinv(hess,max(size(hess))*norm(hess)*eps*tol_factor);
            dwdy   = ihess*(dfdw'/n);
        end
    else
      dwdy  = ihess*(dfdw'/n);
    end
    hkk   = diag(dfdw*dwdy);

    % Check if assumptions are valid:
    % (1) First-order perturbation approx. is good.
    % Leave-one-out residual from 1st-order pert. model.
    delyp = (y - yp)./(1 - hkk);
    delf  = zeros(n,1);
  if 0,
    for k=1:n,
      % Leave-one-out residual from full model with 1st-order pert. params.
      yplvk   = feval(fmodel,x,w-dwdy(:,k)*(delyp(k)),y,p1,p2,p3,tol_factor);
      delf(k) = yplvk(k) - yp(k);
    end
  %  badapproxs = find((delyp-delf).^2>10*thresh);
  end
    badapproxs = [];
    if length(badapproxs),
      disp('Bad approximations found:')
      [badapproxs hkk(badapproxs)]
      pause
    end

    % (2) hkk >= 0.
    neghs = find(hkk<0);
    if length(neghs),
      disp('Negative hkk found:')
      [neghs hkk(neghs)]
      pause
    end

    % (3) hkk <= 1.
    bighs = find(hkk>1);
    if length(bighs),
      disp('Big hkk found:')
      [bighs hkk(bighs)]
      pause
    end

    exceptions = sort([neghs; bighs; badapproxs]);

  %    if n<50,
  %        exceptions = (1:n)';
  %    end

    % Do leave-one-out training for exceptions.
    for j=1:length(exceptions),
      k = exceptions(j);
      doit = j==1;
      if ~doit,
          doit = k>exceptions(j-1); % skip duplicates
      end
      if doit,
          indx       = [(1:(k-1))';((k+1):n)'];
          xlv        = x(indx,:);
          ylv        = y(indx,:);
          fprintf(1,'\nleave-%d-out: y=%e c=%e hkk=%e',k,y(k),y(k)-yp(k),hkk(k));
          [wk,yplv1] = fcv_ls(xlv,ylv,fmodel,fgrad,frechetc,w,0,1,p1,p2,p3);
          [yplv1,q1,q2,q3] = feval(fmodel,xlv,wk,ylv,p1,p2,p3,tol_factor);
          yplvk      = feval(fmodel,x(k,:),wk,y,p1,p2,p3,tol_factor,q1,q2,q3);
          hkk(k)     =  1 - (y(k)-yp(k))/(y(k)-yplvk);
          fprintf(1,' hkk=%e',hkk(k));
      end
    end
    fprintf(1,'\n');
```

```
  if 0,
    subplot(1,2,1);  %***** THIS MAY BE SPECIFIC TO VERSIONS 4.x!!! *****
    hist(hkk,max(20,min(n/2,20)));
    title('Histogram of hkk');
    subplot(1,2,2);
    plot(hkk,1:length(hkk),'*');
    title('k vs. hkk');
    grid on;
    disp('Press any key to continue...')
    pause
  end

  % Compute effective number of parameters and selection criteria.
  peff = sum(hkk);
  havg = peff/n
  c    = y - yp;
  mc2  = mean(c.^2);

  criteria(1) = peff;
  criteria(2) = mean((c./(1-hkk)).^2); % FCV;
  criteria(3) = mc2*(1/(1-havg))^2;    % GCV;
  criteria(4) = mc2*(1+havg)/(1-havg); % Moody's GPE;
%  criteria(5) = % Larsen's generalized FPE;
%  criteria(6) = % Linear GCV;

else % If leave-one-out training just return w.

end


function   [w,wlin,yp,hkk,criteria] = bpnfcv(x,y,h,w0,printout);
% function [w,wlin,yp,hkk,criteria] = bpnfcv(x,y,h,w0,printout);

[n,nx] = size(x);
if nargin<4,
    w0=[];
end
if isempty(w0),
    w0 = randn((nx+1)*h,1);
end
if nargin<5,
    printout = 1;
end
[w,yp,hkk,criteria]   =
fcv_ls(x,y,'fsepnl','dfsepnl','frcsepnl',w0,printout,0,h,'f3bpn','df3bpn2');
[yp,bases,wlin,ranka] = fsepnl(x,w,y,h,'f3bpn',[],100);


function   [yp,bases,wlin,ranka] = fsepnl(x,w,y,h,fbases,p3,tol_factor,q1,wlin0,q3);
% function [yp,bases,wlin,ranka] = fsepnl(x,w,y,h,fbases,p3,tol_factor,q1,wlin0,q3);
% Compute model predictions as separable nonlinear function.
% Bases functions computed by calling function fbases:
% bases = feval(fbases,x,w,h);
% where h = number of non-constant bases.
% M.L. Thompson 11/94

[n,nx]   = size(x);
bases    = feval(fbases,x,w,h);
[n,nbas] = size(bases);
ranka    = rank(bases);

nowlin = nargin<9;
if ~nowlin,
    nowlin = isempty(wlin0);
end

if nowlin,

if 1,

% wlin  = bases\y;
wlin  = pinv(bases max(size(bases))*norm(bases)*eps*tol_factor)*y;
%wlin = myinv(bases'*bases)*bases'*y;
```

```
else

% To avoid problems in regression, use SVD with removal of small
% singular values.  This slows down the estimation of the linear
% parameters, but it is essential for reliable results.
[u,s,v] = svd(bases,0);
s       = diag(s);

if n>=nbas,
    gpad   = [];
    thresh = 1.0e-4*max(s);
else
    gpad   = zeros(nbas-n,1);
    thresh = max(size(bases)) * norm(bases) * eps;
end

good  = s > thresh;
ranka = sum(good);
u     = u(:,good);
v     = v(:,[good; gpad]);
snorm = norm(s,2);
s     = s(good);
vs    = v*diag(s)/snorm;

% Perform regression.
wlin  = v*diag(1.0./s)*(u'*y);

end
else
wlin = wlin0;
end
% Compute predictions.
yp = bases*wlin;


function   dfdw = dfsepnl(x,w,y,yp,h,p2,dfbases,tol_factor,bases,wlin,q3);
% function dfdw = dfsepnl(x,w,y,yp,h,p2,dfbases,tol_factor,bases,wlin,q3);
% Compute the gradient of a separable nonlinear model.
% M.L. Thompson 11/94

if 0,
 [n,nx]   = size(x);
 bias     = ones(n,1);
 [n,nbas] = size(bases);
 c        = y - bases*wlin;
end

dfdw     = feval(dfbases,x,w,y,bases,wlin,tol_factor);

function     activations = f3bpn(x,w,h)
% function   activations = f3bpn(x,w,h)
% Compute node activations of a 3-layer single-output BPN.
% where h = number of nodes and w = input weights.
% M.L. Thompson 11/94

[n,nx]      = size(x);
bias        = ones(n,1);
win         = reshape(w,nx+1,h);
activations = [sigmoid0([x bias]*win) bias];
function   dbases = df3bpn2(x,w,y,bases,wlin,tol_factor);
% function dbases = df3bpn2(x,w,y,bases,wlin,tol_factor);
% Compute the gradient of the node activations of a 3-layer BPN.
% x     = inputs
% w     = input weights
% bases = node activations
% wlin  = output weights.
% M.L. Thompson 11/94

[n,nx]   = size(x);
[n,nbas] = size(bases);
h        = nbas - 1;
a        = bases(:,1:h);
bias     = ones(n,1);
```

```
c        = y - bases*wlin;
Api      = pinv(bases,max(size(bases))*norm(bases)*eps*tol_factor);

dbases = kron(0.5*(1-a.*a).*(bias*(wlin(1:h)')),ones(1,nx+1)).*kron(ones(1,h),[x bias]);
b      = zeros(nbas,h*(nx+1));
indx   = kron((1:h),ones(1,nx+1))' + ((1:(h*(nx+1)))'-1)*nbas;
b(indx)= (kron(0.5*(1-a.*a),ones(1,nx+1)).*kron(ones(1,h),[x bias]))'*c;
%b
Dmat   = Api'* b;
b(indx)= (kron(0.5*(1-a.*a),ones(1,nx+1)).*kron(ones(1,h),[x bias]))'*(Api'*wlin);
%b
AG     = bases*b;
Cmat   = dbases + AG;
dbases = Cmat + bases*(Api*(Dmat-Cmat));
```

Appendix D

Probabilistic Modeling Toolkit for Matlab

This is the Matlab, vers. 4.0, source code used to implement the probabilistic modeling methodology.

D.1. Synthesis

D.1.1 Nonparametric Density Estimation

```
function  [x,mu,shape,p,z,py_mnp,zeta1,missing,xpri] =
emcond3(x,Klag,Neach,h,fltsig,fltpri,bounds,z,mu,shape,p);
%function [x,mu,shape,p,z,py_mnp,zeta1,missing,xpri] =
emcond3(x,Klag,Neach,h,fltsig,fltpri,bounds,z,mu,shape,p);
% EMCOND3 -- Robust Nonparametric Density Estimation.
%   Estimates the (hyper)parameters mu, shape, p for a finite mixture of normals:
%   i.e. the joint pdf p(x(t),x(t-1),x(t-2),...,x(t-Klag)|mu,shape,p,H).
%   Handles case of 2-component fault model with no bias.
%   Does EM with windowed data to lag Klag and estimation using conditionals.
%   Calls routine emrect.m to perform the M-Steps.
%
%   Handles multiple runs each with Neach number of observations.
%   The input parameters are the training data x[ndata,nx], where nx=(Klag+1)*nin,
%   nin is the number of variables in each window.
%   h the number of normal distributions, and optional initial
%   guesses for mu, shape, and p.
%
%   The outputs are mu[h,nin] the means of the h normals,
%   the inverses of the distribution covarsiance matrices shape[h*nin,nin],
%   and p[h+1,nout] the prior probabilities (weights) for each distribution.
%
%   bounds is a [2,nin] matrix.  First row is lower bound. Second is upper bound.
%   If a variable of a point violates the bounds, then set the variable equal
%    to the bound.
%   fltsig is [nin,2]. Its 1st column is the std. devs. of the noise; 2nd col. is std devs
of
%   gross error distn.  1-fltpri is the prior probability of a gross error.

nsets   = Klag + 1;
[ndata,nx] = size(x);
nin     = nx/nsets;

onecol  = ones(ndata,1);
onerow  = ones(1,nx);
if nargin<10,
    shape   = zeros(h*nx,nx);
end
nfaults = length(fltsig);

bounds = [bounds bounds];

toolow  = x<onecol*bounds(1,:);
toohigh = x>onecol*bounds(2,:);
for k=1:nx,
    if sum(toolow(:,k)),
        x(toolow(:,k),k) = ones(sum(toolow(:,k)),1)*bounds(1,k);
    end
```

```
    if sum(toohigh(:,k)),
        x(toohigh(:,k),k) = ones(sum(toohigh(:,k)),1)*bounds(2,k);
    end
end

missing = isnan(x);
clear toolow toohigh

iscomp  = ~sum(missing');
imiss   = find(~iscomp);
xcomplete = x(iscomp,:);
ncomp   = sum(iscomp);

if nargin<8,
    %
    %  Initialize using k-means clustering.
    done=0;
    while ~done,
     [mu,z] = clusterz(xcomplete,h);
     toofew = find(sum(z')<2);
     h      = h - length(toofew)
     done   = length(toofew)==0;
    end
    z = z';
    p = mean(z);
    last = 0;
    for j=1:h,
        first = last+1;
        last  = last+nx;
        dxm     = xcomplete - ones(ncomp,1)*mu(j,:);
        ni      = ncomp*p(j);
        s       = dxm'*((z(:,j)*onerow).*dxm)/ni;
        shape(first:last,:) = pinv(s);
        covars(first:last,:) = s;
    end
elseif nargin<9,
        % Initialize based on given membership z.
        p = mean(z);
        last = 0;
        for j=1:h,
            first   = last+1;
            last    = last+nx;
            mu(j,:) = sum((z(iscomp,j)*onerow).*xcomplete)/(ncomp*p(j));
            dxm     = xcomplete - ones(ncomp,1)*mu(j,:);
            ni      = ncomp*p(j);
            s       = dxm'*((z(iscomp,j)*onerow).*dxm)/ni;
            shape(first:last,:) = pinv(s);
            covars(first:last,:) = s;
        end
    else
        fl = 1:nx;
        for j=1:h,
            if rank(shape(fl,:))==nx,
                covars(fl,:) = inv(shape(fl,:));
            else
                covars(fl,:) = pinv(shape(fl,:));
            end
            fl = fl + nx;
        end
        % Initialize based on given mu,shape,p.
        [z,py_mnp] = e_z_all(x,mu,shape,p);
        p = mean(z);
    end
end

xrect = x;

zmiss = zeros(ndata-ncomp,h);
if ndata>ncomp,
    % Estimate missing values
    for i=1:(ndata-ncomp),
        jclosest = (ones(h,1)*xrect(imiss(i),:)-mu).^2;
        [mindist,jclosest] = min(sum(jclosest(:,missing(imiss(i),:))'));
        kmiss    = missing(imiss(i),:);
```

```
        xrect(imiss(i),kmiss) = mu(jclosest,kmiss);
        zmiss(i,jclosest) = 1;
    end
end

zcomp=z;
z = zeros(ndata,h);
z(iscomp,:) = zcomp;
z(imiss,:) = zmiss;
p = mean(z);

% Eliminate any component with a zero prior probability.
notzros = find(p>(1/ndata/100));
if length(notzros)<length(p),
    for i=1:length(p),
        if ~length(find(notzros)==i),
            disp(['Deleting #',int2str(i),' because p(i) == 0.']);
        end
    end
    p   = p(notzros);
    mu  = mu(notzros,:);
    z   = z(:,notzros);
end
p = mean(z);

h = length(p);
hh = h;
muold = mu;

thresh = (5.0e-7)/100;
itermax = 50;
converged = 0;
iter = 1;

xdata = xrect; % Replaces missing values in the data matrix.
x = xrect;

fxlast = normmixp(mu,shape,p,x);
zro=find(~fxlast);
if ~isempty(zro),
    fxlast(zro)=ones(length(zro),1);
end
fobj=0;
fobjold = 0;
fobj = sum(log(fxlast));
%keyboard
    fobjold= fobj;
    muold  = mu;
    shold  = shape;
    pold   = p;
    zold   = z;

oncemore=1;
twice = 0;
rnk = zeros(h,1);
dets= zeros(h,1);
while ~converged & iter<itermax,
%    disp(['*** ITERATION: ',int2str(iter),' *** ',num2str(sum(fxlast.^2)),' ***']);
    last = 0;
    i    = 1;
    while i<=h,
        first   = last + 1;
        last    = last + nx;

      if p(i)<(2*nx+1)/ndata,
            % Delete components with p too small
            disp(['Deleting #',int2str(i),': ',int2str(h-1),' nodes left.']);
            disp([' because p(',int2str(i),') ==' num2str(p(i)) '.']);
            p     = [ p(1:(i-1))  p((i+1):h)];
            p     = p/sum(p);
            mu    = [mu(1:(i-1),:); mu((i+1):h,:)];
            shape = [shape(1:((i-1)*nx),:); shape((i*nx+1):(h*nx),:)];
            covars = [covars(1:((i-1)*nx),:); covars((i*nx+1):(h*nx),:)];
            rnk   = [rnk(1:(i-1)); rnk((i+1):h)];
```

```
            dets  = [dets(1:(i-1)); dets((i+1):h)];
            last  = last - nx;
            i=1;
            last = 0;
            h=h-1;
            [z,py_mnp] = e_z_all(x,mu,shape,p);
            p  = mean(z);
            h  = length(p);
        else

          mu(i,:) = sum((z(:,i)*onerow).*x)/(ndata*p(i));
          dxm     = x - onecol*mu(i,:);
          ni      = ndata*p(i);
          s       = dxm'*((z(:,i)*onerow).*dxm)/ni;
          shape(first:last,:) = s; % Maximum likelihood estimate.
          covars(first:last,:) = s;
          rnk(i)  = rank(shape(first:last,:));
          dets(i) = det(shape(first:last,:));

          if dets(i)<eps,
              %disp(['Rank of ',int2str(i),' is ',int2str(rnk(i))]);
          end
          if rnk(i),
            if rnk(i)<nx,
%              disp(['Component #',int2str(i),' is rank deficient: ',int2str(rnk)]);
               s = pinv(shape(first:last,:));
            else
              % Simply invert if full rank.
              s = inv(shape(first:last,:));
            end
            shape(first:last,:) = s;
          else
              % Delete components with rank == 0
              disp(['Deleting #',int2str(i),': ',int2str(h-1),' nodes left.']);
              disp([' because rank == 0. Also: p(',int2str(i),') ==' num2str(p(i)) '.']);
              p     = [ p(1:(i-1))  p((i+1):h)];
              p     = p/sum(p)
              mu    = [mu(1:(i-1),:); mu((i+1):h,:)];
              shape = [shape(1:((i-1)*nx),:); shape((i*nx+1):(h*nx),:)];
              covars = [covars(1:((i-1)*nx),:); covars((i*nx+1):(h*nx),:)];
              rnk   = [rnk(1:(i-1)); rnk((i+1):h)];
              dets  = [dets(1:(i-1)); dets((i+1):h)];
              last  = 0;
              i=1;
              h=h-1;
              [z,py_mnp] = e_z_all(x,mu,shape,p);
          end
        end
        i=i+1;
    end
disp('Ready to check p'); pause(1);

    [z,py_mnp] = e_z_all(x,mu,shape,p);
    p  = mean(z);

    last = 0;
    i = 1;
    while i<=h,
        first = last + 1;
        last  = last + nx;

        delete = ~(p(i)*ndata>(2*nx+1));
        %delete = ~(p(i)*ndata>nx);

        if delete,
            % Delete components with p too small.
            disp(['Deleting #',int2str(i),': ',int2str(h-1),' nodes left.']);
            disp([' because p(',int2str(i),') ==' num2str(p(i)) '.']);
            p     = [ p(1:(i-1))  p((i+1):h)];
            p     = p/sum(p)
            mu    = [mu(1:(i-1),:); mu((i+1):h,:)];
            shape = [shape(1:((i-1)*nx),:); shape((i*nx+1):(h*nx),:)];
            covars = [covars(1:((i-1)*nx),:); covars((i*nx+1):(h*nx),:)];
            rnk   = [ rnk(1:(i-1));  rnk((i+1):h)];
```

```
            dets  = [dets(1:(i-1)); dets((i+1):h)];
            last  = 0;
            i = 1;
            h = length(p);
            z = e_z_all(x,mu,shape,p);
            p     = mean(z);
            break % Leave inner while loop.  Reallocate points
        else
            i = i + 1;
        end
    end % while

    disp('Ready to rectify'); pause(1);
    if ~delete & hh==length(p),
        % Rectify
        disp('rectifying');pause(1)
        [x,zeta1,xpri] =
emrect(x,xdata,fltsig,fltpri,bounds,h,nx,nin,ndata,ncomp,Neach,mu,shape,p,covars,z,missing
);
        disp('Done rectifying'); pause(1);
        [z,py_mnp] = e_z_all(x,mu,shape,p);
        p  = mean(z);

        converged = (sum(sum((muold-mu).^2)) < thresh*sum(sum(mu.^2))) & iter>3;
        fx        = normmixp(mu,shape,p,x);
        fobj = sum(log(fx+eps));
        if fobj<=fobjold & iter>3,
            mu = muold;
            shape = shold;
            p     = pold;
            z     = zold;
            converged = 1;
        end
        if ~twice & ~converged & fx<fxlast,
            z=z>0.5; % Reset with all-or-nothing.
            twice = 1;
        else if twice,
                converged=1;
             else
                twice = 0;
             end
        end
    end
    fxlast = fx;
    hh     = length(p);
    fobjold= fobj;
    muold  = mu;
    shold  = shape;
    pold   = p;
    zold   = z;
    iter   = iter + 1;
    if iter==itermax & oncemore,
        oncemore=0;
        z=z>0.5;     % All-or-nothing membership.
        iter=iter-1; % Do one more iteration.
    end
end % while

%disp(['*** ITERATION: ',int2str(iter),' *** ',num2str(sum(fxlast.^2)),' ***']);
disp(['*** ITERATION: ',int2str(iter),' *** ',num2str(fobj),' ***']);

function [fx,g,z]=normmixp(mu,shape,p,x);
%function [fx,g,z]=normmixp(mu,shape,p,x);
%   Computes the normal mixture model pdf.
%   The inputs are the values from mlerbf.m: distribution means
%   m[h,nin] and inverse covariances shape[h*nin,nin],
%   the priors p[1,h], and the values at which the pdf
%   should be evaluated x[ndata,nin].
%   g[ndata,h] is the columns of contributions from each component.
%   Also, computes z[ndata,h], the probability of each component given x.

[h,nin] = size(mu);
[ndata,dum] = size(x);
onecol  = ones(ndata,1);
```

```
onerow  = ones(1,h);

p    = p(:)';
rnk  = zeros(size(p));
dets = zeros(size(p));

fl = 1:nin;
for j=1:h,
    dxm     = x - onecol*mu(j,:);
    s       = shape(fl+nin*(j-1),:);
    rnk(j)  = rank(s);
    dets(j) = mydet(s,rnk(j));

    % Compute the exponents of the exponentials for each point.
    if nin==1,
        g(:,j) = dxm*s.*dxm;
    else
        g(:,j) = sum((dxm*s.*dxm)')';
    end
end

% Found to be better numerically than direct calculation.
% Essentially, divides each component by the largest exponential term of all
% the h components.

clear dxm fl
if h>1,
    gm = min(g')';
    g  = exp(-0.5*(g - gm*onerow)).*(onecol*(p.*(dets.^0.5).*((2*pi).^(-rnk/2))));
    fx = sum(g')';
    z  = g./(fx*onerow);
    gm = exp(-0.5*gm);
    g  = g.*(gm*onerow);
    fx = fx.*gm; % This sets fx equal to the actual p(x|H).
else
    z  = onecol;
    g  = exp(-0.5*g).*(onecol*((dets^0.5)*((2*pi)^(-rnk/2))));
    fx = g;
end


function [z,fx]=e_z_all(x,mu,shape,p);
%function [z,fx]=e_z_all(x,mu,shape,p);
% Compute z, the probability of each component given x.

[h,nin] = size(mu);
[ndata,dum] = size(x);
onecol  = ones(ndata,1);
onerow  = ones(1,h);

p    = p(:)';
rnk  = zeros(size(p));
dets = zeros(size(p));

fl = 1:nin;
for j=1:h,
    dxm     = x - onecol*mu(j,:);
    s       = shape(fl+nin*(j-1),:);
    rnk(j)  = rank(s);
    dets(j) = mydet(s,rnk(j));
    if rnk(j)<nin,
        s = pinv(pinv(s));
    end

    % Compute the exponents of the exponentials for each point.
    if nin==1,
        g(:,j) = dxm*s.*dxm;
    else
        g(:,j) = sum((dxm*s.*dxm)')';
    end
end

% Found to be better numerically than direct calculation.
% Essentially, divides each component by the largest exponential
```

```
% term of all the h components.
[gm,im] = min(g');
if 0,
 gm = gm';
 % Fractional membership.
 g  = g - gm*onerow;
 clear dxm fl
 g  = exp(-0.5*g).*(onecol*(p.*(dets.^0.5).*((2*pi).^(-rnk/2))));
 fx = sum(g')';
 z  = g./(fx*onerow);
 fx = fx.*exp(-0.5*gm); % This sets fx equal to the actual p(x|H).
else
 % Winner-take-all membership.
 if h==1,
    z  = onecol;
    gm = g;
 else
    gm = gm';
    z       = onecol*(1:h)==im'*onerow;
 end
 prefact = onecol*(p.*(dets.^0.5).*((2*pi).^(-rnk/2)));
 prefact = prefact(z(:));
 fx      = exp(-0.5*gm).*prefact;
end
%keyboard


function
[x,zeta1,xpri,invR,invS,Wdat]=emrect(x,xdata,fltsig,fltpri,bounds,h,nx,nin,ndata,ncomp,Nea
ch,mu,shape,p,covars,z,missing);
%function
[x,zeta1,xpri,invR,invS,Wdat]=emrect(x,xdata,fltsig,fltpri,bounds,h,nx,nin,ndata,ncomp,Nea
ch,mu,shape,p,covars,z,missing);
% EMRECT = Performs the M-Step of a rectification scheme involving corrupted data and a
%          finite mixture of normals as the joint pdf
%          p(x(t),x(t-1),x(t-2),...,x(t-Klag)|mu,shape,p,H), which is denoted as
p(X(t:Klag)|H).
%          Calculates the conditional pdf p(x(t)|x(t-1),x(t-2),...,x(t-
Klag),mu,shape,p,H);
%          multiplies by the 2-component finite mixture of normals that serves as the
error pdf;
%          and uses the resultant mode (=mean) as the best estimate of the true values.
%          This estimate is a weighted average of xdata and xpri=f(x(t-1),x(t-2),...,x(t-
Klag)),
%          described below.
% INPUTS:
% x       = initial estimate of true values, [ndata,nx]: the last nin columns are x(t);
%           the first nin columns are x(t-Klag).
% xdata   = windowed measurements [ndata,nx]
% fltsig  = std. devs. of meas. fault components 1 and 2 [nx,2]
% fltpri  = prior probability of being from first fault component [nx,1]
% bounds  = upper and lower bounds of true x (Inf is valid) [nx,2]
% h       = number of component in joint pdf p(X(t:Klag)|H) mixture
% nx      = number of total variables = nin*(Klag+1), where Klag=max lag correlated
w/current obs.
% nin     = number of variables in each original observation
% ndata   = number of windowed observations
% ncomp   = number of complete observations (i.e. obs. w/out missing values in any var.)
% Neach   = number of time series (or runs); if obs.are indep. then Neach=ndata and
nin=nx.
% mu      = component means of joint pdf p(X(t:Klag)|H) [h,nx]
% shape   = inverse covars. of joint pdf p(X(t:Klag)|H) [h*nx,nx]*** ACTUALLY, NOT USED!
% p       = prior prob. of comps for jnt.pdf p(X(t:Klag)|H) [1,h]
% covars  = covars of joint pdf p(X(t:Klag)|H) [h*nx,nx] *** THIS IS USED INSTEAD OF
SHAPE!
%         (i.e. shape holds either invs. or gen. invs of covars)
% z       = component membership probabilities [ndata,h]
% missing = flags of values that are missing [ndata,nx]
% OUTPUTS:
% x    = final estimate of true values (i.e. rectified values) [ndata,nin],
% zeta1= posterior probability of being from first fault component distribution (usually
noise),[ndata,1]
% xpri = estimate based on prior knowledge alone (x is a wtd. avg. of xpri and xdata).
```

```
onecol = ones(ndata,1);
onerow = ones(1,nx);

fltsig = [fltsig; fltsig];
fltpri = [fltpri; fltpri];

% Compute covariance matrix of the joint error pdf (cond. on zeta)
sig1   = onecol*fltsig(:,1)';
sig2   = onecol*fltsig(:,2)';
zeta1  = -0.5*((x-xdata).*(1./sig2 - 1./sig1)).^2;
clear sig1 sig2
zeta1  = 1./(1+exp(zeta1).*(onecol*(fltsig(:,1).*(1-
fltpri(:,1))./fltsig(:,2)./fltpri(:,1))'));
invR   = zeta1./(onecol*(fltsig(:,1)'.^2)) + (1-zeta1)./(onecol*(fltsig(:,2)'.^2));

% Account for missing values: simply sets element to zero, thus missing values are
estimated
% as xpri -- since no data for them.
if ndata>ncomp,
    invR = invR.*(~missing);
end

% Recursively rectify the values by
% building the conditional pdf p(x(t)|x(t-1),x(t-2),...,x(t-Klag),mu,shape,p,H) from the
joint pdf;
% combining with the error pdf; and computing estimates at successive times.
% *** Note that if observations are independent, then nx=nin (i.e., Klag=0) and variable
prev=[].
% *** So just uses p(x(t)|mu,shape,p,H) as given.
prev = 1:(nx-nin);
curr = (nx-nin+1):nx;
xpri = zeros(ndata,length(curr));
for i=1:ndata,
    if length(prev)
        if sum(i==[1:Neach:ndata]),
            x(i,prev) = xdata(i,prev); % This assumes that measurements at time 0 are
exact.
        else
            x(i,prev) = x(i-1,curr);
        end
    end

    % Let us know that something's working
    fprintf(1,'.');
    if ~rem(i,Neach) & Neach>1, fprintf(1,'\n'); end

    invS = 0;
    m    = 0;
    for j=1:h,
      if z(i,j)>eps*eps, % Simply skip all this if component contribution is too small.
        first = (j-1)*nx;

        % Partition covariance matrix of the component.
        S11 = covars(first+prev,prev);
        S21 = covars(first+curr,prev);
        S12 = covars(first+prev,curr);
        S22 = covars(first+curr,curr);

        % Compute covariance of component in conditional distribution.
        if isempty(S11),
            corr = 0;
            S221 = S22;
        else
            if rank(S11)==nin,
                corr = S21*inv(S11);
            else
                corr = S21*pinv(S11); % correlation between current and previous state.
%disp('*** singular ***')
            end
            S221 = S22-corr*S12;
        end

        if rank(S221)==nin,
            compwt = z(i,j)*inv(S221); % component wt. factor
```

```
        else
            compwt = z(i,j)*pinv(S221); % component wt. factor
        end
        invS = invS + compwt;

        if length(prev),
            dprev = corr*(x(i,prev)'-mu(j,prev)');
        else
            dprev = 0;
        end

        onlyone = (z(i,j)-1)>(-eps);
        if onlyone, % Only one component contributes.
            xpri(i,curr) = (mu(j,curr)' + dprev)';
        else
            m = m + compwt*(mu(j,curr)' + dprev);
        end
      end
    end

    % Deal with the headache of rank-deficient covariance matrices.
%     [lamb,Bmat] = goodeigs(invS,rank(invS));
%     lamb   = inv(lamb);
    iR     = diag(invR(i,curr));

    % Compute the weighting for the data and the estimate based on prior knowledge alone.
  if rank(invS)<nin,
    %    Wdat   = Bmat*inv(lamb + Bmat'*iR*Bmat)*Bmat'*iR
    Rr=inv(iR);
    % Compute the weighting for the prior knowledge.
    invS = diag(diag(invS));
    Wpri = Rr*pinv(Rr+pinv(invS));
    Wdat = eye(nin) - Wpri;
    %    Wdat = eye(nin) - Rr*pinv(Rr+Bmat*lamb*Bmat')
    %keyboard
    if ~onlyone,
        xpri(i,curr) = (pinv(invS)*m)';
    end
    disp('*** first one ***')
    Qmatinv = iR + invS;
  else
    Qmatinv = iR+invS;
    Qmat   = inv(iR+invS);
    Wdat   = Qmat*iR;
    if ~onlyone, xpri(i,curr) = (inv(invS)*m)'; end
    %disp('*** second one ***')
    %fprintf(1,'%f',sum(z(i,:)));
    % Compute the weighting for the prior knowledge.
    Wpri   = eye(size(Wdat)) - Wdat; % Qmat*invS;
  end


    % Compute the estimate of the true value (i.e., the rectified or smoothed value).
if 0,
    xrecti = Wdat*xdata(i,curr)' + Wpri*xpri(i,curr)';
else
    xrecti = Qmatinv\(iR*xdata(i,curr)' + invS*xpri(i,curr)');
end
    x(i,curr) = xrecti';

    % *** This is a problem-specific test to see if penicillin product conc. exceeds
    % *** its bounds, which should not be possible unless numerical errors (instability).
    % *** Turn it "on" by setting the 0 in the if to a 1.
    if 0
      if sum(abs(xrecti(3))>2),
        disp(['Questionable point ==> #' int2str(i) ' rank: ' int2str(rank(invS))])
        rank(S11),rank(S12),rank(S21),rank(S22)
        z(i,:)
        compwt
        corr
        diag(Wdat)

        %xrecti
        %Wdat
```

```
        %Wpri
        %Qmat
       keyboard
     end
    end
end

% Pragmatic (brute-force) enforcement of the bounds.
toolow  = x<onecol*bounds(1,:);
toohigh = x>onecol*bounds(2,:);
for k=1:nx,
    if sum(toolow(:,k)),
        x(toolow(:,k),k) = ones(sum(toolow(:,k)),1)*bounds(1,k);
    end
    if sum(toohigh(:,k)),
        x(toohigh(:,k),k) = ones(sum(toohigh(:,k)),1)*bounds(2,k);
    end
end
clear toolow toohigh

if Neach>1, fprintf(1,'\n'); end

% *** Case-specific plot of rectified penicillin conc. ***
% *** Turn it "on" by setting the 0 in the if to a 1.
if 0,
   plot([x(:,6) xdata(:,6) xpri(:,6)])
%    disp('Press any key to continue...')
   pause(1)
%    disp('Type "return" to continue...'); keyboard
end


function [lamb,Bmat,rankA,detA,tol] = goodeigs(A,rankA);
% function [lamb,Bmat,rankA,detA,tol] = goodeigs(A,rankA);

[m,n] = size(A);

if (m-n), error('myrank: A must be square'); end;

% Eliminate small singular values to improve conditioning.
[Bmat,eigs]   = eig(A);
[eigs,iorder] = sort(-real(diag(eigs)));
eigs = -eigs;
Bmat = real(Bmat(:,iorder));

if nargin<2,
  rankA   = sum(eigs>0.00001*eigs(1));
  tol     = sqrt(0.00001*eigs(1));
end

% Compute determinant.
detA = prod(eigs(1:rankA));

lamb = diag(eigs(1:rankA));
Bmat = Bmat(:,1:rankA);
```

D.1.2 Robust State Estimation

```
function [theta,x,y,ynp,mu,shape,p,pm,z,vest,vpri,vbias,zeta,csi]=...

makegen2(theta0,data,x,y,nx,Klag,Neach,bounds,h,mu,shape,p,z,xpg,xsig,ypg,ysig,fbparams,sc
lf);
%function [theta,x,y,ynp,mu,shape,p,pm,z,vest,vpri,vbias,zeta,csi]=...
%
makegen2(theta0,data,x,y,nx,Klag,Neach,bounds,h,mu,shape,p,z,xpg,xsig,ypg,ysig,fbparams,sc
lf);
%
% MAKEGEN2 -- Robust Probabilistic State Estimation by EM Algorithm
%
%            Estimates the parameters of a finite mix. model assuming error-in-vars. &/or
```

```
%            missing values.  Simultaneously, rectifies the y data using function
emcond.m.
%            The input x are assumed rectified already by function emcond.m.
% ACCEPTS:
%   data    = all data windowed; missing values have NAN [ndata,nvars];
nvars=nin+nout*(Klag+1)
%   nx      = number of exogenous input variables
%   Klag    = max lag correlated w/current obs.
%   Neach   = number of independent time series (runs): if Neach=ndata, then all obs. are
indep.
%   h       = number of kernels in nonparam. *** ignored if ixnp is empty.***
%   mu      = means of np. model [h,length(ixnp)]*** ignored if ixnp is empty.***
%   shape   = inverse cov. matrices of np. model [h*length(ixnp),length(ixnp)]
%             *** ignored if ixnp is empty.***
%   p       = priors of np. model [h,1]*** ignored if ixnp is empty.***
%   xpg     = prior probability of gross error in x [nin,1]
%   xsig    = variance of noise in x and gross error in x [nin,2]
%   ypg     = prior probability of gross error in y [nout,1]
%   ysig    = variance of noise in y and gross error in y [nout,2]
%   aparams = indices to optional a-parameters for each model
% RETURNS:
%   theta   = estimated parameters.
%   yp      = predictions at data x. [ndata,nout]
%   pm      = probabilities of applicability for models. [ndata,Km]
%   z       = probabilities of membership to nonparam. kernels. [ndata,h]
%   zeta    = probabilities of gross error in x data. [ndata,nin]
%   csi     = probabilities of gross error in y data. [ndata,nout]
% GLOBALS: DO_E_STEP Z_LAST Y_TRUE ZETA PY_MNP_Z INV_R
% M. L. Thompson 12/20/95

global DO_E_STEP Z_LAST ZETA Y_TRUE INV_R PY_MNP_Z


[ndata,nvars] = size(data);
nx2 = (Klag+1)*nx;
ny  = (nvars - nx2)/(Klag+1);


% Pragmatic (brute-force) enforcement of the bounds.
dbnds = [bounds(:,1:nx) bounds(:,1:nx) bounds(:,(nx+1):(nx+ny)) bounds(:,(nx+1):(nx+ny))];
onecol = ones(ndata,1);
toolow  = data<onecol*dbnds(1,:);
toohigh = data>onecol*dbnds(2,:);
for k=1:nvars,
    if sum(toolow(:,k)),
        data(toolow(:,k),k) = ones(sum(toolow(:,k)),1)*dbnds(1,k);
    end
    if sum(toohigh(:,k)),
        data(toohigh(:,k),k) = ones(sum(toohigh(:,k)),1)*dbnds(2,k);
    end
end
clear toolow toohigh


if isempty(x),
    x = data(:,1:nx2);

    % Perform iterative rectification/density estimation of all data.
    if sum(xsig), % Enter all zeros for xsig if want to skip this step.
    %    p_ge   = [xpg(:); ypg(:)];
    %    esigma = [xsig(:,1); ysig(:,1)];
    %    b      = [xsig(:,2)./xsig(:,1); ysig(:,2)./ysig(:,1)];
        hall   = min(floor(ndata/ny),2*ceil(h/nx*ny+1+h));
        xh     = hall;
        xbounds= bounds(:,1:nx);
        [x,xmu,xshape,xp,xz] = emcond3(x,Klag,Neach,xh,xsig,1-xpg,xbounds);
%starplot(x,data(:,1:nx2),[4 5 6],xmu,xz); pause(1)
%        clear xz
    end
end

ybounds = bounds(:,(nx+1):(nx+ny));
if isempty(y),
    y = data(:,(nx2+1):nvars);
```

```
    yh = h;
    if isempty(mu),
        [y,mu,shape,p,z,py_mnp] = emcond3(y,Klag,Neach,yh,ysig,1-ypg,ybounds);
        if length(p)~=h,
            h      = length(p);
            theta0 = zeros(h,3);
        end
    else
        [y,mu,shape,p,z,py_mnp] = emcond3(y,Klag,Neach,yh,ysig,1-
ypg,ybounds,z,mu,shape,p);
    end
else
    [zjunk,py_mnp] = e_z_all(y,mu,shape,p);
    clear zjunk
end
ynp = y;
Y_TRUE = y;

if 0,
% Hyperparameters for the prior of theta.
meanth = 0;
vars   = ones(ny,1)*3; % vague -- but mean value helps.

% Hybrid model with an output model.
trnstr='trainem(theta0,''fmixnp2'',[],cntrl,data,x,mu,shape,p,z,ysig,1-
ypg,missing,ybounds,fbparams,sclf,meanth,vars)';
[theta,cntrl]   = eval(trnstr);

[fobj,yp,ypk,z] = eval(['fmixnp2(theta,data,x,mu,shape,p,z,ysig,1-
ypg,missing,ybounds,fbparams,sclf,meanth,vars)']);

else
    % Hybrid model with an output model.
    Z_LAST = z;
    PY_MNP_Z = py_mnp;
    missing = zeros(size(y));


    nd = ndata;

%    Y_TRUE
%    fobj = fmixnp4(theta0,data(1:nd,:),x(1:nd,:),mu,shape,p,ysig,1-
ypg,missing,ybounds,fbparams,sclf,meanth,vars);
%    z = Z_LAST;
%    yp= Y_TRUE;
%    keyboard

    trnstr='trainem(theta0,''fmixnp4'',[],cntrl,data,x,mu,shape,p,ysig,1-
ypg,missing,ybounds,fbparams,sclf,meanth,vars)';
for j=1:1,
    DO_E_STEP=0;
    [theta,cntrl] = eval(trnstr);
    theta0 = theta;
    DO_E_STEP=1;
    [fobj,yp,vest,vpri,vbias,pm] = eval(['fmixnp4(theta,data,x,mu,shape,p,ysig,1-
ypg,missing,ybounds,fbparams,sclf,meanth,vars)']);
end
    clear yp
    z = Z_LAST;
    clear Z_LAST
    y = Y_TRUE;
    clear Y_TRUE
    clear PY_MNP_Z
end


function [w,cntrl] =
trainem(w,func,dfunc,cntrl,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17,a18,
a19,a20)
%[w,cntrl]=trainem(w0,func,dfunc,cntrl,a1,a2,...a20)
%
%TRAINEM -- Implements an incremental EM algorithm by making M-Step line searches
%           to improve the parameter estimates w and alternately signalling E-Steps
%           to be performed by routine named by variable func.
```

```
% Global variable DO_E_STEP signals routine named by func to first re-estimate
% indicator vars. in an E-Step before evaluating the function.
% Also, if combined with rectification, func should perform an
% M-Step (rectification) of the variables after the obj. func. evaluation.
%
%  TRAINEM contains the algorithms for training probabilistic models
%  as invoked by makemix.m. The initial values of the parameters are
%  given in w0[nvars].  func and dfunc are the functions supplying
%  the objective function and its derivatives, respectively, with
%  dfunc optional, but suggested.  a1 through a20 are user-defined arguments
%  passed directly to func and dfunc

%       Adapted from train.m by M.L.Thompson 1-21-93

global DO_E_STEP

DO_E_STEP = 1; % This causes next function evaluation to first re-estimate
               % indicator vars. in an E-Step before evaluating the function.
               % Also, if combined with rectification, will perform an
               % M-Step (rectification) of the variables after the evaluation.

w = w(:);
%  TERMINATION TOLERANCES...user can adjust if necessary
gtol = 5.e-5;     % gradient criterion, default 5.e-4
ftol = 5.e-5;     % function progress criterion, default 5.e-5
steptol = eps*10;  % steplength criterion, default 5.e-6
%
nvars=length(w);
np = length(cntrl);
default = [5000 1 1 0 0 0];
if(nvars<100), default(3) = 4; end
cntrl(np+1:6)=default(np+1:6);
if(cntrl(3) < 3),
    cg=1;
    grad_reset = min(50,ceil(0.7*nvars));
else,
    cg=0;
    h=eye(nvars);
end
paramstr='';
for i=1:nargin - 4, paramstr = [paramstr,',a',num2str(i)]; end
str=[func, '(w',paramstr, ')'];
nxtstr=[func, '(nextw',paramstr, ');'];

if(nargin < 4) dfunc=[]; end
numderiv = ~length(dfunc);
if ~numderiv,
    str2 = [dfunc, '(w',paramstr, ',yp,q1,q2,q3)']; end;
end

%******* Evaluate Function and Perform E-Step ********
DO_E_STEP = 0;
f = eval(str); func0 = f; df=zeros(nvars,1); nextw=w;
DO_E_STEP = 0;

dwmin=1.e-4; dwmax=0.1; direc = nextw; sl = 0.01; msgcnt = 0;
count=0; prog=0; msg=[]; fold=0;

meth(1,:) = 'steepest descent';
meth(2,:) = '  FR conj grad  ';
meth(3,:) = '  PR conj grad  ';
meth(4,:) = '   Broyden      ';
meth(5,:) = '   Davidon      ';
meth(6,:) = '   BFGS         ';

if(cntrl(2)>0),
    disp('')
    disp(' FUNCTIONS  OBJECTIVE    METHOD')
    disp([sprintf('%6.0f %12.4g   ', cntrl(4),f) meth(cntrl(3)+1,:)]),
end;
cntrl(2) = abs(cntrl(2));

% First step will be steepest descent.
resetg = 1;
```

```
loop=1;
while(loop)
    extra = 0;
    w=nextw;
    if(numderiv)
        prevf=f;
        dw = sign(direc+eps*ones(nvars,1)).*max(dwmin,min(dwmax, 0.01*abs(sl*direc)));
        for k=1:nvars,
            w(k) = nextw(k) + dw(k);
            %******* Evaluate Function and DO NOT Perform E-Step ********
            f = eval(str);
            df(k) = (f-prevf)/dw(k); w(k) = nextw(k);
        end
        f = prevf; cntrl(4)=cntrl(4)+nvars;
    else
        cntrl(5)=cntrl(5)+1;
        df = eval(str2);
    end
    if(resetg | ~cntrl(3)) % Steepest descent
        direc=-df;
        count = 0;
        if resetg,
%           sl = max(0.01, min([1,2*abs(f/prevdls)]));
%           if (cntrl(2)), disp('...gradient reset...'); end;
            sl = 0.01;
            resetg = 0;
        end;
    elseif(cntrl(3)==3)  % Broyden
        delw = nextw-prevw; deldf = df-prevdf;
        fa = h*delw;
        h = h - fa*fa'/(delw'*fa) + deldf*deldf'/(deldf'*delw);
        direc=-h\df;
    elseif(cg) % Conjugate Gradient
        if (cntrl(3)==1),
            dgg = df'*df;
        else
            dgg = (df - prevdf)'*df;
        end
        gg = prevdf'*prevdf;
        gam = dgg/gg; direc = -df + gam*direc;
        count = count+1;
        resetg = count>=grad_reset;
    elseif(cntrl(3)==4)  % Davidon-Fletcher-Powell
        delw = nextw-prevw; deldf = df-prevdf;
        fa = h*deldf;
        h = h + delw*delw'/(delw'*deldf) - fa*fa'/(deldf'*fa);
        direc=-h*df;
    elseif(cntrl(3)==5)  % Broyden-Fletcher-Goldfarb-Shanno
        delw = nextw-prevw; deldf = df-prevdf;
        fa = h*deldf; u1=delw/(delw'*deldf); u2=fa/(deldf'*fa);
        u = u1 - u2;
        h = h + u1*delw' - u2*fa' + deldf'*fa*(u*u');
        direc=-h*df;
    end

    prevw=nextw; funcC = f;
    prevdf=df; prevdls=df'*direc;
    if(prevdls>=0)
        resetg = 1;
    else
    if(-prevdls<gtol & max(abs(direc))<gtol)
        if(cntrl(2)>0),msg='Training terminated: gradient criterion'; end
        break; % * EXIT WHILE-LOOP
    end
    if(cntrl(4)>=cntrl(1))
        if(cntrl(2)>0),msg='Maximum number of function calls exceeded'; end
        break; % * EXIT WHILE-LOOP
    end

    % no convergence yet, **** Do Line Search ****
    w=prevw+sl*direc;
    wvec = [0 sl]; wvecl = 2;

    %******* Evaluate Function and DO NOT Perform E-Step ********
```

```
DO_E_STEP=0;
fvec = [func0 eval(str)];

cntrl(4)=cntrl(4)+1; go=2;
while(go)
    slold = sl;
    sl = nextstep(prevdls,wvec,fvec);
    if(abs((slold-sl)/(sl+eps))<0.001),
        [fmin,imin]=min(fvec);
        break; %*** EXITS while(go)-loop ***
    end
    wvec=[wvec sl];
    wvecl = wvecl + 1;
    w = prevw+sl*direc;

    %******* Evaluate Function and DO NOT Perform E-Step ********
    DO_E_STEP=0;
    fnew = eval(str);

    cntrl(4)=cntrl(4)+1;
    fvec=[fvec fnew];
    [wvec,isort]= sort(wvec); fvec = fvec(isort);
    [fmin,imin]= min(fvec);

    if (wvecl > 5),
        go=0;
    elseif (imin>1 & imin<wvecl)
        go = go - 1;
    end
    if (go & (abs(sl) < steptol)),
        if(cntrl(2)>0),
            msg='Training terminated: steplength criteria';
        end
        loop=2;
        go=0;
    end
end  % while (go)

sl=wvec(imin);
f=fmin;
nextw = prevw + sl*direc;
if loop==2, break; end; % * EXIT WHILE-LOOP *

if(imin==1)
    sl=0.01;
    if(numderiv & ~msgcnt & cntrl(2))
        disp('Warning: insensitivity in numerical derivatives detected.')
        disp('If training terminates shortly then restarting')
        disp('from termination point may reduce objective further.')
    end
    msgcnt=1;
    break; % * EXIT WHILE-LOOP *
end
if(abs((f-fold)/f)<ftol)
    prog=prog+1;
    if(prog>5)
        msg='Training terminated: no more progress';
        break ; % * EXIT WHILE-LOOP *
    else
        resetg = prog==4;
    end
else,
    prog=0;
end
fold=f;
if(cntrl(2))
    disp(sprintf('%6.0f %12.4g   ',cntrl(4),f));
end
end % * IF (prevdls>0) *

%******* Evaluate Function and Perform E-Step ********
%DO_E_STEP = 1;
eval(nxtstr); % This extra evaluation is to accommodate EM.
extra = 1;
```

```
   DO_E_STEP = 0;

end
w=nextw;

%******* Evaluate Function and Perform E-Step ********
if ~extra,
    %DO_E_STEP = 1;
    cntrl(6) = eval(str);
    DO_E_STEP = 0;
end
disp(msg)


function [fobj,yp,ypk,z]=...
fmixnp2(theta,data,x,mu,shape,p,z,fltsig,fltpri,missing,bounds,fbparams,sclf,meanth,vars);
%function [fobj,yp,ypk,z]=...
%
fmixnp2(theta,data,x,mu,shape,p,z,fltsig,fltpri,missing,bounds,fbparams,sclf,meanth,vars);
%
% theta = parameters (a vector of all parameters to be estimated)
% models = Kx8 matrix, each row is 8-char string which is name of a function
%           (each must return fk(x) and P(Mk|H) &
%           must accept a row from thx, theta, params & a1-a12 and data and datss)
%     Assumes that model probabilities pm are not functions of theta.
% thx    = K x length(theta) matrix of indicators for parameters theta for each model
%           e.g. if 2nd model needs theta(1:5) then thx(2,:)=[ones(1,5),
zeros(1,length(theta)-5)];
% params = Kx12 matrix of indicators for parameters a1-a12 for each model
%           e.g. if 2nd model needs a3,a4,a5, then params(2,:)=[0 0 1 1 1 0 0 0 0 0 0 0];
% data   = first partition of data
% datss = 2nd partition of data (usually only necessary if steady-state data available)
% a1 etc.  = additional parameters required by the models (OPTIONAL)
% *** Essential that user set thx and params correctly ***

global DO_E_STEP Z_LAST ZETA Y_TRUE INV_R

DO_E_STEP=0;

[dum,ny2] = size(Y_TRUE);
ny = ny2/2;
y = Y_TRUE(:,(ny+1):ny2);

% Hyperparameter: P(MNp|H)
P_Mnp = 0.95;

[ndata,nvars] = size(data);
[dum,h]  = size(Z_LAST);
[dum,nx2] = size(x);
nx = nx2/2;
ydata = data(:,(nx2+ny+1):nvars);

if DO_E_STEP,
    % Compute covariance matrix of the joint error pdf (cond. on zeta)
    onecol = ones(ndata,1);
    sig1  = onecol*fltsig(:,1)';
    sig2  = onecol*fltsig(:,2)';
    ZETA  = -0.5*((y-ydata).*(1./sig2 - 1./sig1)).^2;
    clear sig1 sig2
    ZETA  = 1./(1+exp(ZETA).*(onecol*(fltsig(:,1).*(1-
fltpri(:,1))./fltsig(:,2)./fltpri(:,1))'));
    INV_R = ZETA./(onecol*(fltsig(:,1)'.^2)) + (1-ZETA)./(onecol*(fltsig(:,2)'.^2));

    % Account for missing values: simply sets element to zero, thus missing values are
estimated
    % as xpri -- since no data for them.
    iscomp = ~sum(missing');
    imiss  = find(~iscomp);
    ncomp  = sum(iscomp);
    if ndata>ncomp,
        INV_R = INV_R.*(~missing);
    end

    % Use winner-take-all membership rule.
```

```
    dist = dist2(Y_TRUE,mu,shape);
    if (h == 1)
        z = ones(ndata,1);
    else
        z = (~(dist - ones(h,1)*min(dist)))';
    end
    Z_LAST = z;
end

% Compute vp, given theta.
theta = reshape(theta,h,ny);
%plot(theta),pause(1)
vpri  = fbprior(y,fbparams,sclf); % Prior model.
vnp   = Z_LAST*(theta*P_Mnp); % Nonparametric model.
vest  = vnp + vpri;

% Estimate state variables at time t+1.
yp = fbout(vest,x,Y_TRUE(:,1:ny),fbparams,sclf);
%hold on;
%plot(yp)
%pause(1)
if DO_E_STEP,
theta
    Y_TRUE = [[Y_TRUE(1,1:ny); yp(2:ndata,:)] yp];
end

% Compute log-prior of theta.
log_prior = fbthpri(theta(:),meanth,vars);

% Compute objective function.
dy   = ydata - yp;
fobj = (0.5*sum(sum(dy.^2.*INV_R)) - log_prior)/ndata/ny;
disp(['fobj=',num2str(fobj)]),pause(1)
```

D.2. Prediction

```
function
[xp,yupdate,yfore,invR,Wdat]=batchprd(theta,x,ydata,fltsig,fltpri,bounds,h,ny,nin,ndata,nc
omp,Neach,mu,shape,p,missing,fbparams,sclf);
%function
[xp,yupdate,yfore,invR,Wdat]=batchprd(theta,x,ydata,fltsig,fltpri,bounds,h,ny,nin,ndata,nc
omp,Neach,mu,shape,p,missing,fbparams,sclf);
%BATCHPRD= Performs the M-Step of a rectification scheme involving corrupted data and a
%          finite mixture of normals as the joint pdf
% fltsig  = std. devs. of meas. fault components 1 and 2 [nx,2]
% fltpri  = prior probability of being from first fault component [nx,1]
% bounds  = upper and lower bounds of true x (Inf is valid) [nx,2]
% h       = number of component in joint pdf p(X{t:Klag}|H) mixture
% nx      = number of total variables = nin*(Klag+1), where Klag=max lag correlated
w/current obs.
% nin     = number of variables in each original observation
% ndata   = number of windowed observations
% ncomp   = number of complete observations (i.e. obs. w/out missing values in any var.)
% Neach   = number of time series (or runs); if obs.are indep. then Neach=ndata and
nin=nx.
% mu      = component means of joint pdf p(X{t:Klag}|H) [h,nx]
% shape   = inverse covars. of joint pdf p(X{t:Klag}|H) [h*nx,nx]*** ACTUALLY, NOT USED!
% p       = prior prob. of comps for jnt.pdf p(X{t:Klag}|H) [1,h]
% covars  = covars of joint pdf p(X{t:Klag}|H) [h*nx,nx] *** THIS IS USED INSTEAD OF
SHAPE!
%         (i.e. shape holds either invs. or gen. invs of covars)
% z       = component membership probabilities [ndata,h]
% missing = flags of values that are missing [ndata,nx]
% OUTPUTS:
% yfore = estimate based on prior knowledge alone (x is a wtd. avg. of yfore and xdata).
% yupdate = estimate combining prior knowledge and current data point.

xp = x;
h = length(p);
    bounds  = bounds(:,4:6);
onecol = ones(ndata,1);
```

```
onerow = ones(1,ny);
theta = reshape(theta,h,3);

fltsig = [fltsig; fltsig];
fltpri = [fltpri; fltpri];

yupdate = zeros(size(ydata));

% Recursively rectify the values by
% building the conditional pdf p(x(t)|x(t-1),x(t-2),...,x(t-Klag),mu,shape,p,H) from the
joint pdf;
% combining with the error pdf; and computing estimates at successive times.
% *** Note that if observations are independent, then ny=nin (i.e., Klag=0) and variable
prev=[].
% *** So just uses p(x(t)|mu,shape,p,H) as given.
prev = 1:(ny-nin);
curr = (ny-nin+1):ny;
yfore = zeros(ndata,nin);
for i=1:ndata,
    if length(prev)
        if sum(i==[1:Neach:ndata]),
            yupdate(i,prev) = ydata(i,prev); % This assumes that measurements at time 0
are exact.
        else
            yupdate(i,prev) = yupdate(i-1,curr);
        end
    end

    shrdc = [];
    for j=1:h,
        shrdc = [shrdc; shape((j-1)*ny+prev,prev)];
    end
    z = e_z_all(yupdate(i,prev),mu(:,prev),shrdc,p);

    % Let us know that something's working
    fprintf(1,'.');
    if ~rem(i,Neach) & Neach>1, fprintf(1,'\n'); end

    invS = 0;
    m    = 0;
    for j=1:h,
      if z(j)>eps*eps, % Simply skip all this if component contribution is too small.
        first = (j-1)*ny;

        % Partition covariance matrix of the component.
        covars(first+(1:ny),:) = pinv(shape(first+(1:ny),:));
        S11 = covars(first+prev,prev);
        S21 = covars(first+curr,prev);
        S12 = covars(first+prev,curr);
        S22 = covars(first+curr,curr);

        % Compute covariance of component in conditional distribution.
        if isempty(S11),
            corr = 0;
            S221 = S22;
        else
            if rank(S11)==nin,
                corr = S21*inv(S11);
            else
                corr = S21*pinv(S11); % correlation between current and previous state.
%disp('*** singular ***')
            end
            S221 = S22-corr*S12;
        end

        if rank(S221)==nin,
            compwt = z(j)*inv(S221); % component wt. factor
        else
            compwt = z(j)*pinv(S221); % component wt. factor
        end
        invS = invS + compwt;

        if length(prev),
            dprev = corr*(yupdate(i,prev)'-mu(j,prev)');
```

```
        else
            dprev = 0;
        end

        onlyone = (z(j)-1)>(-eps);
        if onlyone, % Only one component contributes.
            yfore(i,:) = (mu(j,curr)' + dprev)';
        else
            m = m + compwt*(mu(j,curr)' + dprev);
        end
      end
    end

    if rank(invS)<nin,
        if ~onlyone,
            yfore(i,:) = (pinv(invS)*m)';
        end
    else
        if ~onlyone,
            yfore(i,:) = (inv(invS)*m)';
        end
    end

    % Pragmatic (brute-force) enforcement of the bounds.
    toolow  = yfore(i,:)<bounds(1,:);
    toohigh = yfore(i,:)>bounds(2,:);
    for k=1:3,
        if toolow(k),
            yfore(i,k) = bounds(1,k);
        end
        if toohigh(k),
            yfore(i,k) = bounds(2,k);
        end
    end
    clear toolow toohigh


%    if Neach>1, fprintf(1,'\n'); end

    [z,py_mnp_z] = e_z_all([yupdate(i,prev) yfore(i,:)],mu,shape,p);

    % Hyperparameters: P(MNp|H) & Py_Mdef.
    P_Mnp   = 0.95;
    %Py_Mdef = 1e-4;
    Py_Mdef = eps*max(py_mnp_z);
    beta    = Py_Mdef*(1-P_Mnp)/P_Mnp;

    % Compute vp, given theta.
    vpri    = fbprior(yfore(i,:),fbparams,sclf(:,4:6)); % Prior model.
    vbias   = z*theta; % Nonparametric model.
    %P_Mnp_y = py_mnp_z./(beta + py_mnp_z); % Probability of applying nonparametric model.
    P_Mnp_y = ones(size(py_mnp_z));
    vest    = vpri + vbias.*(P_Mnp_y*ones(1,nin));

    % Estimate state variables.
    yfore(i,:) = fbout(vest,x(i,:),yupdate(i,prev),fbparams,sclf);

    % Compute covariance matrix of the joint error pdf (cond  on zeta)
    sig1   = fltsig(4:6,1)';
    sig2   = fltsig(4:6,2)';
    zeta1  = -0.5*((yfore(i,:)-ydata(i,curr)).*(1./sig2 - 1./sig1)).^2;
    clear sig1 sig2
    zeta1  = 1./(1+exp(zeta1).*((fltsig(4:6,1).*(1-
fltpri(4:6,1))./fltsig(4:6,2)./fltpri(4:6,1))'));
    invR   = zeta1./((fltsig(4:6,1)'.^2)) + (1-zeta1)./((fltsig(4:6,2)'.^2));

    % Account for missing values: simply sets element to zero, thus missing values are
estimated
    % as yfore -- since no data for them.
    if ndata>ncomp,
        invR = invR.*(~missing(i,:));
    end
    iR = diag(invR);
```

```
    Qmatinv = iR + invS;

    % Compute the estimate of the true value (i.e., the rectified or smoothed value).
%    yrecti = (Qmatinv+iR)\(iR*ydata(i,curr)' + Qmatinv*yfore(i,:)');
    yrecti = 0.5*(ydata(i,curr)' + yfore(i,:)');
    yupdate(i,curr) = yrecti';

    % *** Case-specific plot of rectified penicillin conc. ***
    % *** Turn it "on" by setting the 0 in the if to a 1.
    if 0,
       plot([yupdate(:,6) ydata(:,6) yfore(:,3)])
    %    disp('Press any key to continue...')
       pause(1)
    %    disp('Type "return" to continue...'); keyboard
    end

end
```

Appendix E: State Estimation from Corrupted Data

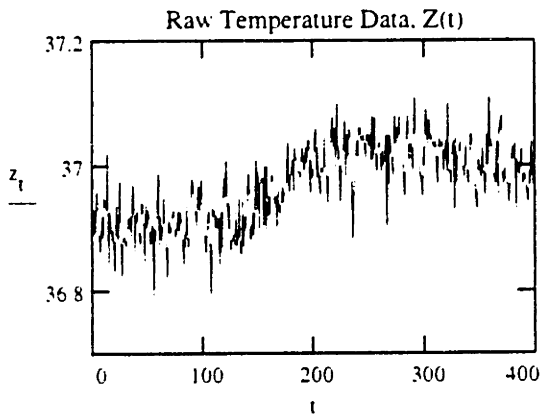
The following is the Mathcad Plus worksheet that performed time series analyses and dynamic rectification for the case study of Section 7.2. Functions **lcorr** and **plcorr** from the "Mathcad Signal Processing Handbook" were used to calculate the sample autocorrelation (ACF) and partial autocorrelation (PACF) functions (see Wei, 1990, for definitions of ACF and PACF). All other calculations were performed by evaluating the expressions I wrote below as Mathcad mathematical relations.

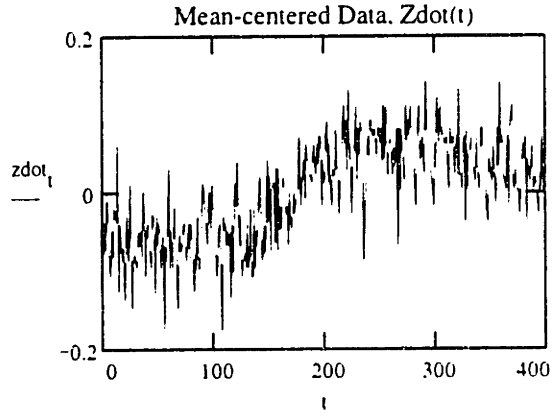
Time Series Analysis and Model Identification:

The time series is actual data from an industrial fermentation process. The series has N=402 temperature measurements taken at 20-minute intervals.

$N = 402 \quad t = 0..N-1 \qquad z_t = A_{t,2} \qquad zdot_t = z_t - mean(z)$

The plots of Z_t and of $Z_{dot_t}=Z_t-\mu$ show a distinct trend (i.e., nonstationarity in the mean)

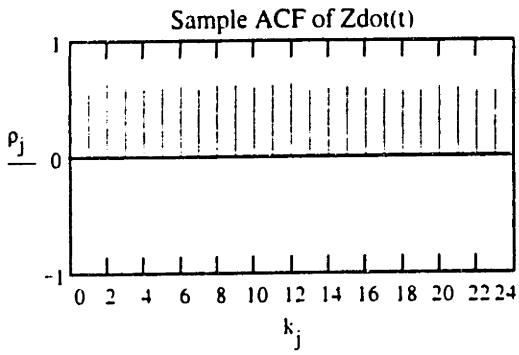


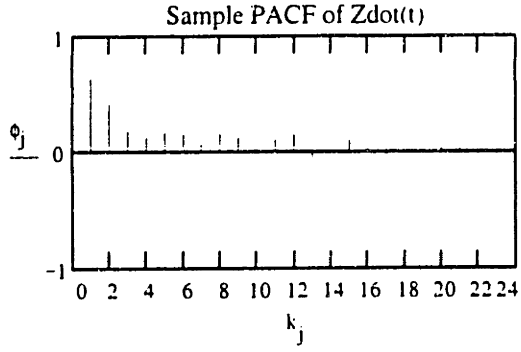


As shown below, the sample ACF ρ_k shows a very slow decay, characteristic of data with trends. Thus differencing is recommended. (The sample PACF ϕ_{kk} is also shown.)

$\rho 0 = lcorr(z,z) \quad \phi 0 = plcorr(z) \quad j = 0..3\cdot(N-1) \quad \rho_j = if\left(mod(j,3)=0,\rho 0_{\frac{j}{3}}\right) \quad \phi_j = if\left(mod(j,3)=0,\phi 0_{\frac{j}{3}}\right)$

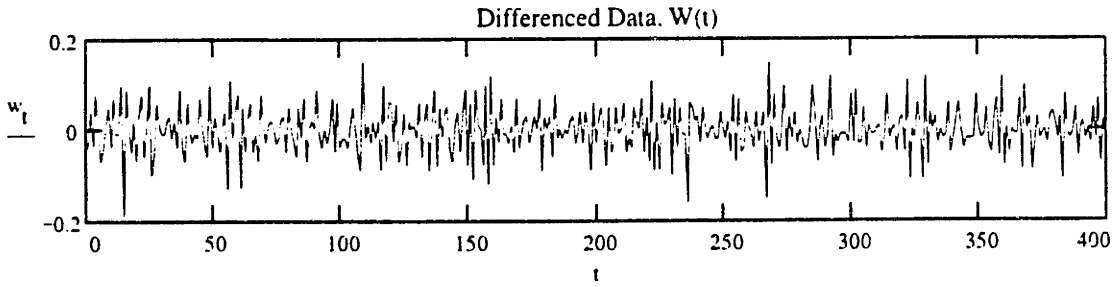
$k_j = if\left(mod(j,3)=1,\frac{j-1}{3},if\left(mod(j,3)=2,\frac{j-2}{3}+1,\frac{j}{3}\right)\right)$ <--(This is done so ACF plots show spikes rather than points connected by lines.)





Differencing succeeded in removing the trend.

$$w_t = \text{if}\left(t, zdot_t - zdot_{t-1}, zdot_0\right)$$



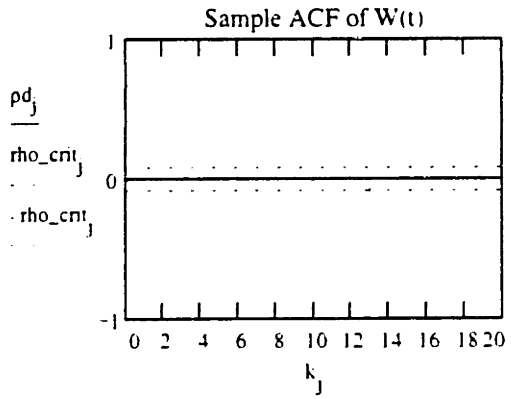
The sample ACF and sample PACF of the differenced data are shown below.

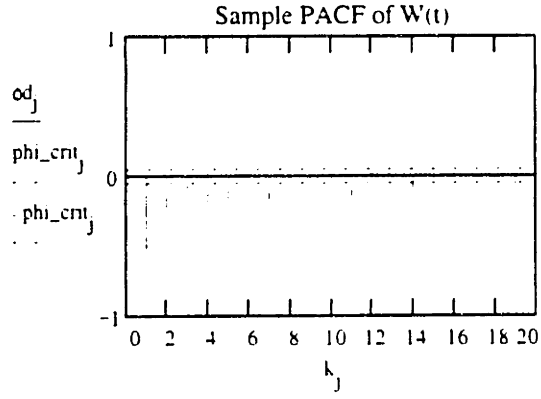
$$rhod = \text{lcorr}(w, w) \qquad pd_j = \text{if}\left(\text{mod}(j,3) = 0, rhod_{\frac{j}{3}}\right)$$

$$t\_crit\_5pct = 1.645 \qquad rho\_crit_j = t\_crit\_5pct \cdot \left(\frac{1}{\sqrt{N}}\right)$$

$$phid = \text{plcorr}(w) \qquad \phi d_j = \text{if}\left(\text{mod}(j,3) = 0, phid_{\frac{j}{3}}\right)$$

$$phi\_crit_j = \frac{1}{\sqrt{N}}$$





After analyzing the sample ACF and sample PACF of the differenced data, I concluded that a suitable ARIMA(p,d,q) model was IMA(1,1) because [1] single differencing (d=1) was used to remove the trend, [2] the sample ACF of the differenced series shows the spike at k=1 being most significant, and [3] the sample PACF dies off exponentially. Thus, p=0 and q=1.

To check if a deterministic trend parameter was necessary, I computed the t-ratio:

$$Wmean = \text{mean}(w) \qquad \gamma 0 = \text{covar}(w, w) \qquad Swmean = \sqrt{\frac{\gamma 0_0}{N}} \qquad t\_ratio = \frac{Wmean}{Swmean}$$

$$Wmean = -4.332 \cdot 10^{-5} \qquad \gamma 0_0 = 1.133 \qquad Swmean = 0.053 \qquad t\_ratio = -8.159 \cdot 10^{-4}$$

The t-ratio indicated that the deterministic trend parameter was not significant. Thus, the model I chose was IMA(1,1): $(1-B)Z_{dot_t} = (1-\theta B)a_t$.

Parameter Estimation:

The next step was to estimate the parameter θ. Conditional least-squares was used. The residuals were estimated using the difference equation $Z_{dec_t} = Z_{dec_{t-1}} + a_t - \theta a_{t-1}$. Thus, $a_t = Z_{dec_t} - Z_{dec_{t-1}} + \theta a_{t-1} = W_t + \theta a_{t-1} = W_t - \theta b_t$, where $b_t = -a_{t-1}$, for $t>1$ and $b_0=0$. Solution yields $\theta = \mathbf{0.8441}$. (Below, successive substitution was used until $\theta := \theta$.)

$\theta 0 = 0.844066$

$t = 1..N-1$

$a_0 = w_0 \quad a_t = w_t - \theta 0 \cdot a_{t-1} \quad b_t = if(t, -a_{t-1}, 0) \quad \theta = \dfrac{(b^T \cdot w)_{0,0}}{(b^T \cdot b)_{0,0}} \qquad \theta = 0.844066 \qquad t = 0..N-1$

The standard error of a_t was used to estimate σ_a: $sighat = \sqrt{\dfrac{(a^T \cdot a)_{0,0}}{N-1}} \qquad sighat = 4.064 \cdot 10^{-2}$

Just to check, the theoretical ACF and PACF of the IMA(1,1) model with θ=0.8441 was compared to the sample ACF ρ_1 and the sample PACF. They matched fairly well.

Model ACF, ρ_1: $\dfrac{-\theta}{1-\theta^2} = -0.4929$ Sample ACF, ρ_1: $rhod_1 = -0.528$

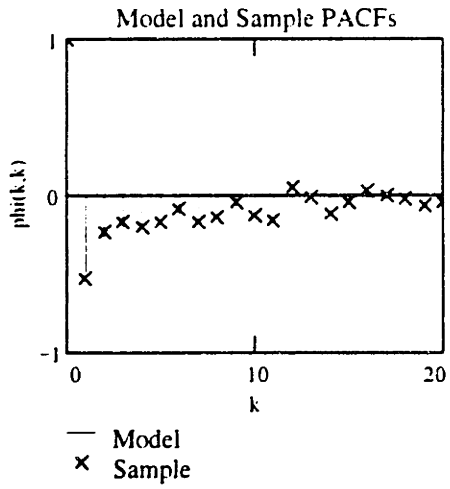
$phi_t = if\left(t, \dfrac{-\theta^t \cdot (1-\theta^2)}{1-\theta^{2\cdot(t+1)}}, 1\right) \quad omodel_j = if(mod(j,3), 0, phi_{\frac{j}{3}}) \quad i = 0..20$

Model PACF:

	0
0	1
1	-0.493
2	-0.321
3	-0.233
4	-0.179
5	-0.142
6	-0.115
7	-0.094
8	-0.078
9	-0.065
10	-0.054
11	-0.045

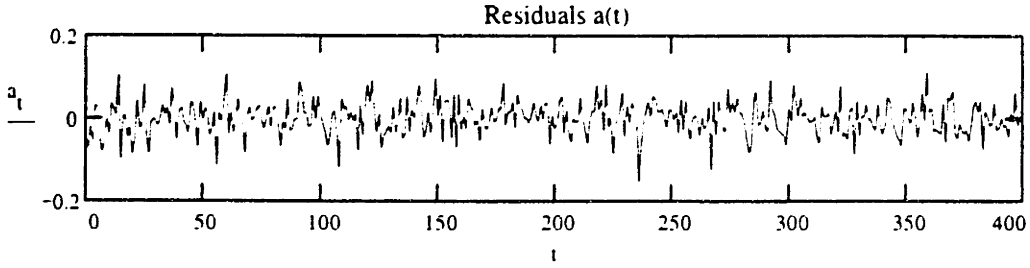
Sample PACF:

	0
0	1
1	-0.528
2	-0.234
3	-0.168
4	-0.198
5	-0.173
6	-0.083
7	-0.171
8	-0.139
9	-0.044
10	-0.121
11	-0.158
12	0.05

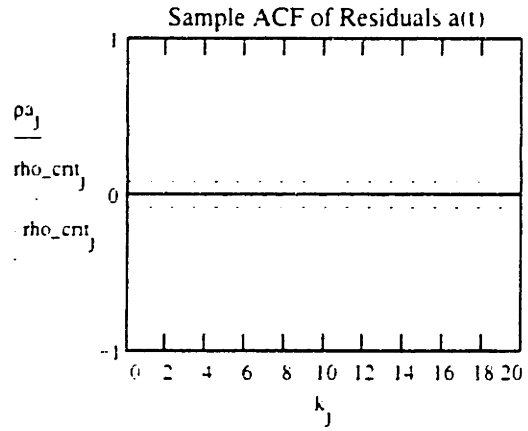


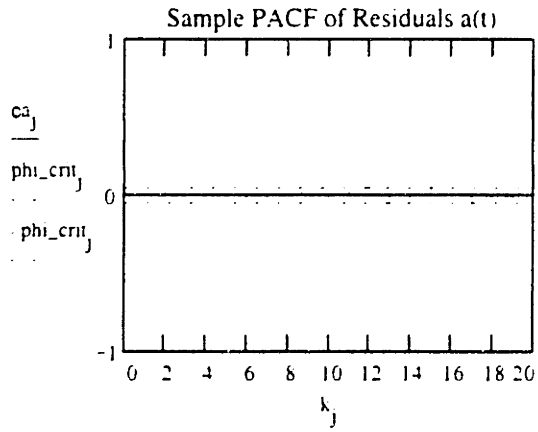
Time Series Analysis on Residuals:

A time series analysis of the residuals a_t was performed to confirm that they did not show significant correlation. The residuals (and an estimate of their variance) were calculated above. Below, their sample ACF and sample PACF are calculated and plotted.



rhoa = lcorr(a,a) phia = plcorr(a) ρa_j = if(mod(j,3) = 0, rhoa$_{\frac{j}{3}}$) ϕa_j = if(mod(j,3) = 0, phia$_{\frac{j}{3}}$)





The residuals showed no appreciable correlation and thus were deemed acceptable.

One-step-ahead Forecasts and 95% Forecast Limits:

Next, the one-step-ahead forecasts at times t=M1,...,M2 were computed and plotted. The forecast from the t-th point is denoted $Z_{hat_t} = (1-\theta)Z_{dot_{(t-1)}} + \theta Z_{hat_{(t-1)}}$. The 95% forecast limits were also computed and plotted.

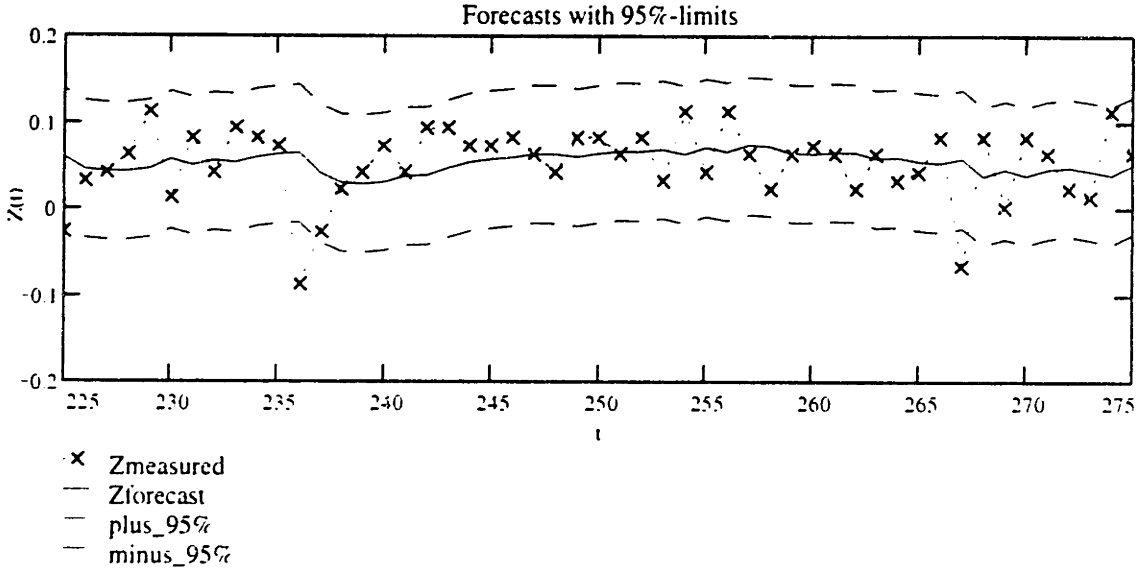
$M1 := 0 \qquad M2 := N - 1$

$t := M1..M2 \qquad tl_t := t - M1$

$zsig := sighat \qquad limit95 := 1.96 \cdot zsig$

$z\_hat_0 := zdot_0$

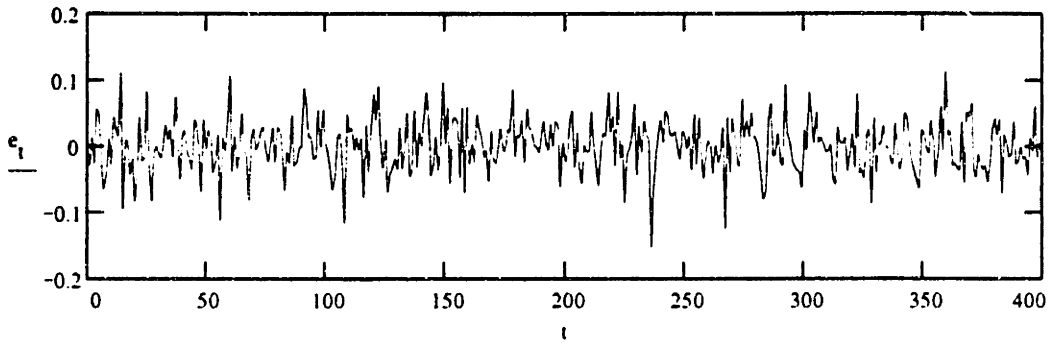
$z\_hat_{(tl_t)} := if\left(t, (1-\theta)\cdot zdot_{tl_t - 1} - \theta \cdot z\_hat_{tl_t - 1}, z\_hat_{(tl_t)}\right)$



As seen from the plot, the model forecasts were reasonable. (However, the data that are forecast were also used in parameter estimation, so a more valid check would be to see how well the forecasts compare with novel data, or limit the range of data used in parameter estimation and use the remaining data for validation).

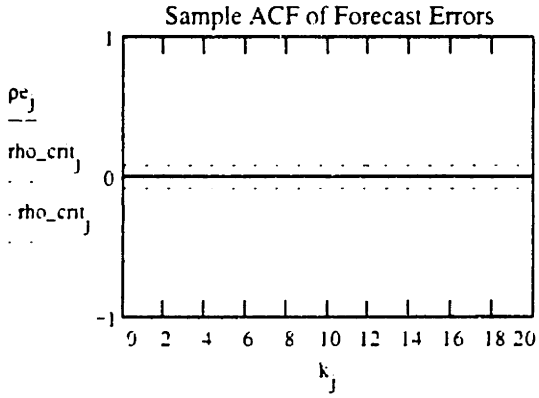
The forecast errors and their sample ACF and sample PACF appear below.
No significant autocorrelation was found in the forecast errors.

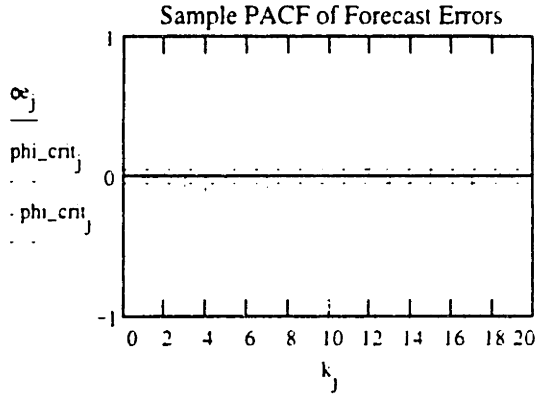
$e_t = zdot_t - z\_hat_t$



rhoe = lcorr(e,e) $\rho e_j = if\left(mod(j,3) = 0, rhoe_{\frac{j}{3}}\right)$

phie = plcorr(e) $\phi e_j = if\left(mod(j,3) = 0, phie_{\frac{j}{3}}\right)$





Outlier Detection -- A first approach:

The first approach to outlier detection, as described in Section 7.2.2, Table 7.1, is demonstrated below.

These functions represent the parameters and resulting distribution of a product of two normal distributions with means μ and m and variances σ^2 and s^2, respectively.

$$mu(\mu,m,\sigma,s) := \frac{s^2\cdot\mu - \sigma^2\cdot m}{\left(s^2 - \sigma^2\right)} \qquad sig\_sqr(\sigma,s) := \frac{s^2\cdot\sigma^2}{\left(s^2 - \sigma^2\right)} \qquad d(\mu,m,\sigma,s) := \exp\left(-0.5\cdot\frac{(m-\mu)^2}{\left(s^2 - \sigma^2\right)}\right)$$

$$g(x,\mu,m,\sigma,s) := \frac{d(\mu,m,\sigma,s)}{2\cdot\pi\cdot s\cdot\sigma}\cdot\exp\left(-0.5\cdot\frac{(x - mu(\mu,m,\sigma,s))^2}{sig\_sqr(\sigma,s)}\right)$$

The next functions define the 2-component mixture model of Gaussians that represents the posterior pdf $p(X_t \mid Z_1, X_{t-k})$. The components are defined by function pz0 for the noise-component and pz1 for the outlier-component. The probability of a gross error p_g was chosen to be 0.01 and the ratio between the gross error pdf and noise pdf standard deviations was defined as b = 20.

$$pg := 0.01 \quad b := 20 \qquad denom(z,zh,zt) := (1 - pg)\cdot d(zh,zt,zsig,sighat)\cdot\sqrt{sig\_sqr(zsig,sighat)} \ldots$$
$$\ldots + pg\cdot d(zh,zt,zsig,b\cdot sighat)\cdot\sqrt{sig\_sqr(zsig,b\cdot sighat)}$$

$$pz0(z,zhat,zt) := \frac{(1 - pg)\cdot g(z,zhat,zt,zsig,sighat)}{denom(z,zhat,zt)} \qquad pz1(z,zhat,zt) := \frac{pg\cdot g(z,zhat,zt,zsig,b\cdot sighat)}{denom(z,zhat,zt)}$$

A point is considered an outlier if the pdf of the outlier component is greater than the pdf of the noise component when both are evaluated at the forecast value $Z_{forecast_t}$. The rectified time series is determined by replacing each outlier with the mean of the outlier-component given Z_t and $Z_{forecast_t}$. Non-outliers are left unchanged.

$$outlier_{(t1_t)} := pz0\left(z\_hat_{(t1_t)}, z\_hat_{(t1_t)}, zdot_{(t1_t)}\right) < pz1\left(z\_hat_{(t1_t)}, z\_hat_{(t1_t)}, zdot_{(t1_t)}\right)$$

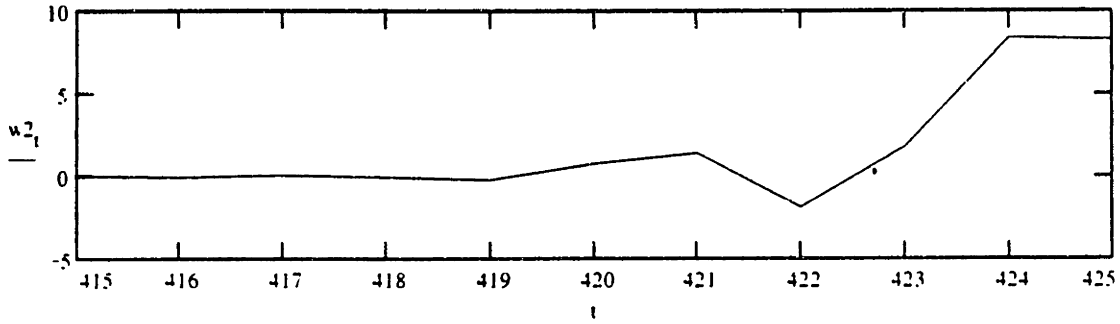
$$\sum_{j=M1}^{M2} outlier_j = 0$$

As expected, this method detects no outliers in the data used to generate the model. However, below are the results when applied to a second series with corrupted values at the end of the series.

Outlier Detection Applied to New Data:

The outlier detection approach proposed above was applied to a new time series. The new series has N_2=426 observations and is plotted below. The plot indicates there are six outliers near the end of the time series: t = {419, 420, 421, 423, 424, 425}.

$$N2 = 426 \quad t = 0..N2 - 1 \qquad z2_t = B_{t,2}\, zdot2_t = z2_t - mean(z2) \qquad w2_t = if\left(t, zdot2_t - zdot2_{t-1}, zdot2_0\right)$$



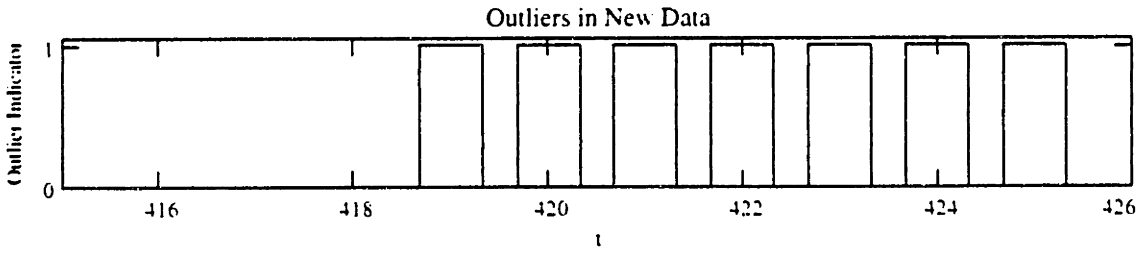
$$M1b = 0 \quad M2b = N2 - 1 \quad t = M1b..M2b \quad z\_hat2_0 = zdot2_0$$

$$z\_hat2_t = if\left(t, (1 - \theta)\cdot zdot2_{t-1} - \theta\cdot z\_hat2_{t-1}, z\_hat2_0\right)$$

This method detects seven outliers in the new data instead of six. Observation 422 is incorrectly identified as an outlier.

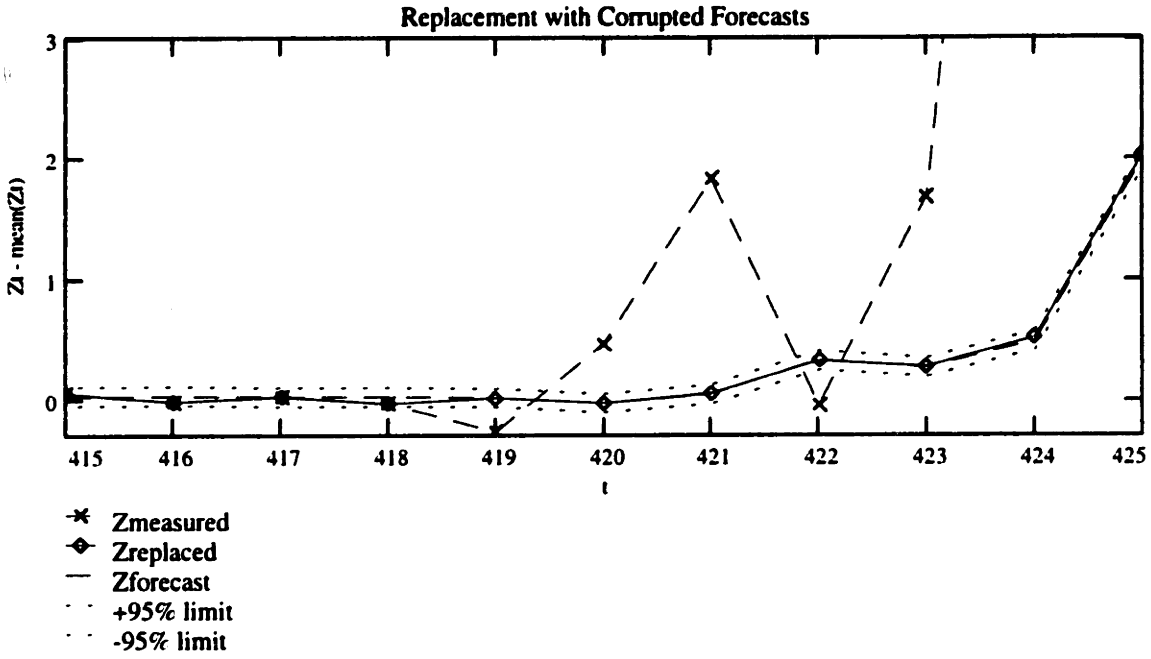
$$outlier2_t = pz0\left(z\_hat2_t, z\_hat2_t, zdot2_t\right) < pz1\left(z\_hat2_t, z\_hat2_t, zdot2_t\right) \qquad \mu_t = mu\left(z\_hat2_t, zdot2_t, zsig, sighat\right) \quad m_t = mu\left(z\_hat2_t, zdot2_t, zsig, b\cdot sighat\right)$$

$$\sum_{t=M1b}^{M2b} outlier2_t = 7$$



Each outlier is replaced by its most probable value conditioned upon the observation being drawn from the outlier component of the posterior pdf. This replacement value is approximately equal to the forecast value. However, at this stage, the forecasts at the end of the series are still based upon corrupted values.

$zrect2_t = if(outlier2_t, m_t, zdot2_t)$ limit95 = 1.96·zsig

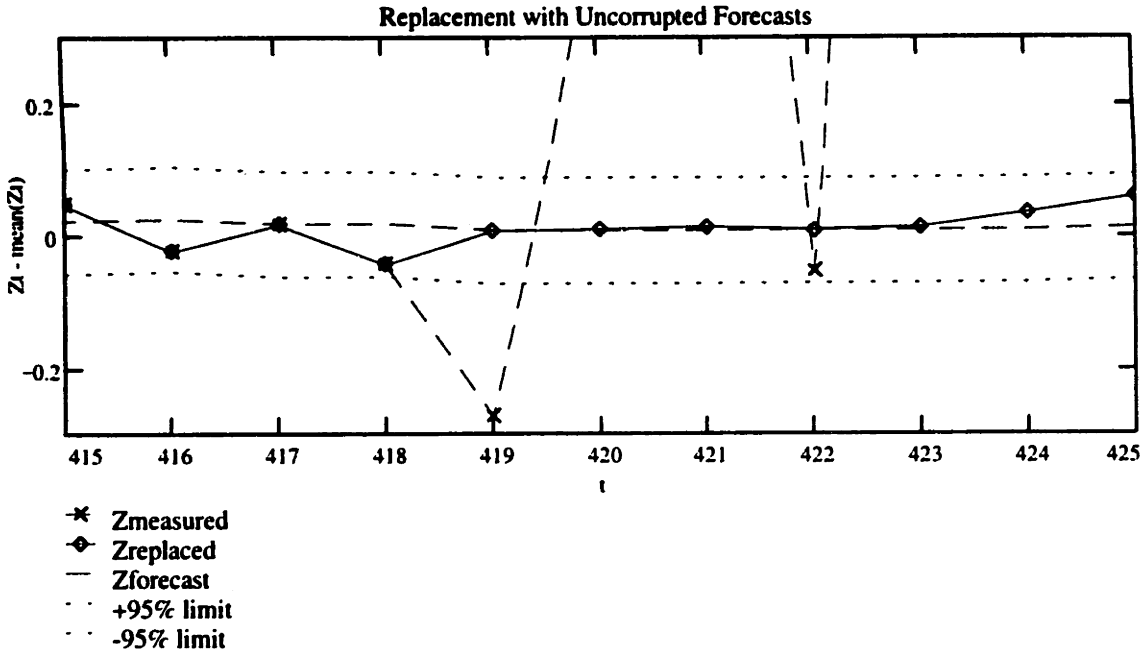


Below, the corrupted data are replaced before making subsequent forecasts. The result is a large reduction in the influence of the outliers on the forecasts.

$$z\_hat2\_r_0 = z\_hat2_0$$

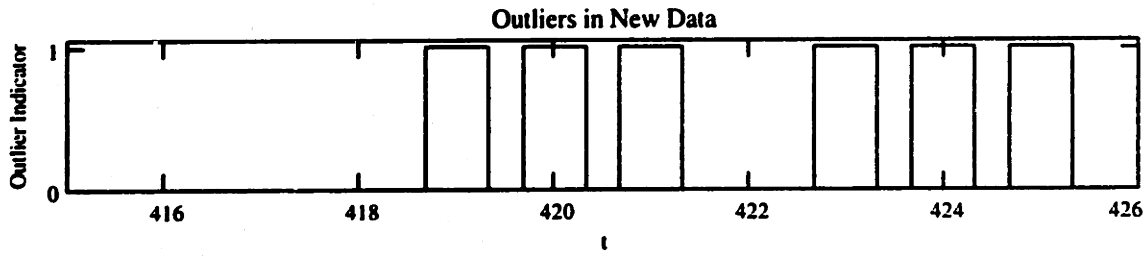
$$z\_hat2\_r_t = \text{if}\left[\begin{array}{l} t,(1-\theta)\cdot \text{if}\left(outlier2_{t-1}, mu\left(z\_hat2\_r_{t-1}, zdot2_{t-1}, zsig, b\cdot sighat\right), zdot2_{t-1}\right) \ldots, z\_hat2\_ \\ \quad +\theta\cdot z\_hat2\_r_{t-1} \end{array}\right.$$

$$zrect2_t = \text{if}\left(outlier2_t, mu\left(z\_hat2\_r_t, zdot2_t, zsig, b\cdot sighat\right), zdot2_t\right)$$



Now the technique correctly classifies observation t=422 as a non-outlier and identifies the six outliers at t = {419, 420, 421, 423, 424, 425}.

$$outlier2_t = pz0\left(z\_hat2\_r_t, z\_hat2\_r_t, zdot2_t\right) < pz1\left(z\_hat2\_r_t, z\_hat2\_r_t, zdot2_t\right) \qquad \sum_{t=M1b}^{M2b} outlier2_t = 6$$



Rectification of the New Data:

Unlike the outlier detection method above, which only replaces the outliers with new values, rectification replaces ALL observations with a more probable value based upon the posterior distribution. This is explained more fully in Section 7.2.2, Table 7.2.

Below are the functions used for rectification using the Expectation-Maximization (EM) algorithm.

E-Step:

Compute the expected values of the indicator variables, which are the posterior probabilities of the observation being drawn from each component pdf given (conditioned upon) the observation.

$$\text{zsum}(t, zx, zfore) = pz0\left(zx, zfore, zdot2_t\right) + pz1\left(zx, zfore, zdot2_t\right)$$

$$\text{zi}(i, t, zx, zfore) = \text{if}\left(i, 1 - \frac{pz0\left(zx, zfore, zdot2_t\right)}{\text{zsum}(t, zx, zfore)}, \frac{pz0\left(zx, zfore, zdot2_t\right)}{\text{zsum}(t, zx, zfore)}\right)$$

M-Step:

Find the most likely value of the process variable, i.e., the value that maximizes the posterior pdf of the true value given (conditioned upon) the observation.

$$\text{xmax}(t, zrect, zfore) = \frac{\frac{\text{zi}(0, t, zrect, zfore)}{\text{sig\_sqr}(zsig, sighat)} \cdot mu\left(zfore, zdot2_t, zsig, sighat\right) \ldots + \frac{\text{zi}(1, t, zrect, zfore)}{\text{sig\_sqr}(zsig, b \cdot sighat)} \cdot mu\left(zfore, zdot2_t, zsig, b \cdot sighat\right)}{\frac{\text{zi}(0, t, zrect, zfore)}{\text{sig\_sqr}(zsig, sighat)} + \frac{\text{zi}(1, t, zrect, zfore)}{\text{sig\_sqr}(zsig, b \cdot sighat)}}$$

EM Rectification was performed below by creating a matrix with even-numbered rows as forecasts and odd-numbered rows as rectified values, and with columns as successive iterations in the EM algorithm. The algorithm was implemented this way so that the forecast at each time would be computed based upon the rectified values at earlier times.

Note that only 3 iterations of the EM algorithm are performed for each point. This value was chosen by experimenting with different iteration counts and checking for convergence. Alternatively, a convergence criterion could be implemented to provide automatic determination of the number of iterations for each data point.

Iteration counter, j: $jmax = 3 \qquad j = 1..jmax$

Initialize forecasts and rectified values at time t=0 and for first iteration.

$Q_{0,0} = z\_hat2_0 \quad Q_{1,0} = Q_{0,0} \quad Q_{0,j} = Q_{0,j-1} \quad Q_{1,j} = Q_{0,j}$

$u = 2..(2 \cdot N2 - 1)$

$$Q_{u,0} = \text{if}\left[\text{mod}(u,2)=0, (1-\theta)\cdot Q_{u-1,0} + \theta \cdot Q_{u-2,0}, \text{xmax}\left(\frac{u-1}{2}, \text{zdot2}_{\frac{u-1}{2}}, Q_{u-1,0}\right)\right]$$

Fill the matrix with alternating rows of forecasts and rectified values for each iteration.

$$Q_{u,j} = \text{if}\left[\text{mod}(u,2)=0, (1-\theta)\cdot Q_{u-1,j} + \theta \cdot Q_{u-2,jmax}, \text{xmax}\left(\frac{u-1}{2}, Q_{u,j-1}, Q_{u-1,jmax}\right)\right]$$

Extract the forecasts.

$t = 0..N2 - 1$

$zf_t = Q_{2\cdot t,jmax}$

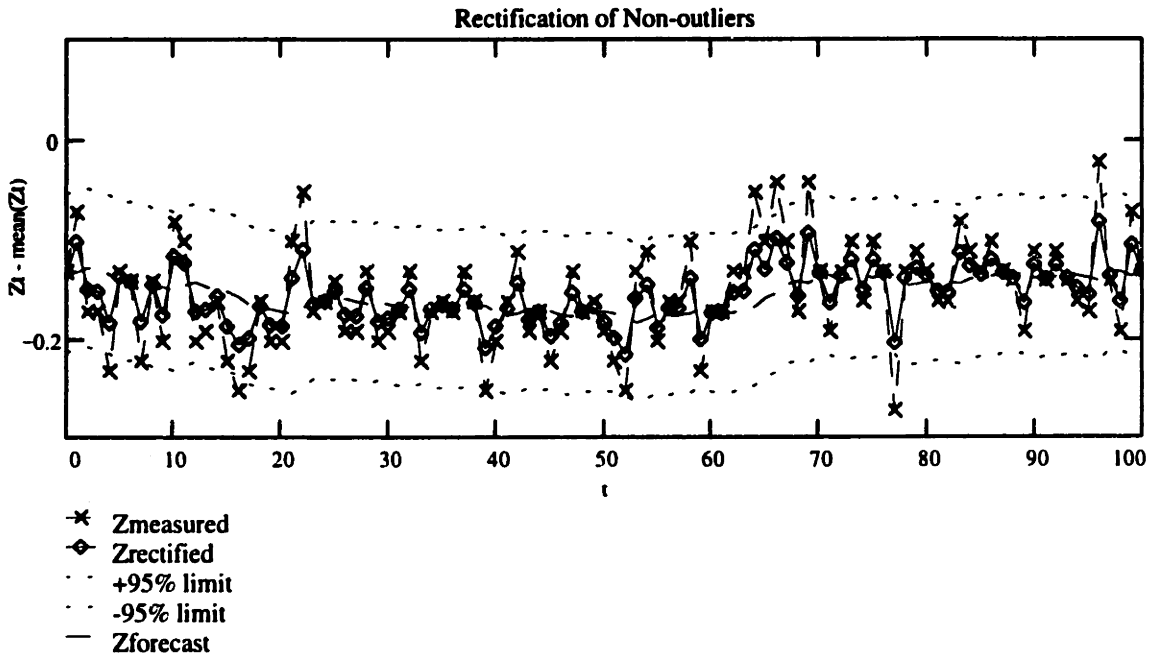
Extract the rectified values.

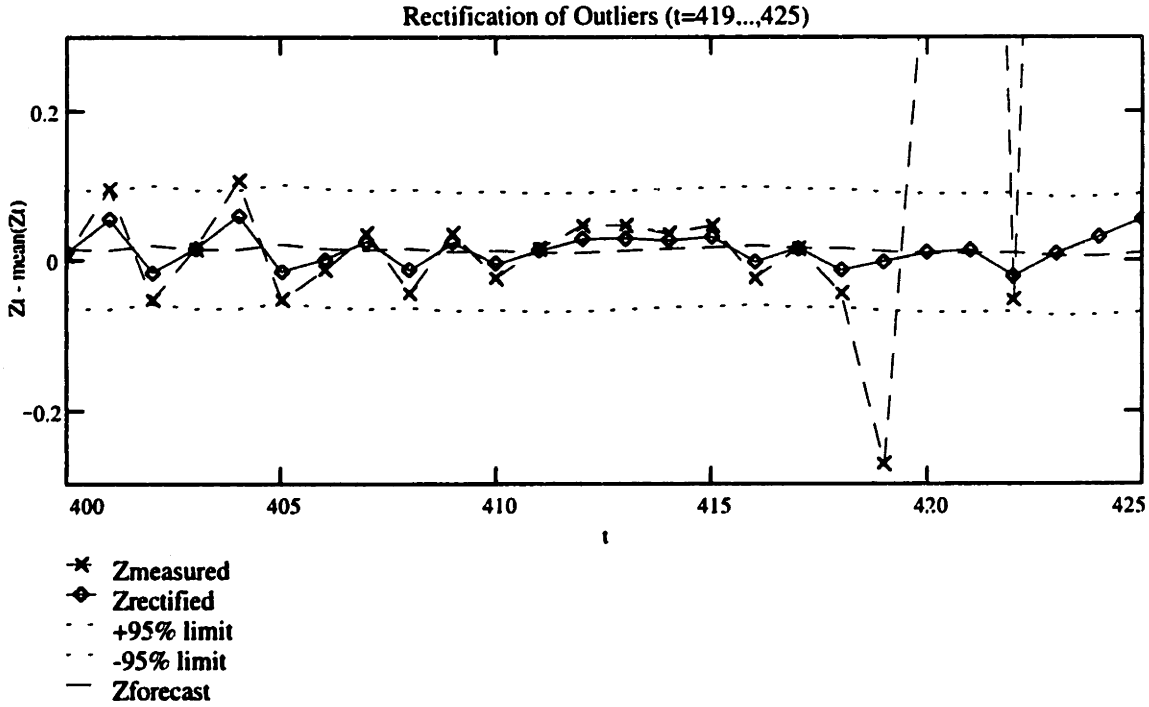
$zr_t = Q_{2\cdot t-1,jmax}$

Outlier indicators:

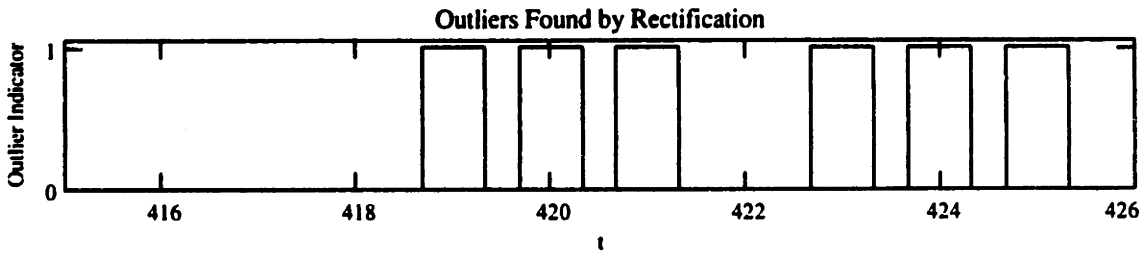
$zeta_t = zi(1, t, zr_t, zf_t)$

The plots of the rectified data show that even the non-outlier observations have been adjusted to be closer to more probable values and the outlier observations have been replaced with acceptable values.





The rectification method correctly identified the six outliers.



Next, the rectified values are analyzed just to see how closely they follow the prior distribution (correlation structure) specified by the prototypical data.

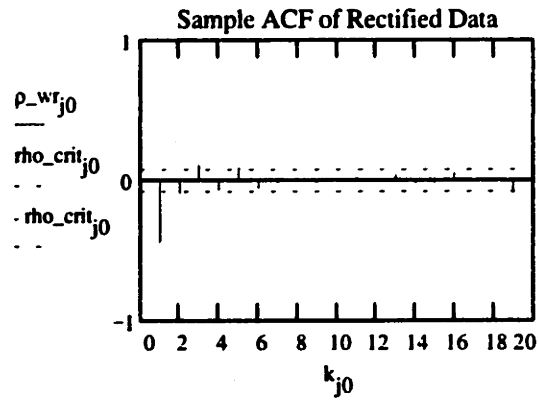
$N2 = 426 \quad i = 0..N2 - 1 \quad wrect_i := if(i, zr_i - zr_{i-1}, zr_0)$

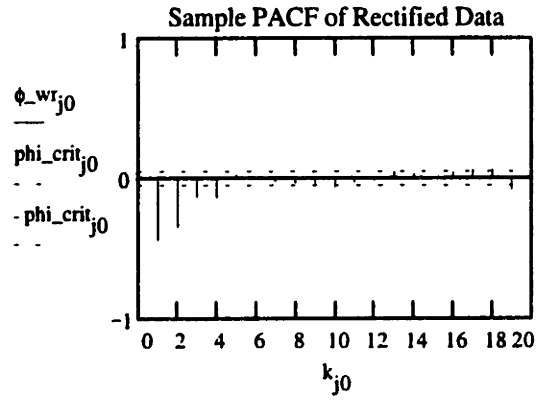
$j0 = 0..3\cdot(N2 - 1) \quad rho\_wr = lcorr(wrect, wrect) \quad rho\_wr_1 = -0.438 \quad phi\_wr = plcorr(wrect$

$rho\_crit_{j0} = t\_crit\_5pct\cdot\left(\frac{1}{\sqrt{N2}}\right) \quad phi\_crit_{j0} = \frac{1}{\sqrt{N2}}$

$\rho\_wr_{j0} = if\left(mod(j0,3), 0, rho\_wr_{\frac{j0}{3}}\right) \quad \phi\_wr_{j0} = if\left(mod(j0,3), 0, phi\_wr_{\frac{j0}{3}}\right)$

$k_{j0} = if\left(mod(j0,3)=1, \frac{j0-1}{3}, if\left(mod(j0,3)=2, \frac{j0-2}{3}+1, \frac{j0}{3}\right)\right)$ <-- (This is done so ACF plots show spikes rather than points connected by lines.)





The correlation structure of the rectified data is similar to that of the original observations. Therefore, we conclude our study.