The Pennsylvania State University
The Graduate School

DATA-DRIVEN PATTERN IDENTIFICATION IN COMPLEX SYSTEMS USING SYMBOLIC DYNAMIC FILTERING

A Dissertation in
Electrical Engineering
by
Chinmay Rao

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2011

The dissertation of Chinmay Rao was reviewed and approved* by the following:

Asok Ray
Distinguished Professor of Mechanical Engineering
Dissertation Advisor, Co-Chair of Committee

W. Kenneth Jenkins
Professor of Electrical Engineering, Head of the Department of Electrical Engineering
Dissertation Co-Advisor, Co-Chair of Committee

Shashi Phoha
Professor of Electrical and Computer Engineering

Jeffrey Mayer
Associate Professor of Electrical Engineering

Joe Horn
Associate Professor of Aerospace Engineering

*Signatures are on file in the Graduate School.

Abstract

Symbolic dynamic filtering (SDF) has been recently reported in literature as a pattern recognition tool for early detection of anomalies (i.e., deviations from the nominal behavior) in complex dynamical systems. Accurate and computationally tractable modeling of such complex system dynamics, solely based on fundamentals of physics, is often infeasible. Hence, it might be necessary to learn the behavior of the system through times series data obtained from sensors. Symbolic dynamics provide a useful tool for time series analysis. Symbolic dynamics attempts to model a continuous time signal by a corresponding symbolized sequence.

This dissertation presents a review of SDF and its performance evaluation relative to other classes of pattern recognition tools, such as Bayesian Filters and Artificial Neural Networks, from the perspectives of: (i) anomaly detection capability, (ii) decision making for failure mitigation and (iii) computational efficiency. The evaluation is based on analysis of time series data generated from a nonlinear active electronic system.

This dissertation also addresses statistical estimation of multiple parameters that may vary simultaneously but slowly relative to the process response in nonlinear dynamical systems. The estimation algorithm is sensor-data-driven and is built upon this concept of SDF for real-time execution on limited-memory platforms, such as local nodes in a sensor network. In this approach, the behavior patterns of the dynamical system are compactly generated as quasi-stationary probability vectors associated with the probabilistic finite-state automata in the symbolic dynamic setting. The estimation algorithm is validated on nonlinear electronic circuits that represent externally excited Duffing and unforced van der Pol systems. It is also evaluated on the NASA C-MAPSS model of an aircraft engine and the simulation of a permanent magnet synchronous motor. Confidence intervals are obtained for statistical estimation of two parameters in these systems.

A framework is also presented for sensor-information fusion. In a complex

system such as an aircraft gas-turbine engine, the patterns generated from a single sensor may not carry sufficient information to identify multiple parameters/faults because different combinations of component faults may generate similar signatures in a particular sensor observation. Low dimensional pattern vectors are identified for the purpose of feature level sensor fusion. The current framework attempts to fuse information from different sensors at the feature level as opposed to the frameworks of data level or decision level fusion.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to firstly thank my advisor, Dr. Asok Ray for his guidance, inspiration and motivation - without which this dissertation would not have been possible. I faced several personal and technical roadblocks through this journey, and his presence at each stage has been invaluable.

I would also like to thank Dr. Kenneth W. Jenkins, my co-advisor. I will especially remember all the healthy discussions that we have shared during the course of my stay at Penn State. I am grateful to my committee members for their inputs - Dr. Shashi Phoha, Dr. Jeffrey Mayer and Dr. Joe Horn.

The contribution of my colleagues and co-authors has been invaluable - Soumik Sarkar, Kushal Mukherjee, Dr. Murat Yasar, Dr. Subhadeep Chakraborty and Dr. Shalabh Gupta each helped in formulating different parts of this work. In addition, my lab mates - Eric, Dheeraj, Xin, Abhishek, Yichen, Ishanu, Andrew, Aparna, Venkatesh all contributed to the excellent learning environment.

I also express my thanks to the sponsors of this research work: The Army Research Laboratory and the Army Research Office under Grant No. W911NF-07-1-0376; Office of Naval Research (ONR) under Grant No. N00014-09-1-0688 and NASA under Cooperative Agreement No. NNX07AK49A.

I would like to express my sincere thanks and deep regards for my father - Dr. Ravi A Rao, mother - Dhvanita Rao and sister Deepna Rao for their support and faith in me at all times.

Finally, I owe everything to my friends in India and the United States, and last but not the least, my incredible roommates and friends in State College who have been a family away from home.

Dedication

I dedicate this thesis to my parents. As a toddler, I kept my father awake while he was writing out his doctoral dissertation. Hopefully, my parents did not loose too much sleep during my journey.

Chapter 1

Introduction

Recent research has extensively explored the development of a signal processing, data mining and pattern identification tool in the framework of Symbolic Dynamic Filtering (SDF) ([2–4]). Of particular interest is the problem of parameter estimation using this symbolic dynamic filtering framework, especially in the context of multiple fault detection and isolation in non-linear dynamical systems.

Complex human engineered systems today are composed of many smaller components, and tractable models are not available for each individual block. Hence, identifying the properties of such a system at a component or parameter level has not received much attention. Nevertheless, when component-level parameter estimation is essential, system identification turns out to be of significant importance. Since most of these systems are highly interconnected, physically, as well as through the use of feedback control loops, a change in a single component is often masked by corrective action taken up by other parts of the system. Thus, the detection and isolation of simultaneously varying parameters and estimation of the magnitude of these variations pose a challenging problem.

There are several non-linear parameter estimation techniques available, such as neural networks, which are capable of performing to a reasonable degree of satisfaction. However, it is recognized that, system identification and parameter estimation in a single component is just a small part of the problem in its entirety, and in the setting of the bigger problem, complex algorithms and optimization techniques such as neural networks have several inherent drawbacks. The general framework described in this dissertation is applicable to a variety of perti-

nent problems, ranging from monitoring and localizing terrorist threats in complex urban environments to detection of off-nominal behavior in mechanical systems. Fortunately, the related issues of the problem, can be explained, without loss of generality, with examples. In this context, several examples are presented in this dissertation, ranging from an electronic circuit based on the Duffing equation to a simulation model of a multi-component aircraft engine.

This chapter is organized into four sections including this one. Section 1.1 presents the motivation and background for the research in this dissertation. Section 1.2 surveys existing literature in symbolic dynamics. It also provides a review of methodologies for parameter estimation and sensor selection. Section 1.3 describes the contributions made in this thesis. The organization of the dissertation is presented in Section 1.4.

1.1 Motivation and Background

Recent literature has reported various methods for estimation of multiple parameters, such as those based on joint state estimation [5], parity equations [6], generalized likelihood ratio [7], Karhunen-Loéve and Galerkin multiple shooting [8], similarity measures [9], and orthogonal Haar transform [10]. Gupta et al have presented [11] an application of parameter estimation is the detection and mitigation of evolving faults in interconnected dynamical systems. Often evolution of gradual deviations from the nominal behavior in individual components of such systems may lead to cascaded faults because of strong input-output and feedback interconnections between the system components. Such faults may eventually cause catastrophic failures and forced shutdown of the entire system. In such a scenario, the problem of degradation monitoring of the system reduces to simultaneous estimation of several slowly-varying critical parameters.

An important issue in the study of any natural or human-engineered complex system is primarily whether the dynamics of the underlying process can be adequately described by a mathematical model whose solution procedure for the process variables is simple, elegant and computationally tractable. Complex behavior emerging from high dimensionality of the phase space, the presence of uncertain chaotic orbits [12], nonlinear stochastic processes, and random noisy excitation

often restricts the applications of the fundamental laws of physics to accurately determine a dynamical model of such systems [13]. Specifically, sole reliance on physics-based modeling for identification of behavioral patterns in complex systems has been found to be infeasible because of the difficulties in achieving requisite accuracy and precision of the nonlinear spatio-temporal stochastic models [14]. Despite these difficulties, the key problem - identification of statistical patterns, which constitute the dynamical characteristics of the system, can be formulated in terms of observation-based estimation of the process variables. In other words, in the absence of a feasible mathematical model, the inherent dynamics of a complex nonlinear system can be inferred from time series data generated from a network of spatially distributed multiple sensing devices which are sensitive enough to capture the essential details of the dynamics of the system.

The above discussions evince the fact that behavioral information can be extracted from the response of process variable(s). However, the analysis of observed time series data for pattern identification is often difficult because of the underlying complexity of the process and also due to disturbances from noisy environments. As such, information-based inference of the underlying process becomes a formidable task. Such complexity issues are of prime concern in engineering applications such as weather forecasting, structural health monitoring, signal processing, system identification and adaptive control, which have motivated the study of dynamical systems [15] from the perspectives of Statistical Mechanics.

Accurate and computationally tractable modeling of complex system dynamics based solely on the fundamental principles of physics is often infeasible. Hence, it might be necessary to rely on time series data generated from sensors and other sources of information. The need to extract relevant information about the observed dynamics has lead to development of Nonlinear Time Series Analysis (NTSA) techniques [16]. Analysis of dynamical systems using NTSA techniques is classified into two areas [17]:

- Behavior identification
- Modeling and Prediction

Identification of nonlinear systems can be achieved using Formal Languages [18]. The first step in this process often involves converting the raw time-series measure-

ments, or carefully chosen wavelet domain signals into a corresponding sequence of symbols. The symbol sequence is then treated as a transform of the original data that retains much of the important temporal information. Therefore partitioning of data to create symbols is a crucial aspect of symbolic analysis. Partitions are created based on information content in the data.

An important practical advantage of working with symbols is increased computational efficiency. This feature is important for real-time monitoring and control applications. Also, analysis of symbolic data is often less sensitive to measurement noise. In some cases, symbolization can be accomplished directly in the instrument by appropriate design of the sensing elements. Such low-resolution (even disposable) sensors combined with appropriate analysis can significantly reduce instrumentation cost and complexity. Applications of symbolic methods are thus favored in circumstances where robustness to noise, speed, and/or cost are paramount [19]. A part of this dissertation is aimed at showing a comparison between symbolic domain filtering and other modern approaches such as Bayesian Filtering and statistical pattern recognition tools.

Anomaly monitoring in complex systems is formulated as a solution of two interrelated problems:

1. The forward (analysis) problem - The primary objective of the forward problem is identification of the statistical changes in the time series data of ultrasonic signals due to gradual evolution of fatigue damage.

2. The inverse(synthesis) problem- A major objective of the inverse problem is to infer the anomalies, and to estimate parameters for forecasting impending failures. Another objective could be to provide the estimates of the remaining useful life from the observed time series data in real time based on the information generated during the forward problem.

Statistical patterns of parameter evolution are generated offline during the forward problem using symbolic time series analysis of sensor data. The patterns obtained here are used in the inverse problem to obtain (possibly online) the estimates of the parameters of the system under test.

1.2 Literature Survey

This section is divided into three subsections. The first subsection presents a background to the topic of symbolic dynamics. The second subsection discusses different methods employed to study non-linear systems for the purposes of parameter identification and modeling. The third subsection reviews methods used for fault detection and isolation, and also discusses the challenges posed for the optimal selection of sensors.

1.2.1 Symbolic Dynamics Background

An aim of this dissertation is to investigate a relationship between classical symbolic dynamics and symbolic time series analysis for pattern identification in complex systems. A theoretical framework is presented and the methodology is formulated in terms of analogy between obtaining unique patterns for different values of system parameters and the generating partitions in statistical mechanics [20]. This analogy is a step towards development of a detailed statistical mechanical formalism of pattern identification in complex systems.

Symbolic dynamics is the practice of modeling a dynamical system by a discrete space consisting of infinite sequences of abstract symbols, each of which corresponds to a state of the system, with the evolution given by the shift operator [2]. The basic idea is to take the state space of any system and divide (partition) it into a finite number of regions, each of which is labeled with a symbol. A point in the state space then gives rise to an infinite sequence of symbols: the symbol for the cell of the partition of the original point, the symbol for the cell of its first iterate, its second, and so forth. This naturally involves a loss of information, due to the transition from continuous to discrete values, which is referred to as coarse graining. An aim of this dissertation is to study this loss of information at various stages, and also apply the techniques of symbolic dynamics to engineering applications.

The field of symbolic dynamics has connections with mathematics, physics and engineering. The different emphasis of the applications in the different disciplines have been summarized by Tuffilaro [21]. In a mathematical setting, the goal is to identify generating partitions on the phase space for chaotic systems. A def-

inite $1-1$ map is desired between the symbol space and the original dynamic system, usually consisting of discrete differential equations. In physics, the goal is somewhat relaxed to creating generating partitions within an experimental error bound. Also, there is an emphasis on modeling the underlying system for the purposes of tasks such as state identification. These goals are further relaxed in an engineering study, where a simple and robust map is generated for the reduction and compression of data. While the primary goal of this dissertation is to employ the underlying method for current engineering challenges, an important aspect is also to explore the connection between the physics and engineering methodologies of symbolic dynamics.

Analysis of dynamical systems through symbolic dynamics consists of two steps: i) Symbol Generation, ii) Modeling of Symbol Sequences. To set up the symbolic dynamics of this system, we must first define a partition. Informally, a partition is the separation of phase space into disjoint regions. This partition is the first step to go from a continuous description of a physical process to a discrete description composed of a finite (usually just a few) symbols. One of the first goals of symbolic dynamics is to understand the connection between continuous systems and discrete systems with typically a small alphabet. A generating partition is one where there is a one-to-one correspondence between the continuous states and the symbol sequences generated. In these cases, studying the symbolic dynamics is completely equivalent to studying the original dynamics.

Unfortunately, there is no satisfactory general theory for finding a generating partition for every case. The exception is one dimensional maps (eg. the Logistic map), where partitions are made at the critical points (minima, maxima, or discontinuities). Davidchack et al [22] had proposed a partition algorithm which successively colors unstable periodic orbits (UPOs) to ensure unique codings (all UPOs have unique codes under a generating partition). This is practical only if the dynamics are already known, since the necessary high-order UPOs are very difficult to obtain from observed data alone.

A practical method to obtain good partitions from observed data was suggested by Kennel and Buhl [23]. This technique is referred to as Symbolic False Nearest Neighbors ($SFNN$). The criterion for a good partition in SFNN is defined as: short sequences of consecutive symbols should localize the corresponding continu-

ous state space point as well as possible. This is achieved by forming a geometrical embedding of the symbol sequence under the candidate partition and minimize a statistic which quantifies the apparent errors in localizing state space points. The nearest neighbor to each point in the embedding is found in terms of Euclidean distance of symbolic neighbors. In general, better partitions yield a smaller proportion of symbolic false nearest neighbors. For convenience of implementation, the partitions are parameterized with a relatively small number of free parameters. This is accomplished by defining the partitions with respect to a set of radial-basis influence functions. The statistic for symbolic false nearest neighbors is minimized over these free parameters. However, this partitioning method may become computationally very inefficient when the dimension of the phase space is large or if the data set is contaminated by noise.

In this dissertation, the Maximum Entropy (ME) and Uniform partitioning schemes are utilized to generate symbols. An objective of these partitioning schemes is to create partitions based on the information content in the data. The concept of maximum entropy partitioning is that regions with more information are partitioned finer and those with sparse information are partitioned coarser. Also with no prior knowledge about the data, it would be prudent to ensure that all symbols are equally distributed. This approach is particularly effective when information about the system is only available at its nominal condition. The chances of detection of any of the underlying (multiple) parameters are enhanced when all symbol probabilities are equally sensitive to change. The uniform distribution of symbols maximizes the entropy and hence the name Maximum Entropy (ME) partitioning. Another approach involves partitioning the phase space into equal blocks. This approach is called Uniform partitioning, and provides a computationally simple means of determining the physical locations of the partitions.

The next step in the process is to model the characteristics of this symbol sequence. In general, a dynamical system may allow only certain concatenations of symbols to occur as there are many illegal (i.e., physically impossible) sequences. Among the legal symbol sequences, some symbols may occur more frequently than others. Therefore, representing symbol sequences with a probabilistic model is very beneficial in studying their characteristics.

Probabilistic Finite State Automata ($PFSA$) provide a compact representation

to symbol sequences. Finite state automata may be constructed from symbol sequences in different ways. One approach, proposed by Crutchfield and Young [24], is called the ϵ-machine. This approach had several shortcomings, such as lack of a systematic procedure for choosing algorithm parameters and slow convergence rates. Sometimes, it may return non-deterministic causal states. Shalizi proposed a different method for constructing ϵ-machines [25]. This algorithm is called Causal State Splitting Reconstruction ($CSSR$) and is based on state splitting instead of state merging. The CSSR algorithm starts with a simple model for the symbolic process and elaborates the model components only when statistically justified. Initially, the algorithm assumes the process to be independent and identically distributed. This can be represented by a single causal state and hence zero statistical complexity and high entropy rate. $CSSR$ then uses statistical tests to determine whether new states should be added to the model. The addition of new states increases the estimated complexity, while lowering the entropy rate. The CSSR algorithm suffers from one drawback. It does not provide a guide for selecting the length of the longest history to be considered, which is left to discretion. Hence an inaccurate model will be obtained if the maximum length chosen is less than the inherent memory of the process being modeled. In this dissertation, the D-Markov machine introduced by Ray [2] is utilized for modeling symbol sequences. An advantage of D-Markov technique is its simplicity of construction. This dissertation also provides a procedure for selecting the parameters associated with D-Markov machine.

1.2.2 Parameter Identification in non-linear systems

Parameter identification in non-linear systems has been a subject of investigation with various methods. One of the early approaches provided in [26] utilizes statistical linearization of non-linear systems for parameter estimation. The nonlinearity is approximated by a series of functions. The parameters of these functions are determined by correlation techniques. This method is applicable even to systems where the nonlinearity cannot be separated from the dynamics. However, the estimates are significantly affected by the presence of noise. When the input is noisy, small changes in covariance had a significant influence on the estimate. Also, the

presence of noise in the output tends to bias the estimate. Other techniques used for parameter estimation include Kalman Filter and its variants, Particle Filtering, Genetic Algorithms, Probabilistic Methods and Wavelets.

Some traditional methods for parameter estimation in non-linear systems are briefly described below.

1.2.2.1 Model based methods

Many conventional methods of parameter estimation are model-based. A lot of recent research has focused on developing linear and non-linear models for complex engineering systems. Techniques used for parameter estimation include Bayesian methods such as the Kalman Filter and its variants [27], Particle Filtering [28], Genetic Algorithms, Probabilistic methods and Wavelets.

Several methods based on the Kalman filter have been developed over the years for state and parameter estimation in non-linear systems. The Kalman Filter is limited to estimation in linear systems. Since most systems are nonlinear, some assumptions involved in Kalman Filtering were relaxed to deal with nonlinearities. In the Extended Kalman Filter (EKF), the state transition and observation models need not be linear functions of the state but may instead be nonlinear (differentiable) functions. Since these functions cannot be directly applied to covariance, Jacobians of these functions are used. At each time step Jacobians are evaluated with current predicted states. This process essentially linearizes the non-linear function around the current estimate. It is important to note that the EKF is not an optimal filter. Moreover, the filter gain cannot be computed off-line as in the Kalman filter, since it depends on previous measurements. Contrary to the Kalman filter, the EKF may diverge, if the consecutive linearizations are not a good approximation of the nonlinear model.

The Unscented Kalman Filter (UKF), addresses this problem by using a deterministic sampling approach. The state distribution is again approximated by a Gaussian distribution, but is now represented using a minimal set of carefully chosen sample points called sigma points. These sigma points completely capture the true mean and covariance, and when propagated through the nonlinear system, captures the posterior mean and covariance accurately to the 3rd order Taylor series expansion for any nonlinearity. This technique removes the requirement to

analytically calculate Jacobians, which for complex functions can be a difficult task in itself.

Particle Filtering (PF) is another widely used method for estimation in non-linear systems. Particle Filtering is a sequential Monte Carlo Markov method. In essence, the particle filter can be interpreted as a large number of simulations, where each simulation consists of a sample from the distribution that is to be estimated. There is a weight associated with each sample, which corresponds to how likely the sample is. These samples together with the corresponding weights constitute a discrete approximation of the posterior distribution.

1.2.2.2 Data driven methods for parameter estimation

For many systems, first principle models (also known as white-box models) are not available. A black-box model is a system for which there is no prior information available. These are data-driven or regressive models, for which the functional form of relationships between variables and the numerical parameters in those functions are unknown and need to be estimated. The best known data-driven techniques are hidden markov models (HMM), artificial neural networks (ANN) and genetic programming approaches (GP).

Artificial neural networks use highly simplified models composed of neurons connected together by links of variable weights to form a black-box representation of the system. These models are trained using sets of input and output data and 'learn' complex model functions from examples. The principal advantage of this technique is that ANNs have the ability to model complex, non-linear processes without any assumptions between the input and output variables. The main problems faced in this method are to determine the a-priori structure of the neural network, and over-fitting. Also, any a-priori information known about the system cannot be easily incorporated into the neural network.

Genetic programming is another modeling approach currently in vogue. Genetic algorithms imitate the natural selection process to obtain a solution and the goodness of the solutions improves through successive generations. A frequently used method in this regime is known as symbolic regression proposed by Koza. This technique generates a mathematical expression to fit a set of data points using evolutionary genetic programming. An advantage of this method is that the

user can resolve the information required on the system behavior. This gives an insight into the relationship between input and output data.

Model based approaches discussed above are often inadequate for human-engineered complex systems due to unavailability of a reliable model of the process dynamics. A major drawback of the data driven methods described above is that their principal aim is to reconstruct the model from the data. In doing so, the functions they produce tend to grow in length and complexity over time. The method presented in this dissertation, Symbolic dynamic filtering (SDF) [2,29] is based on the concept of symbolic time series analysis ($STSA$) [19]; SDF belongs to the class of data-driven statistical pattern recognition and enables compression of information into pattern vectors of low dimension for real-time execution on limited-memory platforms, such as small microprocessors in a sensor network. This dissertation shows the performance of SDF to be superior to that of several pattern classification techniques such as principal component analysis (PCA), artificial neural networks (ANN), kernel regression analysis (KRA), particle filtering (PF) and unscented Kalman filtering (UKF), in terms of early detection of changes, computation speed and memory requirements.

As an extension of the parameter estimation method of Tang et al. [30], which is also based on $STSA$, Piccardi [31] proposed a multiple-parameter estimation scheme in chaotic systems. Although the symbolic analysis was performed on a probabilistic finite-state model, the parameter vector was estimated by a genetic algorithm based optimization in a deterministic setting. The present dissertation, which is built upon the concept of SDF, presents a more engineering and statistics based approach to estimation of multiple parameters in a more stochastic setting in the presence of sensor and process noise.

Repeated diagnosis on complex dynamical systems may often be computationally prohibitive due to expensive simulation requirements. Fault dictionaries are sometimes used to alleviate this problem, but they may be unfeasible to store because of their large sizes [32], and more importantly, they typically provide only a black box view of the system and hence almost no diagnostic flexibility. The problem also occurs because dictionaries usually only store primary output information.

1.2.3 Sensor-data-fusion and optimum sensor selection

Several data-driven techniques have been reported in literature for fault detection, diagnosis and prognosis. These technqiues include statistical linearization [33], Kalman filtering [27], Unscented Kalman filtering (UKF) [34,35], Particle Filtering (PF) [28], Markov Chain Monte Carlo (MCMC) [36], Bayesian Networks [37], Artificial Neural Networks (ANN) [38], Maximum Likelihood Estimation (MLE) [39], Wavelet-based tools [40], and Genetic Algorithms (GA) [41]. However, system identification and anomaly detection in a single component is just a small part of the health monitoring problem in its entirety. In the setting of a more complex problem of fault detection in multiple components under changing input/operating conditions, the underlying algorithms and associated optimization techniques may have certain drawbacks. For example, computationally expensive algorithms may not be suitable for engineering systems like commercial-scale transport aircraft, where on-board health monitoring is needed to ensure flight safety. Furthermore, online ground-based vehicle control requires data communications over wireless sensor networks, where it is essential to compress packets without any significant loss of information.

The propulsion systems of modern aircraft engines are comprised of a large number of complex sub-components where fault detection and health monitoring at both component and system levels are issues of paramount importance. The inherent complexity and uncertainties in these systems pose a challenging problem because pertinent first-principle models are usually unavailable or are oversimplified as lumped parameter models. Therefore, in the absence of high fidelity models, the major challenge is fault detection by developing a description of the component dynamics primarily from the input output characteristics. These decisions of fault detection should not only be sensitive to changes in the parameters of the actual dynamical system but also be invariant to changes in the operating or input conditions.

The problem of anomaly detection using symbolic dynamic filtering (SDF) [2, 3, 42] with applications to fault diagnosis in aircraft engines [11, 43] has been addressed; however, the issues of system identification have not been addressed in SDF-based fault detection. The aircraft engine system is an example of a complex system where the components are interconnected physically as well as through

feedback control loops, and thus the effects of degradation even in a single component may affect the input streams to the remaining components. Furthermore, in many situations (e.g., tactical aircraft in military applications), the system might need to be operated in different regimes and under diverse operating conditions.

To address the above-mentioned issues and achieve the associated objectives, this dissertation develops a robust and computationally inexpensive system identification technique that is built upon formal-language-theoretic formulation, based on the symbolic information. Specifically, the algorithms are designed to be robust with respect to sensor noise and, at the same time, simple enough to be embedded in the sensors themselves. This method would also facilitate construction of a reliable sensor network to serve as a backbone to higher levels in the decision-making hierarchy of large-scale complex systems (e.g., communication and control of aircraft's vehicle health and energy management system).

The real-time fault detection method, developed in this disseratation, has been validated on a test-bed that is built upon the NASA Commercial Modular Aero Propulsion System Simulation (C-MAPSS) model. This facility is particularly relevant for testing and validation of condition-monitoring algorithms as it allows the users to choose and design operational profiles (e.g., thrust levels), controllers, and environmental conditions to simulate scenarios of interest. Most importantly, the test facility allows the users to tune efficiency and flow parameters to simulate specific fault modes.

1.3 Objectives and Contributions

The objective of this research was to develop a system identification strategy which can make accurate and reliable assessment of the the parameters of a dynamical system, and can provide statistical bounds for these parameters. This work focuses on the development of a method for analyzing complex systems which can be utilized over a sensor network, and can be implemented efficiently with respect to memory requirements and processing needs. A central step in this kind of identification methodology is discretizing the raw time-series measurements from multiple sensors into a corresponding sequence of symbols. From these perspectives, the contributions of the thesis can be summarized as follows:

1. Assessment of Symbolic Dynamic Filtering in terms of information capturing, memory requirements and processing time. The Symbolic Dynamic Filter is compared to several prevailing methods such as the Particle filter, unscented Kalman filter, neural networks, Principal Component Analysis (PCA) and Kernel Regression Analysis. The test bed used for this comparison is the classical Duffing equation that has been implemented on an electronic circuit. The superior performance of SDF is one of the key motivations to pursue further applications of this technique to tackle different engineering problems.

2. The operating conditions of the Symbolic Dynamic Filter have been presented in the framework of Information theory. Specifically, the choice of depth, number of symbols, alphabet size, wavelet etc. are validated using classical concepts like entropy and information gain. A conflicting objective optimization technique is used to determine ideal choices of these values.

3. Development of a scheme using Symbolic Dynamics for classifying multiple parameters in complex systems. A statistical formalism has been presented for this method. An important contribution of this method lies in identifying and developing a mapping from a parameter space to a feature vector space that captures the changes in the given parameters.

4. Applications to the problem of resource allocation in sensor networks have been identified. A information theoretic metric is proposed to determine the relative significance of data generated from one sensor at particular location with respect to another sensor. This directly relates to the field of sensor selection and fusion.

5. The results of the theoretical research have been validated by demonstration on the following test beds: The Duffing and Van der Pol equations simulated on an electronic circuit. A model of a NASA turbo-fan engine. A simulation of a Permanent Magnet Synchronous Motor ($PMSM$). The problem of multiple parameter estimation is formulated for each of these frameworks and solved for different test cases; confidence intervals are presented for each estimate obtained.

The methods of parameter estimation and sensor selection developed are designed to be robust with respect to sensor noise, and simple enough to be implemented in mobile platforms or even embedded in the sensors themselves, thus producing a reliable monitoring network.

1.4 Organization

In addition to this chapter the rest of the dissertation has been divided into the following sections:

- In chapter 2, a brief introduction to Symbolic Dynamic Filtering is provided. A study is presented that outlines the selection of various parameters of the Symbolic Dynamic Filter. Also, this method is compared with other commonly used methods for parameter estimation and fault detection, and the superiority is validated.

- Chapter 3 presents the framework for the statistical estimation of multiple parameters using Symbolic Dynamic Filtering. The problem is also formulated as a multi-class classification problem, and possible solutions are presented.

- Chapter 4 presents a sensor selection technique in the framework of Symbolic Dynamic Filtering. The forward and inverse problems are framed in a multiple sensor scenario. Also, a method is presented to select an optimum subset of sensors.

- In chapter 5, the test beds used in the thesis are elaborated. These are the Duffing and Van der Pol equations, a model of a gas turbine engine, and the simulation of a Permanent Magnet Synchronous Motor.

- In chapter 6, the results of the various methods described above have been presented.

- In the last chapter, the conclusions of various methods presented in this dissertation are drawn, and directions for future research are discussed.

- Four appendices have been added. Appendix A describes various comparative anomaly detection algorithms using Bayesian and Statistical tools. Appendix B describes the electronic circuit used to generate the Duffing and Van der Pol equations. Appendix C outlines several techniques used for multiclass classification. Appendix D presents a thermodynamics based viewpoint for the estimation of multiple parameters using escort probabilities.

Chapter 2

Comparative Evaluation of Symbolic Dynamic Filtering: Forward and inverse problem

2.1 Introduction

The major objective of this chapter is to introduce and evaluate Symbolic Dynamic Filtering (SDF) with other pattern recognition methods such as Bayesian Filtering (BF), which is both model-based and dynamic data-driven, and is capable of detecting parametric or non-parametric changes in the model. The Kalman (Extended Kalman) Filter [44] is often adequate for linear (linearized) systems, but it may fail to capture the dynamics of a nonlinear system, specifically with non-additive uncertainties [45]. Recent literature has reported Monte Carlo Markov Chain ($MCMC$) techniques, such as Particle Filtering [46] and Sigma Point techniques such as Unscented Kalman filtering [35] that yield numerical solutions to Bayesian state estimation problems and have been applied for anomaly detection in nonlinear dynamical systems [47]. In addition to BF, this thesis investigates other classes of well-known pattern recognition tools such as Artificial Neural Networks (ANN), Principal Component Analysis (PCA), and Kernel Regression Analysis (KRA) for pattern change detection [48]. In the class of ANN, mutilayer perceptron [49] and radial basis function [50] configurations have been widely used for

detection of anomalous patterns, and PCA [51] and KRA [52] are also commonly used for data-driven pattern recognition. These pattern recognition tools have been evaluated for comparison with SDF from the following perspectives.

- Performance evaluation in terms of quality of anomaly detection (e.g., enhanced detection capability and reduced rate of false alarm)
- Decision making for mitigation of forthcoming failures
- Computational efficiency in terms of execution time and memory requirements

2.2 Review of Symbolic Dynamic Filtering

The theory of symbolic dynamic filtering (SDF) for time series data analysis is built upon the underlying principles of *Nonlinear Dynamics* [53], *Symbolic Dynamics* [15], *Information Theory* [54], and *Statistical Pattern Recognition* [48]. This chapter presents the underlying concepts and salient features of symbolic dynamic filtering (SDF) for anomaly detection in complex dynamical systems. While the details are reported as pieces of information in previous publications [2,3,42,55,56], the essential concepts of space partitioning, symbol generation, and construction of a finite-state machine from the generated symbol sequence are succinctly explained in this section for completeness.

Detection of anomaly patterns is formulated as a two-time-scale problem. The *fast time scale* is related to response time of the process dynamics. Over the span of a given time series data sequence, dynamic behavior of the system is assumed to remain invariant, i.e., the process is quasi-stationary at the fast time scale. In other words, the variations in the behavior of system dynamics is assumed to be negligible on the fast time scale. The *slow time scale* is related to the time span over which parametric or non-parametric changes may occur and exhibit non-stationary dynamics. The concept of two time scales is illustrated in Fig. 2.1.

An observable non-stationary behavior of the system dynamics can be associated with the anomalies evolving at a slow time scale. In general, a long time span in the fast time scale is a tiny (i.e., several order of magnitude smaller) interval

Figure 2.1. Pictorial view of the two time scales: (i) *Slow time scale* of anomaly evolution and (ii) *Fast time scale* for data acquisition and signal conditioning

in the slow time scale. For example, evolution of anomalies (causing a detectable change in the system dynamics) may occur on the slow time scale in the order of hundreds of hours of operation; in contrast, the process dynamics may remain essentially invariant on the fast time scale in the order of seconds. Nevertheless, the notion of fast and slow time scales is dependent on the specific application, loading conditions and operating environment. From the perspective of anomaly pattern detection, time series data sets are collected on the fast time scale at different slow time epochs separated by uniform or non-uniform intervals.

The continuously varying process of system dynamics is often modelled as a finite-dimensional dynamical system in the setting of an initial value problem as:

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \theta(t_s);\ \mathbf{x}(0) = \mathbf{x}_0, \tag{2.1}$$

where $t \in [0, \infty)$ denotes the (fast-scale) time; $\mathbf{x} \in \mathbb{R}^n$ is the state vector in the phase space; and $\theta \in \mathbb{R}^\ell$ is the (possibly anomalous) parameter vector varying in (slow-scale) time t_s. Sole usage of the model in Eq. (2.1) may not always be feasible due to parametric and non-parametric uncertainties and noise. A convenient way of learning the dynamical behavior is to rely on the additional information provided by (sensor-based and/or model-based) time series data [20] [57], as described in the following subsections.

Figure 2.2. Concept of Symbolic Dynamic Filtering

2.2.1 Symbolic Dynamics, Encoding, and State Machine

This subsection briefly describes the concepts of *Symbolic Dynamics*, encoding nonlinear system dynamics from observed time series data, and state machine construction for generation of symbol sequences. It also presents a procedure for online computation of the machine state probability vectors that are representatives of the evolving patterns of the system's dynamical characteristics.

Let $\Omega \in \mathbb{R}^n$ be a compact (i.e., closed and bounded) region, within which the trajectory of the dynamical system, governed by Eq. (2.1), is circumscribed as illustrated in Fig. 2.2. The region Ω is partitioned into a finite number of (mutually exclusive and exhaustive) cells, so as to obtain a coordinate grid. Let the cell, visited by the trajectory at a time instant, be denoted as a random variable taking a symbol value from the alphabet Σ. An orbit of the dynamical system is described by the time series data as $\{x_0, x_1, \cdots, x_k, \cdots\}$ with $x_i \in \Omega$, which passes through or touches one of the cells of the partition. Each initial state $x_0 \in \Omega$ generates a sequence of symbols defined by a mapping from the phase space into the symbol space as:

Figure 2.3. An Example of Space Partitioning

$$x_0 \rightarrow s_0 s_1 s_2 \cdots s_k \cdots \tag{2.2}$$

where each $s_i,\ i = 0, 1, \cdots$ takes a symbol from the alphabet Σ.

The mapping in Eq. (2.2) is called *Symbolic Dynamics* as it attributes a legal (i.e., physically admissible) sequence of symbols to the system dynamics starting from an initial state. (Note: A symbol alphabet Σ is called a generating partition of the phase space Ω if every legal sequence of symbols uniquely determines a specific initial condition x_0, i.e., every symbolic orbit uniquely identifies one continuous space orbit.) Figure 2.2 pictorially elucidates the concepts of partitioning a finite region of the phase space and the mapping from the partitioned space into the symbol alphabet. This represents a spatial and temporal discretization of the system dynamics defined by the trajectories. Figure 2.2 also shows conversion of the symbol sequence into a finite-state machine as explained in the following subsections.

Symbolic dynamics can be viewed as coarse graining of the phase space, which is subjected to (possible) loss of information resulting from granular imprecision of partitioning boxes. However, the essential robust features (e.g., periodicity and chaotic behavior of an orbit) need to be preserved in the symbol sequences through an appropriate partitioning of the phase space [18] [19].

2.2.2 Space Partitioning

A crucial step in SDF is partitioning of the phase space for symbol sequence generation [19]. Several partitioning techniques have been reported in literature for symbol generation [23], primarily based on symbolic false nearest neighbors ($SFNN$), which may become cumbersome and extremely computation-intensive if the dimension of the phase space is large. Moreover, if the time series data is noise-corrupted, then the symbolic false neighbors would rapidly grow in number and require a large symbol alphabet to capture the pertinent information on the system dynamics. Therefore, symbolic sequences as representations of the system dynamics should be generated by alternative methods because phase-space partitioning might prove to be a difficult task in the case of high dimensions and presence of noise. The wavelet transform [58] largely alleviates these shortcomings and is particularly effective with noisy data from high-dimensional dynamical systems [56]. A comparison of wavelet partitioning and other partitioning methods, such as $SFNN$, is reported in recent literature [3], where wavelet partitioning has been shown to yield comparable performance with several orders of magnitude smaller execution time. This feature is very important for real-time detection of anomaly patterns.

In wavelet-based partitioning, the time series data are first converted to wavelet domain, where wavelet coefficients are generated at different time shifts. The wavelet space is then partitioned with alphabet size $|\Sigma|$ into segments of coefficients on the ordinate separated by horizontal lines. In the illustrative example of Fig. 2.3, the partitioning has been done to create $|\Sigma| = 10$ cells (i.e., intervals along the ordinate in this case). The choice of $|\Sigma|$ depends on specific experiments, noise level and also the available computation power. A large *alphabet* may be noise-sensitive while a small alphabet could miss the details of signal dynamics.

Once the partitioning is done with alphabet size $|\Sigma|$ at the nominal condition (time epoch t_0), it is kept constant for all (slow time) epochs $\{t_1, t_2, ...t_k....\}$, i.e. the structure of the partition is fixed at the nominal condition. Therefore, the partitioning structure generated at the nominal condition serve as the reference frame for data analysis at subsequent slow time epochs.

2.2.3 State Machine Construction

The partitioning (see Fig. 2.2) is performed at the slow time epoch t_0 of the nominal condition that is chosen to be the healthy state having zero anomaly measure. A finite state machine is then constructed, where the states of the machine are defined corresponding to a given *alphabet* set Σ and window length D. The alphabet size $|\Sigma|$ is the total number of partition segments while the window length D is the length of consecutive symbol words [2], which are chosen as all possible words of length D from the symbol sequence. Each state belongs to an equivalence class of symbol words of length D or more, which is characterized by a word of length D at the leading edge. Therefore, the number n of such equivalence classes (i.e., states) is less than or equal to the total permutations of the alphabet symbols within words of length D. That is, $n \leq |\Sigma|^D$; some of the states may be forbidden with zero probability of occurrence. For example, if $\Sigma = \{0, 1\}$, i.e., $|\Sigma| = 2$ and if $D = 2$, then the number of states is $n \leq |\Sigma|^D = 4$; and the possible states are $00, 01, 10$, and 11, as shown in Fig. 2.4.

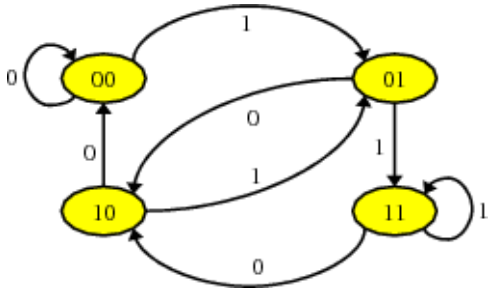


Figure 2.4. Example of Finite State Machine with D=2 and $\Sigma = \{0, 1\}$

The choice of $|\Sigma|$ and D depends on specific experiments, noise level and also the available computation power. A large *alphabet* may be noise-sensitive and a small alphabet could miss the details of signal dynamics. Similarly, while a larger value of D is more sensitive to signal distortion, it would create a much larger number of states requiring more computation power.

Using the symbol sequence generated from the time series data, the state machine is constructed on the principle of sliding block codes [15]. The window of length D on the symbol sequence $\ldots \sigma_{i_1}\ \sigma_{i_2} \ldots \sigma_{i_k} \ldots$ is shifted to the right by one symbol, such that it retains the last (D-1) symbols of the previous state and

appends it with the new symbol σ_{i_ℓ} at the end. The symbolic permutation in the current window gives rise to a new state. The machine constructed in this fashion is called the D-Markov machine [2], because of its Markov properties. A symbolic stationary process is called D-Markov if the probability of the next symbol depends only on the previous D symbols, i.e., $P\left(\sigma_{i_0}|\sigma_{i_{-1}}....\sigma_{i_{-D}}\sigma_{i_{-D-1}}....\right) = P\left(\sigma_{i_0}|\sigma_{i_{-1}}....\sigma_{i_{-D}}\right)$.

The finite state machine constructed above has D-Markov properties because the probability of occurrence of symbol σ_{i_ℓ} on a particular state depends only on the configuration of that state, i.e., the previous D symbols. Once the alphabet size $|\Sigma|$ and word length D are determined at the nominal condition (i.e., time epoch t_0), they are kept constant for all slow time epochs $\{t_1, t_2, ...t_k....\}$. That is, the partitioning and the state machine structure generated at the nominal condition serve as the reference frame for data analysis at subsequent slow time epochs.

The states of the machine are marked with the corresponding symbolic word permutation and the edges joining the states indicate the occurrence of a symbol σ_{i_ℓ}. The occurrence of a symbol at a state may keep the machine in the same state or move it to a new state. On a given symbol sequence $....\sigma_{i_1}\sigma_{i_2}...\sigma_{i_l}....$ generated from the time series data collected at a slow time epoch, a window of length D is moved by keeping a count of occurrences of word sequences $\sigma_{i_1}\cdots\sigma_{i_D}\sigma_{i_{D+1}}$ and $\sigma_{i_1}\cdots\sigma_{i_D}$ which are respectively denoted by $N(\sigma_{i_1}\cdots\sigma_{i_D}\sigma_{i_{D+1}})$ and $N(\sigma_{i_1}\cdots\sigma_{i_D})$. Note that if $N(\sigma_{i_1}\cdots\sigma_{i_D}) = 0$, then the state $q \equiv \sigma_{i_1}\cdots\sigma_{i_D} \in Q$ has zero probability of occurrence. For $N(\sigma_{i_1}\cdots\sigma_{i_D}) \neq 0$, the transitions probabilities are then obtained by these frequency counts as follows:

$$\pi_{jk} \equiv P(q_k|q_j) = \frac{P(q_k, q_j)}{P(q_j)} = \frac{P(\sigma_{i_1}\cdots\sigma_{i_D}\sigma)}{P(\sigma_{i_1}\cdots\sigma_{i_D})}$$
$$\Rightarrow \pi_{jk} \approx \frac{N(\sigma_{i_1}\cdots\sigma_{i_D}\sigma_)}{N(\sigma_{i_1}\cdots\sigma_{i_D})} \tag{2.3}$$

where the corresponding states are denoted by $q_j \equiv \sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_D}$ and $q_k \equiv \sigma_{i_2}\cdots\sigma_{i_D}\sigma$. The state transition matrix, $\mathbf{\Pi} = [\pi]_{jk}$ satisfies the Stochastic matrix properties, i.e. $\Sigma_k \pi_{jk} = 1 \forall j$

2.2.4 Stopping Rule for Determining Symbol Sequence Length

This subsection presents a stopping rule that is necessary to find a lower bound on the length of symbol sequence required for parameter identification of the stochastic matrix $\mathbf{\Pi}$. The stopping rule [59] is based on the properties of irreducible stochastic matrices [60]. The state transition matrix is constructed at the r^{th} iteration (i.e., from a symbol sequence of length r) as $\Pi(r)$ that is an $n \times n$ irreducible stochastic matrix under stationary conditions. Similarly, the state probability vector $\mathbf{p}(r) \equiv [p_1(r)\ p_2(r)\ \cdots\ p_n(r)]$ is obtained as

$$p_i(r) = \frac{r_i}{\sum_{j=1}^{n} r_i} \tag{2.4}$$

where r_i is the number of symbols in the i^{th} state such that $\sum_{i=1}^{n} r_i = r$ for a symbol sequence of length r. The stopping rule makes use of the Perron-Frobenius Theorem [60] to establish a relation between the vector $\mathbf{p}(r)$ and the matrix $\Pi(r)$. Since the matrix $\Pi(r)$ is stochastic and irreducible, there exists a unique eigenvalue $\lambda = 1$ and the corresponding left eigenvector $\mathbf{p}(r)$ (normalized to unity in the sense of absolute sum). The left eigenvector $\mathbf{p}(r)$ represents the state probability vector, provided that the matrix parameters have converged after a sufficiently large number of iterations. That is,

$$\mathbf{p}(r+1) = \mathbf{p}(r)\Pi(r) \quad \Rightarrow \quad \mathbf{p}(r) = \mathbf{p}(r)\Pi(r) \ \text{ as } r \to \infty \tag{2.5}$$

Following Eq. (2.4), the absolute error between successive iterations is obtained such that

$$\| \left(\mathbf{p}(r) - \mathbf{p}(r+1)\right) \|_\infty = \| \mathbf{p}(r) \left(\mathbf{I} - \Pi(r)\right) \|_\infty \leq \frac{1}{r} \tag{2.6}$$

where $\| \bullet \|_\infty$ is the max norm of the finite-dimensional vector $\bullet$.

To calculate the stopping point r_{stop}, a tolerance of η $(0 < \eta \ll 1)$ is specified for the relative error such that:

$$\frac{\| \left(\mathbf{p}(r) - \mathbf{p}(r+1)\right) \|_\infty}{\| \left(\mathbf{p}(r)\right) \|_\infty} \leq \eta \quad \forall\ r \geq r_{stop} \tag{2.7}$$

The objective is to obtain the least conservative estimate for r_{stop} such that the dominant elements of the probability vector have smaller relative errors than the

remaining elements. Since the minimum possible value of $\| \mathbf{p(r)} \|_\infty$ for all r is $\frac{1}{n}$, where n is the dimension of $\mathbf{p}(r)$, the least of most conservative values of the stopping point is obtained from Eqs. (2.6) and (2.7) as:

$$r_{stop} \equiv int\left(\frac{n}{\eta}\right) \tag{2.8}$$

where $int(\bullet)$ is the integer part of the real number $\bullet$. At the (slow time) epoch, t_k, the state probability vector is denoted as $\mathbf{p}^k$.

A recent publication [61] describes a stopping rule based on Markov chain Monte Carlo (MCMC) computations. The stopping criterion is obtained via a relation between the tolerance η and the absolute error bound ϵ, which is generated offline by the learning algorithm. Subsequently, the stopping rule is executed online to obtain an adaptive confidence interval of the state probability vector p of the PFSA for anomaly detection.

2.2.5 Anomaly Evolution and Pattern Identification

Behavioral pattern changes may take place in dynamical systems due to accumulation of faults and progression of anomalies. The pattern changes are quantified as deviations from the nominal pattern (i.e., the probability distribution at the nominal condition). The resulting anomalies (i.e., deviations of the evolving patterns from the nominal pattern) are characterized by a scalar-valued function, called *Anomaly Measure* μ. The anomaly measures at slow time epochs $\{t_1, t_2, \ldots\}$ are obtained as:

$$\mu_k \equiv d\left(\mathbf{p}^k, \mathbf{p}^0\right) \tag{2.9}$$

where the $d(\bullet, \bullet)$ is an appropriately defined distance function.

The major advantages of SDF for small anomaly detection are listed below:

- Robustness to measurement noise and spurious signals [3]
- Adaptability to low-resolution sensing due to the coarse graining in space partitions [2]
- Capability for early detection of anomalies because of sensitivity to signal distortion and real-time execution on commercially available inexpensive plat-

forms [42].

2.3 Construction of Anomaly Detection Algorithms

This section explains how anomaly detection algorithms are constructed for symbolic dynamic filtering (SDF) and several other pattern recognition tools that are briefly described in A.

2.3.1 Symbolic Dynamic Filtering for Anomaly Detection

The following steps, summarize the procedure of SDF for anomaly detection.

- ***Time series data acquisition*** on the fast scale from sensors and/or analytical measurements (i.e., outputs of a physics-based or an empirical model). Data sets are collected at different slow time epochs.

- ***Generation of wavelet transform coefficients*** [58], obtained with an appropriate choice of the wavelet basis and scales [3]. The wavelet transform largely alleviates the difficulties of phase-space partitioning and is particularly effective with noisy data from high-dimensional dynamical systems [56].

- ***Partitioning*** *[3] of the wavelet space* at the nominal condition at time epoch t_0. Each segment of the partitioning is assigned a particular symbol from the symbol alphabet set Γ. This step enables transformation of time series data from the continuous domain to the symbolic domain [15]. The partitioning is fixed for subsequent slow time epochs.

- ***Construction of a finite state automaton*** at time epoch t_0 (nominal condition) from alphabet size $|\Gamma|$ and window length D. The structure of the finite state machine is fixed for subsequent slow time epochs $\{t_1, t_2, ...t_k....\}$, i.e., the state machine structure generated at the nominal condition serve as the reference frame for data analysis at subsequent slow time epochs.

- ***Calculation of the state probability vector*** $\mathbf{p}^0$ at time epoch t_0 whose elements represent the state visiting probabilities of the finite state machine.

The probability distribution $\mathbf{p}^0$ of damage patterns is recursively computed as an approximation of the natural invariant density of the dynamical system at the slow time epoch t_0, which is a fixed point of the local Perron-Frobenius operator.

- ***Time series data acquisition on fast time scale at subsequent slow time epochs***, $t_1, t_2, ...t_k...$, and their conversion to the wavelet domain to generate respective symbolic sequences based on the partitioning at time epoch t_0.

- ***Generation of the state probability vectors*** $\mathbf{p}^1, \mathbf{p}^2, ...\mathbf{p}^k...$ at slow time epochs, $t_1, t_2, ...t_k...$ from the respective symbolic sequences using the finite state machine constructed at time epoch t_0.

- ***Computation of scalar anomaly measures*** $\mu_1, \mu_2, ..., \mu_k, ...$ at time epochs, $t_1, t_2, ..., t_k, ...$ based on evolution of these probability vectors and by defining an appropriate distance function $d(\mathbf{p}^k, \mathbf{p}^0)$ with respect to the nominal condition [2]. Therefore, the pattern changes in the state probability vector are quantified as deviations from the nominal behavior and are characterized by a scalar-valued function, called *Anomaly Measure* μ. The distance function is chosen as the standard Euclidean norm.

2.3.2 Bayesian Filtering for Anomaly Detection

Bayesian filtering tracks the states more effectively if the system is closer to the nominal condition. In other words, and the error would be greater when the system is in an anomalous condition. To this effect, the innovation sequences are computed, and their histograms are obtained, where the innovation ϵ is defined as the difference between the true output y and the predictor output $\hat{y}^-$ [44].

$$\text{Innovation} : \epsilon = y - \hat{y}^- = y - C\hat{x}^- \tag{2.10a}$$

$$\tag{2.10b}$$

In the nominal condition, the model is a very close approximation of the data

that is generated, and the system is able to estimate the states with the lowest error. The histogram of the innovation sequence thus resembles a Gaussian sequence with very small variance. As the anomaly is increased, the model becomes less accurate and the estimation errors become higher. Thus, the histogram of the innovation sequence shows an increase in the variance and the distribution diverges from Gaussian. Ultimately, the histograms are expected to converge to a uniform distribution if the filters no longer tracks the system. This increase is characterized as a measure of the anomaly. To this effect, the probability density of the innovation sequences $\mathbf{p}^k(\epsilon)$ are generated at slow time epochs t_k and the anomaly measure at any epoch k is given by an appropriate distance function $d(\mathbf{p}^k(\epsilon), \mathbf{p}^0(\epsilon))$ between the density functions at epoch t_k and at nominal condition at epoch t_0. The distance function is chosen as the standard Euclidean norm.

2.3.3 Neural Networks for Anomaly Detection

The training data set for both types of neural networks, namely, Radial Basis Function Neural Networks, ($RBFNN$) and Multi Layer Perceptron Neural Networks, ($MLPNN$) (see A), are prepared in the same manner. In both cases, the neural networks are trained based on the standard NARX model [62] from the input-output data sets at the nominal condition. Thus, the training input vector for the networks contain the current input $u(k+1)$, the current output $y(k+1)$ as well as two past outputs, $y(k)$ and $y(k-1)$. The target for the network is the current output $y(k+1)$. After an error goal is achieved, the neural network is allowed to track the output signal of the system under both nominal and anomalous conditions. Upon feeding the input of the (possibly) anomalous system, the neural network generates an output signal estimate $\hat{y}$. The innovation $\epsilon_k \triangleq (y_k - \hat{y}_k)$ serves as a measure for the tracking performance of the neural network filters. The probability density function (pdf) is created for the innovation sequence. If at nominal condition the pdf is $\mathbf{p}^0$ and the pdf at slow time epoch t_k is $\mathbf{p}^k$, then the anomaly measure is given by the distance $d(\mathbf{p}^k, \mathbf{p}^0)$. The distance function is chosen as the standard Euclidean norm.

2.3.4 Statistical methods for Anomaly Detection

Two statistical analysis methods, namely, Principal Component Analysis (PCA) and Kernel Regression Analysis (KRA) (see A) have also been investigated.

Principal Component Analysis serves as a feature selector in the pattern analysis via dimension reduction from n to m. The $n \times n$ covariance matrix, obtained from the time series data, generates the orthonormal eigenvectors v^k and the corresponding non-negative real eigenvalues λ_k. The eigenvalues are arranged in the increasing order of magnitude. The m largest eigenvalues and associated eigenvectors are selected such that $\sum_{i=1}^{m} \lambda_i > \eta \sum_{i=1}^{n} \lambda_i$, where η is a real positive number close to 1 (e.g., $\eta = 0.95$). The principal feature matrix F is defined as:

$$F = \left[\begin{array}{ccc} \sqrt{\frac{\lambda_1}{\sum_{i=1}^{m} \lambda_1}}\, v^1 & \cdots & \sqrt{\frac{\lambda_d}{\sum_{i=1}^{m} \lambda_d}}\, v^d \end{array} \right] \tag{2.11}$$

The feature matrix F^0 represents the status of the system derived from the time series data at the nominal condition t_0. Similarly, feature matrix F^k is obtained from time series data at slow time epoch t_k. Then, the anomaly measure at t_k is obtained as the distance $d(F^k, F^0)$. The distance function is chosen as the standard Euclidean norm.

In Kernel Regression Analysis At the nominal condition, the kernel estimator is $\hat{f}_0(x)$. For different anomalous conditions, the regression parameters,(μ, θ_α), are kept fixed; and the kernel estimator $\hat{f}_k(x)$ is evaluated from the data set under the (possibly anomalous) condition at the slow time epoch t_k . Then, the anomaly measure at the k^{th} epoch is obtained as the distance $d(\hat{f}_k, \hat{f}_0)$. The distance function is chosen as the standard Euclidean norm.

2.4 Operation of the Symbolic Dynamic Filter

This section outlines the selection of several operating parameters of the Symbolic Dynamic Filter. While a number of parameters such as wavelet basis (or the choice between wavelet and Hilbert transform) are discussed in other publications ([2, 3]), this dissertation in particular deals with the selection of Depth D and the alphabet size $|\Sigma|$. The method is an extension of the information theoretic procedure followed in [3]. In addition to the depth and alphabet size, two more

operating parameters are looked at, namely the data length to be considered, and the rate of sampling required. The data length deals directly with the stopping rule when the maximum amount of data required is considered. Also, some aspects of varying the sampling rate are studied. Much of the following discussion in this section is in the context of the Duffing equation which is outlined below for completeness.

$$\frac{d^2x(t)}{dt^2} + \beta\frac{dx}{dt} + x(t) + x^3(t) = A\cos(\omega t) \tag{2.12}$$

The Duffing system is explained in greater detail later in Section 5.1.1 It should be noted that all operating parameters are chosen for the *nominal conditions* of the system under consideration. Here, this represents $\beta = 0.10$, and the excitation input given by $A = 22$ and $\omega = 5$. The subsequent discussion is for un-processed Duffing data (i.e. no wavelet analysis). However, a very similar procedure can be followed if the user chooses to apply the wavelet or Hilbert transforms to the data.

2.4.1 Selection of Depth D and Alphabet size $|\Sigma|$ for SDF

The depth D and alphabet size $|\Sigma|$ are selected using a conflicting objective pareto optimization technique based on the principles of information theory. Choosing a larger value of depth would imply that the future value of the signal has a higher dependency on a larger number of past values of the signal. A lower value of depth implies that the signal has a lower order Markovian nature. Two other aspects that come into play while selecting both depth and alphabet size are

1. The amount of information lost by coarse graining the analog signal to discrete symbols.

2. The computational cost in terms of memory and space requirements that is associated with computing the state transition matrix Π.

These are two conflicting objectives, and hence a Pareto-optimization technique is utilized. A Neyman Pearson technique can then be used to determine the optimum operating conditions for a given user, and for a given system.

A primary task involves defining a metric to quantify the information lost in the symbolizing process. The metric used is the information content, or the entropy of

the state transition matrix $\tilde{\Pi}$. It should be noted that this entropy is greater than that of the entropy of simply the probability vector, since the latter only stores the state (or symbol) visit probabilities, while the former contains information about not just the frequency of visits, but also the transitions between different states (or symbols). The states and symbols are as always, analogous in the case of depth $D = 1$.

In order to define the information loss, consider time series data from the duffing equation given in Eq.2.12. As outlined in Section 2.3, the state transition matrix $\tilde{\Pi}$ and the state probability vector $\mathbf{p}^0$ are calculated. For ease of notation, this is referred to simply as $\mathbf{p}$ in this section, since all SDF parameters are selected for the nominal condition of the data (i.e. $k = 0$). In this case, for a given depth D and alphabet size $|\Sigma|$, the structure of this state probability vector is given as:

$$\mathbf{p} = \begin{bmatrix} p_1 & p_2 & \cdots & p_m \end{bmatrix} \tag{2.13}$$

where $|\Sigma| \leq m \leq |\Sigma|^D$

That is p_i are the constituents of the state probability vector, where i ranges from 1 to $|\Sigma|^D$. Note that the state probability vector is fundamentally different from the symbol probability vector which would have only $|\Sigma|$ elements for all values of depth D.

Similarly, let each element of the $\tilde{\Pi}$ matrix be given by π_{jk} where j, the number of rows ranges from 1 to $|\Sigma|^D$ and k, the number of columns ranges from 1 to D. Note that π_{jk} represents the probability of transition from state j to state k.

Now, the information contained in the $\tilde{\Pi}$ matrix, denoted by $I_{\tilde{\Pi}}$ is defined as:

$$I_{\tilde{\Pi}} \triangleq E\left[\sum_k \pi_{jk} \ln(\pi_{jk})\right] = -\sum_j \sum_k p_j \times \pi_{jk} \ln(\pi_{jk}) \tag{2.14}$$

This represents the standard entropy equation, where the entropy is calculated over each row of the $\tilde{\Pi}$ matrix and normalized by the probability of occurence of that particular state.

Figure 2.5 shows a graph of $I_{\tilde{\Pi}}$ for the Duffing data, for different values of alphabet size and depth D. The blue line shows different values of information for $D = 1$ while the green line shows the same quantity for $D = 2$. A similar trend is

expected for larger depths.

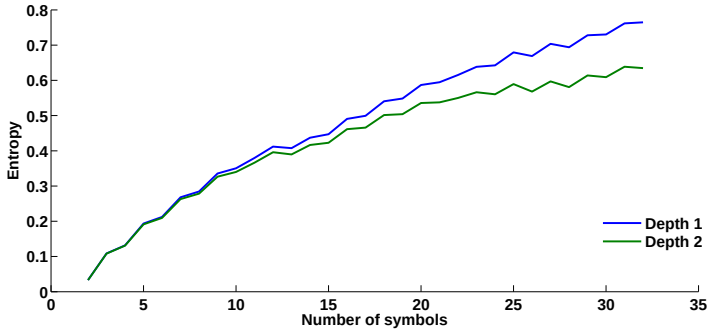


Figure 2.5. Entropy for different depths and alphabet sizes

While this quantity $I_{\tilde{\Pi}}$ from equation 2.14 should be maximized, its important to note that there is generally a bound on the memory and computation power available. Also, it can be noted that even with a large alphabet size, there is always a loss in information as compared to the original analog signal.

The next aspect involves a brief study of the time and memory requirements for SDF. Using the `tic` and `toc` functions of MATLAB, the actual amount of time required to compute the state transition matrix for a fixed length of data was computed for different depths and alphabet sizes. Also, using a profiler, the memory required to store the different variables was calculated. The following figures show the results obtained.

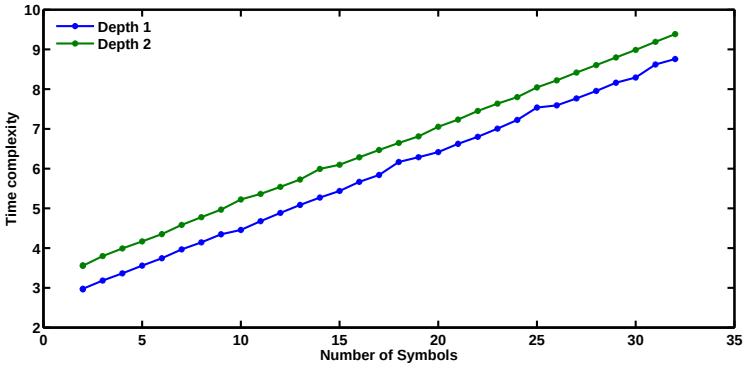


Figure 2.6. Time Complexity for different depths and alphabet sizes

One interesting observation from Figure 2.6 is that SDF shows linearity in time (and hence computational) complexity with respect to number of symbols,

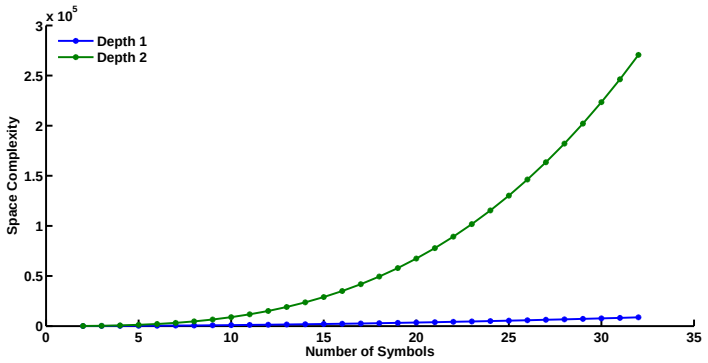


Figure 2.7. Space Complexity for different depths and alphabet sizes

for both depth 1 and depth 2. Similar results are expected for higher values of depth. However, space complexity increases exponentially with an increase in the number of symbols. It is expected that the order of the exponent is equal to the numerical value of the selected depth. These parameters are extremely important when SDF is deployed on a mobile platform with limited computation capacity and memory.

To select an optimum value of depth and alphabet size, a Neyman Pearson technique is employed. Basically, the information content needs to be maximized within a specific computation threshold. The following figure shows a plot of $I_{\tilde{\Pi}}$ vs. the time complexity computed above.

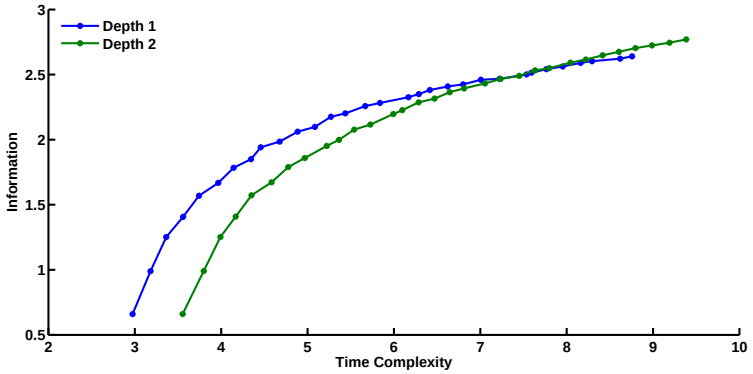


Figure 2.8. Selection of Optimum Depth and Alphabet Size

Figure 2.8 shows the tradeoff between entropy and time complexity. The values of these two quantities are computed for different combinations of depth and alphabet size. The blue line connects points for different number of symbols for

$D = 1$, while the green line represents $D = 2$. The fact that the blue line is entirely over the green line for lower values of time complexity implies that for the same time complexity (or in other words, available computation power), a selection for $D = 1$ would have a higher entropy than the corresponding selection for $D = 2$. If a higher amounts of computation power is available, then a higher value of depth can be selected. Another observation is that for the same alphabet size, increasing the Depth does not really increase the value of $I_{\tilde{\Pi}}$, although there is an increase in the time complexity involved. This is a very important aspect of SDF that validates the choice of $D = 1$ for several previous publications. It can be seen that a knee point of the $D = 1$ curve occurs for the number of symbols to be between 8 and 12, after which there is not a significant increase in the amount of information obtained. For most of this dissertation, the alphabet size selected for the Duffing data is $|\Sigma| = 8$.

2.4.2 Aspects of sampling rate for SDF

One important difference between SDF and methods like the Kalman filter etc. is that the choice of operating parameters depends on the sampling rate used. If the data is very coarsely sampled (i.e. very close to Nyquist rate), then it would become more difficult to make predictions for lower values of depth D. However, for very finely sampled data, the value of $I_{\tilde{\Pi}}$ might appear to be extremely high since making predictions on the partitioned data would be easier. In such a case, a higher value of D might need to be selected, or in other words, a larger amount of history might need to be considered. Thus, it is important to have a rich $\tilde{\Pi}$ matrix for efficient implementation of the Symbolic Dynamic Filter. Figure 2.9 shows the information contained in the $\tilde{\Pi}$ matrix for Duffing data that has been down-sampled at different rates.

It can be seen in Figure 2.9 that the same signal contains different amounts of information for 2 different sampling rates. This happens since signal dynamics change at the state machine level for different levels of sampling. It is expected that for very finely sampled data, the signal would stay in the same state very often. Hence, the diagonal elements would dominate the $\tilde{\Pi}$ matrix, while only a few off diagonal elements are expected to be populated. However, as the signal is

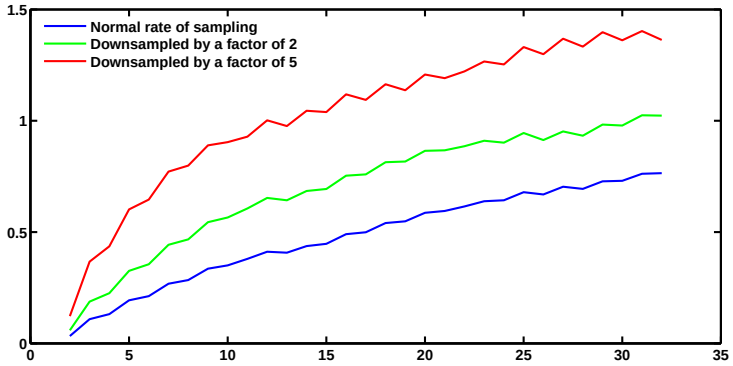


Figure 2.9. Effects of Sampling Rate

downsampled (or more coarsely sampled to begin with), it is expected that the $\tilde{\Pi}$ matrix has more non-zero elements, and the diagonal terms are non-dominated. This in turn leads to a higher value of information entropy as defined in Equation 2.14. It should be noted here that while the sampling rate is not a criteria for future calculations using the $\tilde{\Pi}$ matrix or the $\mathbf{p}$ vector, the selection of Depth and alphabet size need to be made for data that has the same sampling rate as the test case data. Also, it should be expected that data sampled at a different rate would show different stationarity properties at the $\tilde{\Pi}$ matrix level, even if it is obtained from the same source. However, the $\mathbf{p}$ vector would be similar in the special case of $D = 1$ since it records only state visit probabilities.

2.4.3 Aspects of data length - Minimum amount of data required

Some aspects of the maximum data length required for SDF were discussed in Section 2.2.4. However, it is possible to observe spurious information in the state transition matrix (and hence in the state probability vector) if adequate data is not considered. Here, a necessary condition is defined to ensure that this minimum data length is met. This condition is that for a similar length of data, the information content of a purely noisy signal should be very close to zero. It is expected that in a pure noise case (additive white Gaussian noise), each element is independent of the previous element. That is, pure white noise is a Markov process of order zero. Hence, from a given state, transitions to all other states are equally likely. Also,

the occurrence of each state should be equally probable. Hence, the information entropy as defined above should be very close to zero if sufficient data is considered. However, if less data is used for the computation, it is possible that the state transition matrix $\tilde{\Pi}$ does not get completely populated, and has some entries that have not converged. In this case, such a matrix would indeed result in a non-zero value of $I_{\tilde{\Pi}}$. This aspect must always be considered while implementing SDF on any system.

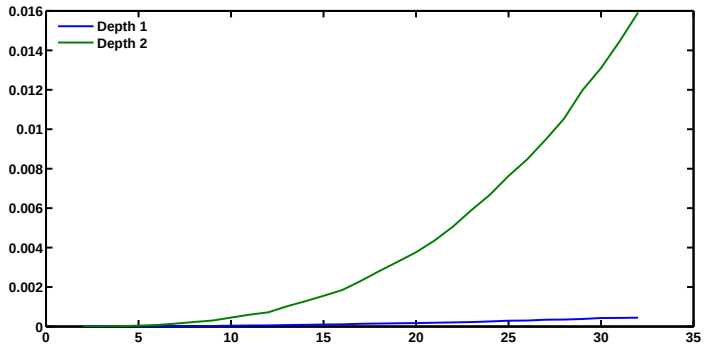


Figure 2.10. Effects of Sampling Rate

Figure 2.10 shows this effect. It can be seen that for white noise, for a sufficient data length (50000 points), for $D = 1$, and for low values of alphabet size, the information content is very close to 0. However, as the alphabet size is increased, the number of states is increased, and the number of elements in the Π matrix increases. Hence, a larger amount of data is required to populate the $\tilde{\Pi}$ matrix. For $D = 2$, the number of elements of the $\tilde{\Pi}$ matrix increases with order $|\Sigma|^2$. Hence, for the same alphabet size, much more data is required as compared to $D = 1$. This can be seen by the fact that the curve has its knee point for far fewer symbols, and noise appears to contain information. This is also an important aspect, since if lesser data is available, this puts an upper bound on the depth and alphabet size that can be selected for SDF.

Chapter 3

Framework of Statistical Estimation of Multiple Parameters

3.1 Introduction

The framework of *SDF* includes preprocessing of time series data by time-frequency analysis (e.g., wavelet transform [58] and Hilbert transform [63, 64]). The transformed data set is partitioned using the maximum entropy principle [3] to generate the symbol sequences from the transformed data set without any significant loss of information. Subsequently, statistical patterns of the evolving system dynamics are identified from these symbol sequences through construction of probabilistic finite-state automata (*PFSA*). An additional advantage of transform space-based partitioning is reduction of spurious noise in the data set from which the *PFSA* is constructed; this feature provides additional robustness to *SDF* as discussed in [3]. The state probability vectors that are derived from the respective state transition probability matrices of *PFSA* serve as behavioral patterns of the evolving dynamical system under nominal and off-nominal conditions.

In a system with multiple fault conditions present, it is important to distinguish between the fault conditions. Multiple fault conditions often influence the conventional criteria by which the degradation of faults are trended. For example, traditional frequency based methods cannot be used to estimate the presence of multiple faults [38,65]. A situation may arise where certain fault conditions are sta-

ble while the conventional analytical procedures indicate degradation of the fault. This phenomenon occurs due to the presence of other fault conditions that are deteriorating. Also, repeated fault diagnosis on complex dynamical systems may often be computationally prohibitive due to expensive simulation requirements. Fault dictionaries are sometimes used to alleviate this problem, but they may be infeasible to store because of their large sizes [32]. The problem occurs because dictionaries usually only store primary output information.

Parameter estimation algorithms, based on symbolic dynamic filtering (*SDF*), have been experimentally validated for real-time execution in different applications, such as degradation monitoring in electronic circuits [66] and fatigue damage monitoring in polycrystalline alloys [42]. While these applications of *SDF* have focused on estimation of only a single parameter, the work reported here addresses statistical estimation of multiple parameters. Specifically, this work is an extension of the earlier work [67] on single parameter estimation to estimation of multiple parameters that may vary simultaneously. The resulting algorithms are validated on the various testbeds described in Chapter 5.

3.2 Symbolic Dynamic Filtering and Single parameter Estimation

This section succinctly reviews the underlying concept of single-parameter estimation [67] in the SDF framework.

Extraction of statistical behavior patterns from time series data is posed as a two-scale problem. The *fast scale* is related to response time of the process dynamics. Over the span of data acquisition, dynamic behavior of the system is assumed to remain invariant, i.e., the process is quasi-stationary at the fast scale. In other words, variations in the statistical behavior of the dynamical system are assumed to be negligible on the fast scale. The *slow scale* is related to the time span over which deviations (e.g., parametric changes) may occur and exhibit non-stationary dynamics. The parameters are estimated based on the information generated by *SDF* of the data collected over the fast scale at a slow scale epoch. This method is also applicable to estimation of slowly varying parameters. The rationale is

that, since the parameters vary slowly, they are treated as invariants at a given slow scale epoch; accordingly, the fast-scale statistical behavior of the dynamical system may change at different slow scale epochs (that are simply referred to as epochs in the sequel).

3.2.1 Forward Problem in the Symbolic Dynamic Setting

This subsection summarizes the *Forward Problem* for detection of deviation patterns in the SDF setting.

1. *Time series data acquisition on the fast scale from sensors and/or analytical measurements (i.e., outputs of a physics-based or an empirical model).* Data sets are collected at the parameter values as a set $\{s^0, s^1, \cdots, s^k, \cdots\}$, where s^k denotes the value of the parameter at the epoch k.

2. *Generation of wavelet transform coefficients with an appropriate choice of the wavelet basis and scales.* The wavelet transform largely alleviates the difficulties of phase-space partitioning and is particularly effective with noisy data from high-dimensional dynamical systems.

3. *Maximum Entropy Partitioning of the wavelet space at a reference condition.* Each segment of the partitioning is assigned a particular symbol from the symbol alphabet Σ. This step enables transformation of time series data from the continuous domain to the symbolic domain [15].

4. *Construction of a probabilistic finite state automaton (PFSA) at the reference condition.* The structure of the finite state machine is fixed for subsequent parameter values until a new reference is selected.

5. *Computation of the reference pattern vector $\boldsymbol{p}(s^0)$ whose elements represent state occupation probabilities of the PFSA at the reference condition.* Such a pattern vector is recursively computed as an approximation of the natural invariant density of the dynamical system, which is a fixed point of the local Perron-Frobenius operator [60]. Thus, $\mathbf{p}(s^0) \equiv [p_1(s^0)\ p_2(s^0)\ \cdots\ p_{|\Sigma|}(s^0)]$, where $|\Sigma|$ is the number of states in the *PFSA*.

6. *Time series data acquisition on the fast scale at subsequent parameter values, and their conversion to respective symbolic sequences based on the reference partitioning at the reference value.*

7. *Generation of the pattern vectors,* $\mathbf{p}(s^1), \mathbf{p}(s^2), \cdots, \mathbf{p}(s^k) \cdots$ *at parameter values,* $s^1, s^2, ...s^k...$ *from the respective symbolic sequences using the state machine constructed at nominal parameter value* s^0. Thus, $\mathbf{p}(s^k) \equiv \left[p_1(s^k)\ p_2(s^k) \cdots p_{|\Sigma|}(s^k)\right]$, where $|\Sigma|$ is the number of states in the *PFSA*. (Note that only $(|\Sigma| - 1)$ out of the $|\Sigma|$ elements of $\mathbf{p}(s^k)$ are linearly independent because $\mathbf{p}(s^k)$ is sum-normalized to unity.) The structure of the *PFSA* at all epochs is identical in the *SDF* framework, while the pattern vectors $\mathbf{p}(s^k)$ are possibly different at different parameter values s^k.

8. *Computation of deviation measures*: Evolving deviation measures $\mathcal{M}(s^1), \mathcal{M}(s^2), \cdots, \mathcal{M}(s^k), \cdots$ at parameter values, $s^1, s^2, \cdots, s^k, \cdots$, are computed with respect to the nominal condition at s^0, by selecting an appropriate distance function $d(\bullet, \bullet)$ (e.g., the standard Euclidean norm) such that

$$\mathcal{M}(s^k) \triangleq d(\mathbf{p}(s^k), \mathbf{p}(s^0)) \tag{3.1}$$

3.2.2 Inverse Problem of Single-parameter Estimation

This subsection focuses on the inverse problem of single-parameter estimation based on computed values of the deviation measure in the forward problem. The parameter to be estimated is treated as a random variable at each epoch, for which the deviation measure is an observable. To account for the inherent uncertainties in the system components and to ensure robust estimation, a large number of experiments are performed and the deviation measures are calculated from observed sets of time series data during each experiment, with the objective of estimating the unknown parameter. The steps for the statistical identification of the system parameter from the measured value of deviation measure are delineated below.

1. *Upon generation of deviation measure profiles in the forward problem, a statistical relationship is identified between deviation measure and the parameter*

associated with the deviation. In particular, probability distributions of the parameter are obtained for various values of the deviation measure. Then, statistical tests are performed to determine goodness-of-fit of the distributions. For example, mean and variance associated with a two-parameter distribution provide adequate statistical information on the bounds and confidence levels of the estimated parameter.

2. *Data acquisition on the fast scale at an unknown parameter value*. Time series data are collected (in the fast scale) under operating conditions similar to those in Step 1 of the forward problem. Data are analyzed to generate pattern vectors as described in the forward problem. The deviation measure $\mathcal{M}^{\text{test}}$ at parameter value s^{test} is then calculated by quantifying the deviation of the current pattern vector $\mathbf{p}^{\text{test}}$ from the nominal pattern vector $\mathbf{p}(s^1)$.

3. *Parameter estimation from generated statistics of deviation profile*. The estimated value of the parameter and its confidence interval are obtained based on the computed deviation measure and the probability distribution derived in Step 1 of the inverse problem.

In the above procedure, the range of the computed deviation measure profile is discretized into finitely many levels. A statistical distribution is hypothesized for determining spread of the parameter and goodness-of-fit of the hypothesized distribution that is assessed with χ^2 and Kolmogorov-Smirnov tests [68].

3.3 Overview of Single Parameter Solution of the Inverse Problem

This section presents the parameter estimation scheme used to estimate the value of a single parameter, specifically, the parameter β in the Duffing System. At this stage it is assumed that the steps required for calculating the anomaly measure from the time series data can be performed, i.e. for a certain input condition and a certain value of the parameter β, it is possible to come up with an anomaly measure, which characterizes the health of the system.

This section focuses on the inverse problem of parameter estimation based on computed values of the deviation measure. The parameter is a slowly varying random process and is therefore assumed to be a random variable at each slow time epoch, for which the deviation measures are the only observables. To account for the inherent uncertainties in the system components and to ensure robust estimation, a large number of experiments are performed and the deviation measures are calculated from observed time series data during every experiment, with the objective of estimating the unknown parameter.

The range of the computed deviation measure is discretized into finitely many levels. A pattern matrix is created where each column represents the spread of the parameter for a particular value of the deviation measure. A statistical distribution is hypothesized for the spread of the parameter and the goodness-of-fit of the hypothesized distribution is assessed with χ^2 and Kolmogorov-Smirnov tests. Confidence Intervals are then assessed for confidence levels at 99% and 95%.

3.3.1 Example for single parameter estimation

The Duffing experiement is considered for this example. Time series data is generated for different values of β, and the experiment is repeated about 40 number of times to acquire stochastic information on the system dynamics. For each run of the experiment, deviation measures M are calculated for different values of the parameter β.

The range of deviation measure is discretized into $n = 20$ levels. A pattern matrix is constructed where each column represents the spread of the parameter β for a particular segment of the deviation measure. For the elements of each column, a two parameter $log - normal$ distribution is hypothesized, and its goodness-of-fit is examined by both χ^2 and Kolmogorov-Smirnov tests [69]. For each of the 20 data sets, the hypothesis of two-parameter log-normal distribution passed the χ^2-test at 10% significance level [69]. This satisfies the conventional standard of 5% significance level. Also, for each of the 20 measure levels, the hypothesis passed the Kolmogorov-Smirnov test at 30% significance level,which again exceeds the conventional standard of 5% significance level.

3.4 Framework of Multi-parameter Estimation

In general, extension of single-parameter estimation [67] to multiple-parameter extension is not a straight-forward task as explained below.

Let us consider the Duffing system in Eq. (5.1), where the parameters to be estimated are chosen as α_1 and β; and the deviation measure $\mathcal{M}$ (see Eq. (3.1) is obtained for the parameter pair $\underline{s} \triangleq (s_1, s_2) = (\alpha_1,\ \beta)$. Fig. 3.1 shows a plot with α_1 on the x-axis, β on the y-axis and the contours of the deviation measure $\mathcal{M}$. Each contour is constructed by joining points with the same value of deviation measure $\mathcal{M}$; this is indicated by the gray scale (color) corresponding to the vertical bar on the right hand side of the plot. Values of deviation measure $\mathcal{M}$ are chosen in steps of 0.1 and a plane parallel to the x-y axis is constructed at these values of $\mathcal{M}$ to join points of equal values of deviation measure. As the system deviates in either direction from the nominal condition of $\alpha_1 = 1.0$ and $\beta = 0.1$, the deviation measure $\mathcal{M}$ increases until bifurcation occurs (e.g., $\alpha_1 = 1.0$ and $\beta \approx 0.32$). It is obvious that the inverse image of a singleton set of $\mathcal{M}$ may contain infinitely many combinations of α_1 and β. Hence, the information on $\mathcal{M}$ alone is insufficient for uniquely identifying the parameters α_1 and β that characterize the system. It demonstrates that non-uniqueness of estimation could occur if a scalar-valued function is chosen as the cost functional for optimized estimation of multiple parameters. This problem is resolved by considering the individual elements of the frequency probability vector for statistical estimation of the parameters as explained below.

It is shown that the estimation problem is non-convex in the sense that the the scalar distance between two frequency probability vectors could be zero for (possibly infinitely) many different combinations of a pair of system parameters. The problem of non-convexity is resolved by considering the individual components of the frequency probability vectors.

If the estimation of multiple parameters is set as an optimization problem with deviation measure $\mathcal{M}$ being the cost functional, then non-convexity may arise due to existence of contours; this situation could occur even if the range of optimization is narrow. Therefore, instead of relying on the deviation measure for parameter estimation $\mathcal{M}(s^k)$, as it was done in [30,31,67], variations in the individual elements

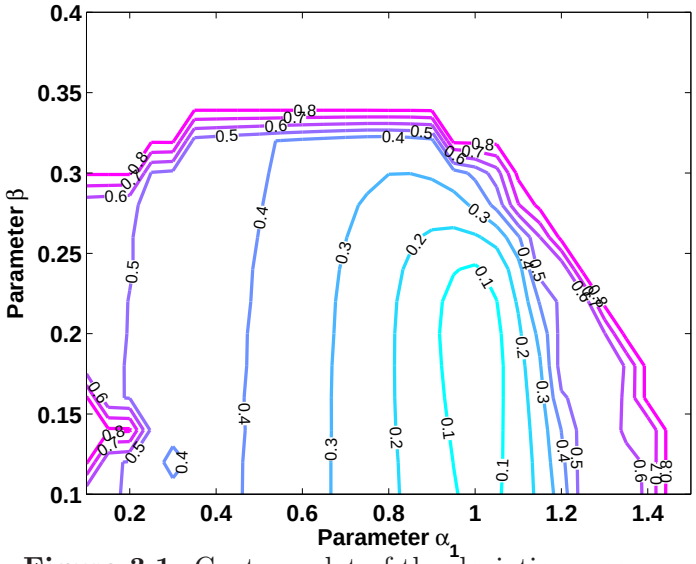


Figure 3.1. Contour plot of the deviation measure $\mathcal{M}$

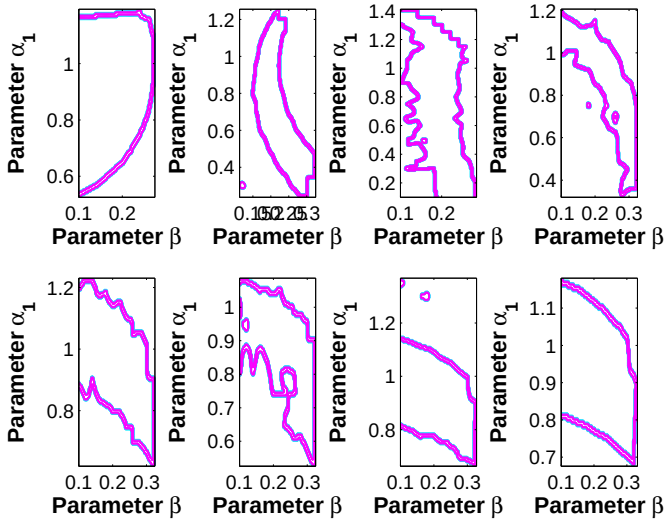


Figure 3.2. Contour plots of each element of the frequency probability vector $\mathbf{p}^k$ for a typical test case in the Duffing system

of $\mathbf{p}(s^k) \triangleq [p_1(s^k), p_2(s^k), \cdots, p_{|\Sigma|}(s^k)]$ are used in this dissertation. That is, the parameter estimation problem is reduced to identification of contours for $(|\Sigma| - 1)$ independent elements of the state probability vector $\mathbf{p}(s^k)$. The information derived from these $(|\Sigma|-1)$ independent contours would yield a statistical estimate of the parameter vector $\underline{s}^k$. This approach circumvents the aforementioned non-convexity problem.

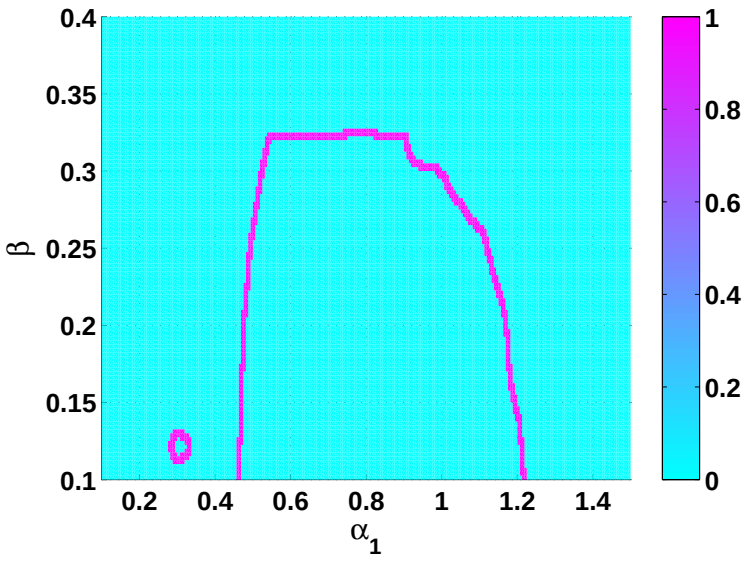


Figure 3.3. Contour showing all points where $\mu \sim 0.40$

For a given parameter pair of the Duffing system having values $s^k = (\alpha_1^k,\ \beta^k)$, the pattern vectors are generated as $\mathbf{p}(s^k) \equiv [p_1(s^k)\ p_2(s^k)\ \cdots\ p_{|\Sigma|}(s^k)]$. Having $|\Sigma| = 8$, eight plates in Fig. 3.2 shows contours for each of the 8 elements of $\mathbf{p}(s^k)$ for a given value of $(\alpha_1^k = 0.75,\ \beta^k = 0.23)$.

It is clear from Figure 3.1 that a single value of μ would contribute to infinitely many combinations of (β, α_1). For example, taking a section along $\mu = 0.4$ gives a contour shown in Figure 3.3. Now, every point (β, α_1) along this contour yields the same value of μ and hence the current information does not help in isolating the exact values of (β, α_1) that describe the system at the present case.

The following two sections describe a method that makes use of the ensemble of information in different contours to arrive at a more precise estimation of the parameters.

3.5 Construction of a Statistical Framework for Estimation of Multiple Parameters

Let $\mathcal{S}$ denote the collection of (finitely many) points in the n-dimensional parameter space, where the positive integer n is the number of parameters that are to

be estimated. That is, $\mathcal{S} = \left\{\underline{s}^1, \underline{s}^2, \cdots, \underline{s}^{|\mathcal{S}|}\right\}$, on which the training process is executed. Let Ω be the convex hull of $\mathcal{S}$, which represents the range over which the parameters take values. It is noted that Ω is a convex and compact subset of the separable space $\mathbb{R}^n$.

Let each element $\underline{s}^k \triangleq \left(s_1^k, s_2^k, \cdots, s_n^k\right)$ represent a particular set of parameters. For the Duffing system in Eq. (5.1), the set $\mathcal{S}$ consists of different values of parameters α_1 and β in the range where the experiments have been conducted; for example, a permissible value of $\underline{s}$ is $(\alpha_1,\ \beta) = (0.3,\ 0.4)$. Given an experimental time series data set Υ, the problem at hand is to identify the conditional probability density $f(\underline{s}|\Upsilon)$, where $\underline{s} \in \Omega$. The procedure of multi-parameter estimation consists of the forward problem and the inverse problem that are analogous to, but much more involved than, the single-parameter estimation procedure described in Section A.

3.5.1 Forward Problem/Training in a multiple parameter setting

For the forward problem, sets of time series data are generated by experimental runs at parameter values $\underline{s}^k, \forall k = 1, 2, \cdots, |\mathcal{S}|$. A symbolic dynamic filter is constructed to analyze each data sequence as outlined in Section A. For $|\Sigma|$ being the number of automaton states, the n-dimensional pattern vector $\mathbf{p}\left(\underline{s}^k\right)$ is generated for every $\underline{s}^k \in \mathcal{S}$. The procedure for data acquisition and storage for statistical analysis is described below.

For each $\underline{s}^k \in \mathcal{S}$, L different samples of the random parameter vector were collected, which were realized as identically manufactured but different electronic cards in experimental apparatus [67]. This implies that $L \times |\mathcal{S}|$ experiments need to be conducted on the apparatus. In this dissertation, there were $L = 40$ different realizations of the experimental apparatus.

In the specific case of the Duffing and van der Pol experiments, $|\mathcal{S}| = 50$ and hence the total number of experiments was $40 \times 50 = 2000$. Each experiment had an average duration of 3 minutes, implying that each realization with 50 different parameter pairs took about 2.5 hours. Experiments were conducted in parallel with different electronic cards on identical apparatuses. The entire task of data

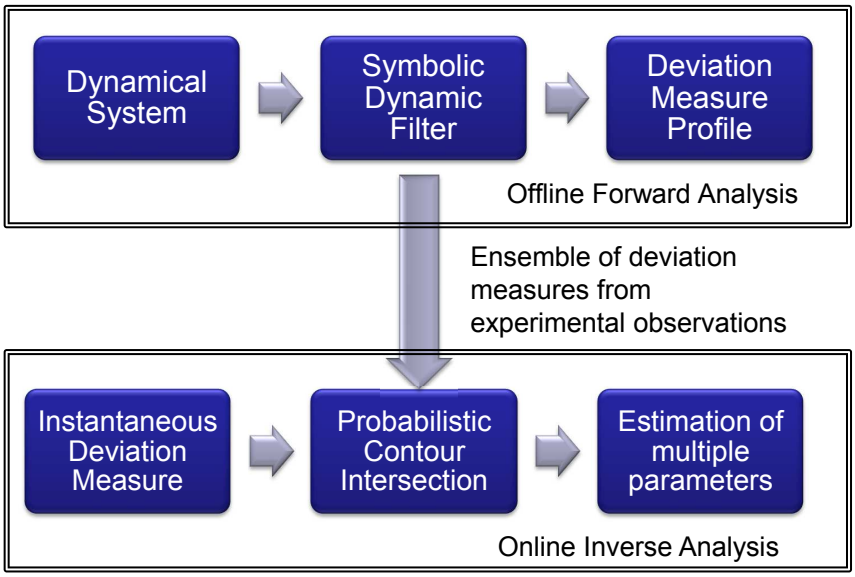


Figure 3.4. Flowchart for statistical estimation of multiple parameters in the SDF Framework

acquisition and storage was completed within 10 days by a team of two researchers.

The entire process of the forward and inverse problem is outlined in the flowchart in Figure 3.5.1

Let the elements $p_j(s^k), j = 1, 2, \cdots, |\Sigma|$ of the state probability vector $\mathbf{p}(s^k), k = 1, 2, \cdots, |\mathcal{S}|$ be modeled as a random variable $q_j(s^k)$ that is constructed from the ensemble of data points. The resulting random vector is obtained as

$$\mathbf{q}(s^k) \equiv \left[\begin{array}{cccc} q_1(s^k) & q_2(s^k) & \cdots & q_{|\Sigma|}(s^k) \end{array} \right] \tag{3.2}$$

where $q_j(s^k) \sim \mathcal{N}\left[m_j(s^k), \sigma_j^2(s^k)\right]$, i.e., $q_j(s^k)$ is modeled to be Gaussian with mean $m_j(s^k)$ and variance $\sigma_j^2(s^k)$, as explained below from the perspectives of state machine construction in the SDF setting. The equation for the modeled distribution is given as

$$f_{q_j|\mathcal{S}}\left(p_j|s^k\right) = \frac{1}{\sqrt{2\pi\sigma_j^2(s^k)}} \exp\left(-\frac{\left(p_j - m_j(s^k)\right)^2}{2\sigma_j^2(s^k)}\right) \tag{3.3}$$

The underlying dynamical system is modeled as an irreducible Markov process

via *SDF*, where the state probability vector is the sum-normalized eigenvector of the state transition matrix corresponding to the unique unity eigenvalue. Hence, no element in the state probability vector is either 0 or equal to 1. However, due to process noise and sensor noise, the random variable $q_j(s^k)$ fluctuates around its mean $m_j(s^k)$. While analyzing the experimental data, the standard deviation $\sigma_j(s^k)$ of the random variables $q_j(s^k)$ was found to be very small compared to its expected value $m_j(s^k)$, i.e., the ratio $\frac{\sigma_j(s^k)}{m_j(s^k)} \ll 1 \; \forall k = 1, 2, \cdots, |\mathcal{S}| \;\; \forall j = 1, 2, \cdots, |\Sigma|$. Therefore, a parametric or non-parametric two-sided unimodal distribution should be adequate to model the random variable $q_j(s^k)$. The choice of Gaussian distribution for q_j would facilitate estimation of the statistical parameters and involve only second order statistics. This assumption has been validated by using the χ^2 and Kolmogorov-Smirnov tests for goodness of fit [68] of each q_j for Gaussian distribution.

Remark 3.5.1. *The random variables $q_j(s^k)$ must satisfy the following two conditions:*

- *Positivity, i.e., $q_j(s^k) > 0 \;\; \forall \underline{s} \in \mathcal{S} \;\; \forall j = 1, 2, \cdots, |\Sigma|$,. This is made possible by truncating the far end of the Gaussian distribution tail on the left side. The goodness of fit of the distribution as Gaussian still remains valid at a very high significance level.*
- *Unity sum of the state probabilities, i.e., $\sum_{j=1}^{|\Sigma|} q_j(s^k) = 1 \;\; \forall \underline{s} \in \mathcal{S}$. This is achieved by sum-normalization.*

Remark 3.5.2. *The automaton states are analogous to energy states in statistical mechanics of ideal gases [70]. This fact is used formulating the inverse problem as explained below.*

3.5.2 Inverse Problem/Testing in a multiple parameter setting

Let time series data be generated from a new test on the experimental apparatus. The task at hand is to identify, from this data set, the unknown parameter vector

$\underline{s} \in \Omega$; however, it is possible that $\underline{s} \notin \mathcal{S}$. The data are analyzed using the same symbolic dynamic filter constructed in the forward/training problem (see Section 3.5.1), and the resulting probability vector $\mathbf{p} \equiv [p_1 \cdots p_{|\Sigma|}]$ is a realization of a random vector $\mathbf{q} \equiv [q_1 \cdots q_{|\Sigma|}]$. The density function $f_{\Omega|q}(\underline{s}|\mathbf{p})$ is obtained as

$$\begin{aligned} f_{\Omega|\mathbf{q}}(\underline{s}|\mathbf{p}) &= \frac{f_{\mathbf{q}|\Omega}(\mathbf{p}|\underline{s})\, f_{\Omega}(\underline{s})}{f_{\mathbf{q}}(\mathbf{p})} \\ &= \frac{f_{\mathbf{q}|\Omega}(\mathbf{p}|\underline{s})\, f_{\Omega}(\underline{s})}{\int_{\Omega} f_{\mathbf{q}|\Omega}(\mathbf{p}|\tilde{\underline{s}})\, f_{\Omega}(\tilde{\underline{s}}) d\tilde{\underline{s}}} \end{aligned} \tag{3.4}$$

In the absence of *a priori* information, an assumption is made that all operating conditions are equally likely, i.e., $f_{\Omega}(\underline{s}) = f_{\Omega}(\tilde{\underline{s}})\ \forall \underline{s},\ \tilde{\underline{s}} \in \Omega$. With this assumption of uniform probability, Eq. (3.4) reduces to

$$f_{\Omega|\mathbf{q}}(\underline{s}|\mathbf{p}) = \frac{f_{\mathbf{q}|\Omega}(\mathbf{p}|\underline{s})}{\int_{\Omega} f_{\mathbf{q}|\Omega}(\mathbf{p}|\tilde{\underline{s}}) d\tilde{\underline{s}}} \tag{3.5}$$

It is noted that accuracy of the above distribution would be improved if the actual prior mapping, i.e., $f_{\Omega}(\tilde{\underline{s}})$ is known.

The integral in the denominator of Eq. (3.5) is approximated by a Reimann sum as

$$f_{\Omega|\mathbf{q}}(\underline{s}|\mathbf{p}) \approx \kappa \frac{f_{\mathbf{q}|\Omega}(\mathbf{p}|\underline{s})}{\sum_{\mathcal{S}} f_{\mathbf{q}|\Omega}(\mathbf{p}|\tilde{\underline{s}})} \tag{3.6}$$

where κ is a constant. This approximation converges to the exact solution as the training set $\mathcal{S}$ approaches a (countable) dense subset of $\Omega \subset \mathbb{R}^n$.

The density function in Eq. (3.6) is now sampled at the points s^k in the training set $\mathcal{S}$ to construct the following sampled density to yield

$$f_{\Omega|\mathbf{q}}(\underline{s}|\mathbf{p})\big|_{\underline{s}=s^k} \approx \kappa \frac{f_{\mathbf{q}|\Omega}\left(\mathbf{p}|s^k\right)}{\sum_{\tilde{\underline{s}} \in \mathcal{S}} f_{\mathbf{q}|\Omega}(\mathbf{p}|\tilde{\underline{s}})}\ \forall s^k \in \mathcal{S} \tag{3.7}$$

The derivation till this stage (i.e. Eq. 3.7) does not use any information that was gathered during the forward problem. Depending upon the underlying distribution of $f_{\Omega|\mathbf{q}}(\underline{s}|\mathbf{p})$, this equation can be further simplified. A natural choice for the distribution based on Remark 3.5.1 would be to use the Dirichlet distribu-

tion [71]. The support of the Dirichlet distribution is a vector of real numbers in the range $(0, 1)$, all of which sum to 1, a property that is satisfied by the feature vectors $\mathbf{q}$. An advantage of this distribution is that it would be parametrized by only $|Q|$ values. For $D = 1$, this would reduce to $|\Sigma|$ parameters. However, verifying the goodness of fit for a multi-parameter Dirichlet distribution is a challenging problem that makes this approach difficult.

In this dissertation, the data is fitted to a Gaussian distribution, as given in Eq. 3.2. A co-ordinate transformation can be used to diagnolize the co-variance matrix to make the individual components of the vector orthogonal (and hence, independent). However, in doing so, the physical attributes of the state probabilities would cease to hold. In the case that the fluctuations of the first $|\Sigma| - 1$ components of the probability vector are assumed to be independent, Eq. 3.7 can be simplified further. That is, the joint density function of the Gaussian random vector $\mathbf{p}$ is reduced to the product of individual Gaussian distributions of the random variables p_j. That is,

$$f_{\Omega|\mathbf{q}}\left(\underline{s}|\mathbf{p}\right)\big|_{\underline{s}=s^k} \approx \kappa \frac{\prod_{j=1}^{|\Sigma|-1} f_{q_j|\Omega}\left(p_j|s^k\right)}{\sum_{\tilde{\underline{s}}\in\mathcal{S}} \prod_{j=1}^{|\Sigma|-1} f_{q_j|\Omega}\left(p_j|\tilde{\underline{s}}^k\right)} \tag{3.8}$$

The density functions in the numerator and denominator of Eq. (3.8) are obtained from Eqs. (3.7) and (3.3), which were determined in the training phase. A most likely estimate of the parameter vector $\underline{s}$ is obtained from the probabilistic map in Eq. (3.8). It should be noted that the nature of the density function $f_{q_j|\Omega}\left(p_j|s^k\right)$ does not depend on the constant κ.

The probability mass functions are obtained by evaluating the probability density function in Eq. (3.7) at points $s^k \in \mathcal{S}$.

$$\begin{aligned} P(\underline{s}^k|\mathbf{p}) &\triangleq \frac{f_{\Omega|\mathbf{q}}\left(\underline{s}^k|\mathbf{p}\right)}{\sum_{j=1}^{|\mathcal{S}|} f_{\Omega|\mathbf{q}}\left(\underline{s}^j|\mathbf{p}\right)} \\ &\approx \frac{f_{\mathbf{q}|\Omega}\left(\mathbf{p}|s^k\right)}{\sum_{j=1}^{|\mathcal{S}|} f_{\mathbf{q}|\Omega}\left(\mathbf{p}|\underline{s}^j\right)} \end{aligned} \tag{3.9}$$

Substitution of Eqs. (3.3) and (3.8) in Eq. (3.9) yields

$$P(\underline{s}^k|\mathbf{p}) \approx \frac{\prod_{j=1}^{|\Sigma|-1} \frac{1}{\sqrt{2\pi\sigma_j^2(s^k)}} \exp\left(-\frac{\left(p_j - m_j(s^k)\right)^2}{2\sigma_j^2(s^k)}\right)}{\sum_{l=1}^{|\mathcal{S}|} \prod_{j=1}^{|\Sigma|-1} \frac{1}{\sqrt{2\pi\sigma_j^2(s^l)}} \exp\left(-\frac{\left(p_j - m_j(s^l)\right)^2}{2\sigma_j^2(s^l)}\right)} \tag{3.10}$$

where the probability vector $\mathbf{p} \equiv [p_1 \cdots p_{|\Sigma|}]$ is calculated from the observed time series data; and the remaining parameters are already evaluated in the training phase.

Estimated mean $\underline{\hat{s}}$ and estimated covariance matrix $\hat{C}_{\underline{s}}$ of the parameter vector $\underline{s}$ are obtained directly from Eq. (3.10) as

$$\underline{\hat{s}}(\mathbf{p}) \triangleq \sum_{k=1}^{|\mathcal{S}|} \underline{s}^k \; P(\underline{s}^k|\mathbf{p}) \tag{3.11}$$

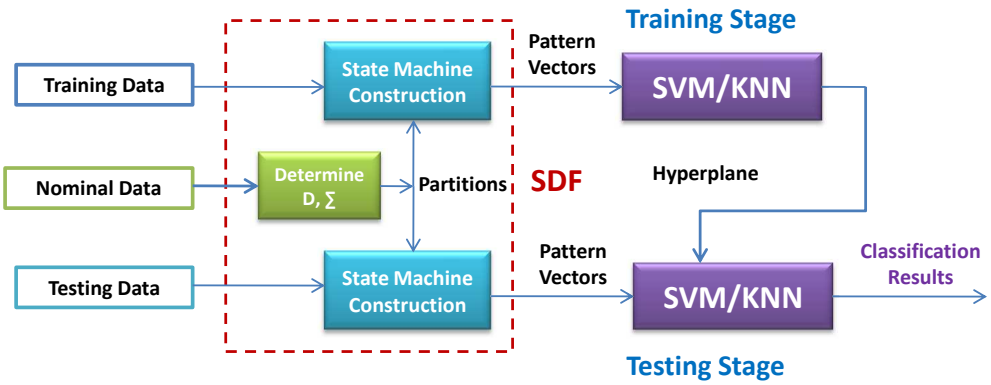
$$\hat{C}_{\underline{s}}(\mathbf{p}) \triangleq \sum_{k=1}^{|\mathcal{S}|} \left(\underline{s}^k - \underline{\hat{s}}(\mathbf{p})\right) \; P(\underline{s}^k|\mathbf{p}) \; \left(\underline{s}^k - \underline{\hat{s}}(\mathbf{p})\right)^T \tag{3.12}$$

Since the statistical information is available in the form of probability mass functions, the third and higher moments of the parameter vector can be estimated in a similar way; however, third and higher moments are redundant because the inherent distribution is assumed to have a Gaussian structure that carries full statistical information in the first two moments.

3.6 Multiple parameter classification using Symbolic Dynamics

The problem of estimation of multiple parameters can be simplified to that of a classification problem.

The flowchart for this approach is represented in Figure 3.6. In this dissertation, the k nearest neighbors technique is used. Other techniques such as support vector machines or Naive Bayes classifiers can also be used effectively. The different classification approaches that can be used are outlined in Appendix C. SDF is used to generate a feature vector that is unique for each of the classes, and is



robust to noise.

As in the previous section, let $\mathcal{S}$ denote the collection of (finitely many) points in the n-dimensional parameter space, where the positive integer n is the number of parameters that are to be estimated. That is, $\mathcal{S} = \left\{\underline{s}^1, \underline{s}^2, \cdots, \underline{s}^{|\mathcal{S}|}\right\}$, on which the training process is executed. Let Ω be the convex hull of $\mathcal{S}$, which represents the range over which the parameters take values. It is noted that Ω is a convex and compact subset of the separable space $\mathbb{R}^n$. In this approach, the hull Ω is partitioned into finitely many regions. The forward and inverse problems get simplified as follows - In the forward problem, it is required to obtain statistics for each of the regions of the hull Ω. In the inverse problem, it is required to determine which region has the closest matching pattern to the pattern obtained during testing.

Chapter 4

Improving estimation using Multiple sensors and sensor selection

4.1 Introduction

In a complex system such as an aircraft gas-turbine engine, the patterns generated from a single sensor may not carry sufficient information to identify multiple parameters/faults because different combinations of component faults may generate similar signatures in a particular sensor observation. A key contribution of this section is the compression of data into pattern vectors of low-dimension for feature-level sensor fusion as needed for onboard vehicle health monitoring and resilient control.

4.2 Multiple Sensor Methodology using covariance matrix

4.2.1 Problem Statement

In general, the fault scenarios in major engine components can be categorized into three different types based on their mode of occurrence. These types are:

1. Gradual deterioration,

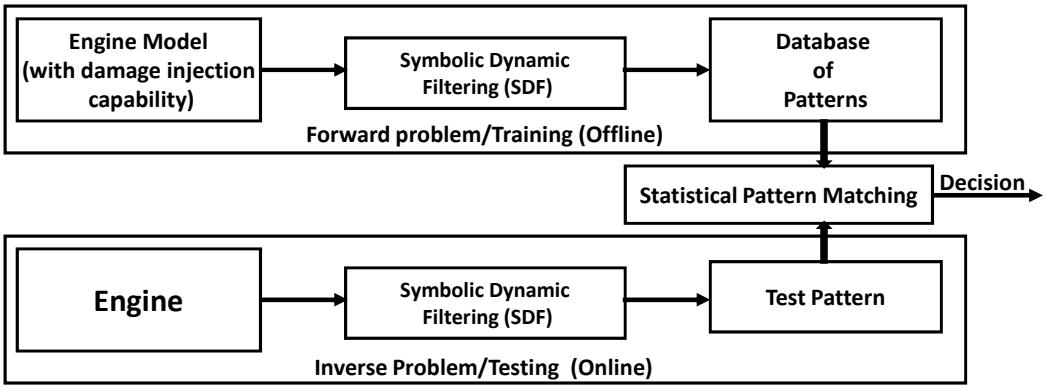


Figure 4.1. Outline of the fault estimation procedure

2. Intermittent faults, and

3. Abrupt large faults.

However, no matter what type of fault occurs in a particular component, the problem can be reduced to a parameter identification problem from the point of view of fault estimation as presented in Chapter 3.

Let $\mathcal{S}$ denote a collection of (finitely many) data points in the n-dimensional parameter space, where the positive integer n is the number of parameters that are to be estimated. That is, $\mathcal{S} = \left\{\boldsymbol{s}^0, \boldsymbol{s}^1, \cdots, \boldsymbol{s}^{|\mathcal{S}|-1}\right\}$, on which the training process is executed. In the context of gas-turbine engines, $\boldsymbol{s}^k$ signifies a particular faulty condition in the set of fault conditions $\mathcal{S}$ under consideration. Let $\boldsymbol{s}^0$ denote the nominal condition of the engine, and $\mathcal{Y}$ be the set of sensors for the engine system consisting of sensors y_j for $j = 1, 2, ..., |\mathcal{Y}|$. Let Ω be the convex hull of $\mathcal{S}$, which represents the range over which the parameters take values. It is noted that Ω is a convex and compact subset of the separable space $\mathbb{R}^n$. The problem at hand is to statistically estimate fault condition $\boldsymbol{s} \in \Omega$, given an experimental data set Υ, i.e., to identify the conditional probability density $f(\boldsymbol{s}|\Upsilon)$. It is noted that $\boldsymbol{s} \in \Omega$ may not be one of the points in set $\mathcal{S}$.

The multiple-fault estimation procedure is divided into two steps, which are: (i) Forward Problem/Training, and (ii) Inverse Problem/Testing, as shown in Fig. 4.1. The following subsections 4.2.2 and 4.2.3 describe the two steps in detail.

4.2.2 Forward Problem/Training with Multiple Sensors

In the forward problem, a database of patterns is created at parameter values, $\boldsymbol{s}^k, \forall k = 0, 1, \cdots, (|\mathcal{S}| - 1)$, by collecting time-series data from sensors $y_j \in \mathcal{Y}$, as shown in Fig. 4.1. Generation of statistical patterns from time series data is posed as a two-scale problem [2] [11]. The *fast scale* is related to the response time of the process dynamics, over the span of which the process is assumed to be quasi-stationary. The *slow scale* is related to the time span over which deviations (e.g., parametric or non-parametric changes) may occur and exhibit non-stationary dynamics. In the present context, time-series data are collected with the system being quasi-stationary at a particular slow-scale epoch $\boldsymbol{s}^k$. The procedural steps of the forward problem are presented below.

- *Time series data acquisition on the fast scale from the available sensors*: Time series data sets from each sensor $y_j \in \mathcal{Y}$ are collected for each epoch $\boldsymbol{s}^k \in \mathcal{S}$.

- *Wavelet/Hilbert transform pre-processing of the time-series data*: The wavelet or Hilbert transforms largely alleviate the difficulties of phase-space partitioning and are particularly effective with noisy data from high-dimensional dynamical systems [3] [64].

- *Maximum Entropy Partitioning of the transformed space at the reference condition of epoch* $\mathbf{s}^0$: This step enables transformation of the pre-processed time series data from the continuous domain to the symbol domain [2] by partitioning the transformed phase space, where the data set from each sensor $y_j, j = 1, \cdots, |\mathcal{Y}|$, has its own alphabet; for each sensor, a specific symbol is assigned to each partition segment from the respective alphabet. Maximum entropy partitioning [3] [64] is constructed separately for different sensor data sets at epoch $\boldsymbol{s}^0$. These partitions are kept invariant for analysis at subsequent epochs $\boldsymbol{s}^1, \boldsymbol{s}^2, ..., \boldsymbol{s}^{|\mathcal{S}|-1}$ of respective sensor data.

- *Construction of a probabilistic finite state automaton (PFSA) at the reference condition* $\mathbf{s}^0$ *and Computation of state probabilities*: *PFSA* are constructed for every sensor data at epoch $\boldsymbol{s}^0$ and their structures remain invariant for subsequent epochs of each sensor data. Let the N_j, be the number of states

in the *PFSA* corresponding to the sensor $y_j, j = 1, \cdots, |\mathcal{Y}|$. The sum of the probabilities of all states is equal to unity, i.e., $\sum_{i=1}^{N_j} p_i^j(\boldsymbol{s}^k) = 1 \ \forall j \in \{1, \cdots, |\mathcal{Y}|\} \ \forall k \in \{0, \cdots, |\mathcal{S}|-1\}$, where $p_i^j(\boldsymbol{s}^k)$ denotes the probability of the i^{th} state of the PFSA constructed fron time series of j^{th} sensor at epoch $\boldsymbol{s}^k$; at most $N_j - 1$, out of the N_j elements of the state probability vector can be independent. Therefore, the pattern for each sensor labeled by $j = 1, \cdots, |\mathcal{Y}|$ is represented by a $(N_j - 1)$-dimensional row vector $\boldsymbol{p}^j \triangleq \left[p_1^j \cdots p_{N_j-1}^j\right] \ \forall j \in \{1, \cdots, |\mathcal{Y}|\}$; this notation holds for all epochs $\boldsymbol{s}^k \ \forall k \in \{0, \cdots, |\mathcal{S}| - 1\}$.

- *Computation of the pattern vectors*: A concatenated reference pattern vector $\mathcal{P}$ is generated whose elements represent state occupation probabilities of the *PFSA* at the reference condition $\underline{s}^0$ for data from the respective probability vectors $\mathbf{p}^l(\underline{s}^0)$ from all the sensors $y_\ell \in \mathcal{Y}$. Thus, $\mathbf{p}(\underline{s}^0)$ has an element $p_j^l(\underline{s}^0)$, where the superscript $l \in \{1, 2, ..., |\mathcal{Y}|\}$ corresponds to the sensor $y_l \in \mathcal{Y}$ and the subscript $j \in \{1, 2, ..., N_l\}$ corresponds to the j^{th} state of the PFSA generated from the time series data of the l^{th} sensor at epoch s^0. Therefore, the total number of elements in $\mathbf{p}(\underline{s}^0)$ is $N = N_1 + N_2 + \cdots + N_{|\mathcal{Y}|}$. Similarly, $\mathbf{p}(\underline{s}^1), \mathbf{p}(\underline{s}^2), ..., \mathbf{p}(\underline{s}^i), ..., \mathbf{p}(\underline{s}^{|\mathcal{S}|})$ are generated at epochs $\underline{s}^1, \underline{s}^2, ..., \underline{s}^{|\mathcal{S}|-1}$ from the respective symbol sequences based on the same structure of the *PFSA* constructed at epoch $\underline{s}^0$. Note that the structure of the *PFSA* at all epochs for a particular sensor is identical while the pattern vectors $\mathbf{p}(\underline{s}^k)$ are possibly different at different $\underline{s}^k \in \mathcal{S}$ due to parametric changes in the process.

- *Construction of the pattern database*: A reference pattern array $\mathcal{P}(\boldsymbol{s}^0)$ is constructed by vertical stacking of the reference row vectors, $\boldsymbol{p}^j(\boldsymbol{s}^0), \ j \in \{1, \cdots, |\mathcal{Y}|\}$, as shown below.

$$\mathcal{P}(\underline{s}^0) \triangleq \left\{ \begin{array}{l} \mathbf{p}^1(\underline{s}^0) \\ \mathbf{p}^2(\underline{s}^0) \\ \cdots \\ \cdots \\ \mathbf{p}^{|\mathcal{Y}|}(\underline{s}^0) \end{array} \right\} = \left\{ \begin{array}{l} \left[p_1^1(\underline{s}^0) \cdots p_{N_1-1}^1(\underline{s}^0)\right] \\ \left[p_1^2(\underline{s}^0) \cdots p_{N_2-1}^2(\underline{s}^0)\right] \\ \cdots \\ \cdots \\ \left[p_1^{|\mathcal{Y}|}(\underline{s}^0) \cdots p_{N_{|\mathcal{Y}|}-1}^{|\mathcal{Y}|}(\underline{s}^0)\right] \end{array} \right\}$$

$$\mathcal{P}(\boldsymbol{s}^0) \triangleq \left\{ \begin{array}{l} \boldsymbol{p}^1(\boldsymbol{s}^0) \\ \boldsymbol{p}^2(\boldsymbol{s}^0) \\ \cdots \\ \cdots \\ \boldsymbol{p}^{|\mathcal{Y}|}(\boldsymbol{s}^0) \end{array} \right\} = \left\{ \begin{array}{l} p_1^1(\boldsymbol{s}^0) \cdots p_{N_1-1}^1(\boldsymbol{s}^0) \\ p_1^2(\boldsymbol{s}^0) \cdots p_{N_2-1}^2(\boldsymbol{s}^0) \\ \cdots \\ \cdots \\ p_1^{|\mathcal{Y}|}(\boldsymbol{s}^0) \cdots p_{N_{|\mathcal{Y}|}-1}^{|\mathcal{Y}|}(\boldsymbol{s}^0) \end{array} \right\}$$

Note that the individual rows in the array $\mathcal{P}$ may have different lengths because the PFSA corresponding to different sensors may have different state cardinalities; hence, $\mathcal{P}$ should not be viewed as a matrix but it is a two-dimensional array of positive fractions, where the total number of elements is $(N_1 + \cdots + N_{|\mathcal{Y}|} - |\mathcal{Y}|)$. Similarly, $\mathcal{P}(\boldsymbol{s}^1), \mathcal{P}(\boldsymbol{s}^2), ..., \mathcal{P}(\boldsymbol{s}^{|\mathcal{S}|-1})$ are computed at epochs $\boldsymbol{s}^1, \boldsymbol{s}^2, ..., \boldsymbol{s}^{|\mathcal{S}|-1}$ from the respective patterns. Note that the structure of the *PFSA* at all epochs for a particular sensor is identical while the pattern arrays $\mathcal{P}(\boldsymbol{s}^k)$ are possibly different at different $\boldsymbol{s}^k \in \mathcal{S}$ due to parametric or non-parametric changes in the process.

- *Computation of pattern statistics*: Different units of identically manufactured engines are different in behavior or performance; this inevitable uncertainty is modeled as the process noise. Therefore, several runs are performed for each fault condition, with a certain value of process noise along with an *a priori* determined sensor noise (e.g., calculated from instrumentation manufacturer's specifications) to obtain the pattern vector statistics. Let the pattern array $\mathcal{P}(\boldsymbol{s}^k)$ be modeled as a random array $\mathcal{Q}(\boldsymbol{s}^k)$, whose elements are $q_i^l(\boldsymbol{s}^k)$ that is constructed from the ensemble of realizations $p_i^l(\boldsymbol{s}^k)$. Considering up to second order statistics, elements of the random array $\mathcal{Q}(\boldsymbol{s}^k)$ are modeled to have multivariate structures from the perspectives of state machine construction in the SDF setting, as explained later in Remark 4.2.1. Thus, for each epoch $\boldsymbol{s}^k$, a mean pattern vector $\mu(\boldsymbol{s}^k)$ and a corresponding covariance matrix $\Gamma(\boldsymbol{s}^k)$ of the pattern are calculated from the elements of $\mathcal{Q}(\boldsymbol{s}^k)$. An element of $\mu(\boldsymbol{s}^k)$ is expressed as $m_i^l(\boldsymbol{s}^k)$, $\forall l \in \{1, 2, \cdots, |\mathcal{Y}|\}$ and $\forall i \in \{1, 2, ..., N_l - 1\}$, which signifies the mean values of $p_i^l(\boldsymbol{s}^k)$ generated from the data sets of different runs. Similarly, an element of the covariance matrix $\Gamma(\boldsymbol{s}^k)$ is expressed as $\gamma_{ij}^{l\ell}(\boldsymbol{s}^k)$, $\forall l, \ell \in \{1, 2, ..|\mathcal{Y}|\}$ and $\forall i \in \{1, 2, ..., N_l - 1\}$,

and $\forall j \in \{1, 2, ..., N_\ell - 1\}$, which signifies the value of cross-covariance between $p_i^l(\boldsymbol{s}^k)$ and $p_j^\ell(\boldsymbol{s}^k)$, which is also generated from the data sets of different runs. Note that, for $l = \ell$, the covariance matrix terms yield correlation among the states i and j of the *PFSA* generated from the same sensor data and, for $l \neq \ell$, the covariance matrix terms yield correlation among the states i and j of different *PFSA* corresponding to different sensors.

For the purpose of book-keeping in statistical calculations, each of the (two-dimensional) arrays $\mathcal{P}(\boldsymbol{s}^k)$ is rearranged as a single row vector $\mathbf{p}(\boldsymbol{s}^k)$ by horizontally concatenating the row vectors $\boldsymbol{p}^j(\boldsymbol{s}^k),\ j \in \{1, \cdots, |\mathcal{Y}|\}$, i.e., the random pattern array $\mathcal{Q}(\boldsymbol{s}^k)$ is rearranged as the random pattern vector $\mathbf{q}(\boldsymbol{s}^k)$. The mean pattern vector $\mu(\boldsymbol{s}^k)$ and covariance matrix $\Gamma(\boldsymbol{s}^k)$ are constructed correspondingly. The covariance matrix $\Gamma(\boldsymbol{s}^k)$ is comprised of several blocks of elements. The square diagonal blocks correspond to the covariance among states of same sensor data, where as the off-diagonal possibly non-square (due to possible different alphabet size for different sensor data) blocks correspond to the covariance among states of different sensor data.

Remark 4.2.1. *The underlying dynamical system is modeled as an irreducible Markov process via* SDF*, where the state probability vector is the sum-normalized eigenvector of the state transition matrix corresponding to the unique unity eigenvalue. Hence, no element in the state probability vector is either* 0 *or equal to* 1*. However, due to process noise and sensor noise, the random vector* $\mathbf{q}(\mathbf{s}^k)$ *fluctuates around its mean* $\mu(\mathbf{s}^k)$*. Analyzing the experimental data, the terms of the covariance matrices of the random vectors* $\mathbf{q}(\mathbf{s}^k)$ *was found to be very small compared to the mean. Therefore, a parametric or non-parametric two-sided uni-modal distribution should be adequate to model the random vector* $\mathbf{q}(\mathbf{s}^k)$*. The choice of Gaussian distribution for* $\mathbf{q}$ *would facilitate estimation of the statistical parameters and involve only second order statistics. Also, the elements of* $\mathbf{q}(\mathbf{s}^k)$ *have to be positive, which is made possible by truncating the far end of the Gaussian distribution tail on the left side. The goodness of fit of the distribution as Gaussian still remains valid at a very high significance level. For a particular sensor* y_j *summation of the elements* $q_i^j(\mathbf{s}^k)$*,* $\forall i \in \{1, 2, ..., N_j\}$ *has to be unity, which is achieved by sum-normalization.*

The (jointly Gaussian) conditional probability distribution of a random pattern vector $\boldsymbol{q}$ is given as

$$f_{\boldsymbol{q}|\Omega}\left(\mathbf{p}|\boldsymbol{s}^k\right) = \frac{1}{(2\pi)^{N/2}|\Gamma(\boldsymbol{s}^k)|^{1/2}} \cdot \exp\Big(-\frac{1}{2}(\boldsymbol{p}-\mu(\boldsymbol{s}^k))(\Gamma(\boldsymbol{s}^k))^{-1}(\boldsymbol{p}-\mu(\boldsymbol{s}^k))^T\Big) \tag{4.1}$$

where $N = N_1 + \cdots + N_{|\mathcal{Y}|} - |\mathcal{Y}|$.

4.2.3 Inverse Problem/Testing with Multiple Sensors

The objective here is to identify the probabilistic location of the fault in the multi-dimensional parameter space, i.e., identification of the unknown parameter vector $\boldsymbol{s} \in \Omega$; however, it is possible that $\boldsymbol{s} \notin \mathcal{S}$. Therefore, for a particular test case, time series data are collected from different sensors. The data are analyzed using the same symbolic dynamic filter constructed in the forward problem/training (see Section 3.5.1), and the resulting row vector $\boldsymbol{p}$ is a realization of a random pattern vector $\boldsymbol{q}$. The density function $f_{\Omega|\boldsymbol{q}}(\boldsymbol{s}|\boldsymbol{p})$ is obtained as

$$\begin{aligned} f_{\Omega|\boldsymbol{q}}(\boldsymbol{s}|\boldsymbol{p}) &= \frac{f_{\boldsymbol{q}|\Omega}\left(\boldsymbol{p}|\boldsymbol{s}\right) f_{\Omega}(\boldsymbol{s})}{f_{\boldsymbol{q}}\left(\boldsymbol{p}\right)} \\ &= \frac{f_{\boldsymbol{q}|\Omega}\left(\boldsymbol{p}|\boldsymbol{s}\right) f_{\Omega}(\boldsymbol{s})}{\int_{\Omega} f_{\boldsymbol{q}|\Omega}\left(\boldsymbol{p}|\tilde{\boldsymbol{s}}\right) f_{\Omega}(\tilde{\boldsymbol{s}}) d\tilde{\boldsymbol{s}}} \end{aligned} \tag{4.2}$$

In the absence of *a priori* information, an assumption is made that all operating conditions are equally likely, i.e., $f_{\Omega}(\boldsymbol{s}) = f_{\Omega}(\tilde{\boldsymbol{s}})\ \forall \tilde{\boldsymbol{s}},\ \tilde{\boldsymbol{s}} \in \Omega$. With this assumption of uniform probability, Eq. (4.2) reduces to

$$f_{\Omega|\boldsymbol{q}}(\boldsymbol{s}|\boldsymbol{p}) = \frac{f_{\boldsymbol{q}|\Omega}\left(\boldsymbol{p}|\boldsymbol{s}\right)}{\int_{\Omega} f_{\boldsymbol{q}|\Omega}\left(\boldsymbol{p}|\tilde{\boldsymbol{s}}\right) d\tilde{\boldsymbol{s}}} \tag{4.3}$$

It is noted that accuracy of the above distribution would be improved if the actual prior mapping, i.e., $f_{\Omega}(\boldsymbol{s})$ is known. The integral in the denominator of Eq. (4.3) is approximated by a Reimann sum as

$$f_{\Omega|\boldsymbol{q}}(\boldsymbol{s}|\boldsymbol{p}) \approx \kappa \frac{f_{\boldsymbol{q}|\Omega}\left(\boldsymbol{p}|\boldsymbol{s}\right)}{\sum_{\mathcal{S}} f_{\boldsymbol{q}|\Omega}\left(\boldsymbol{p}|\widetilde{\boldsymbol{s}}\right)} \tag{4.4}$$

where κ is a constant. This approximation converges to the exact solution as the training set $\mathcal{S}$ approaches a countable dense subset of $\Omega \subset \mathbb{R}^n$.

The density function in Eq. (4.4) is now sampled at the points $\boldsymbol{s}^k$ in the training set $\mathcal{S}$ and the following sampled density is constructed as to yield

$$f_{\Omega|\boldsymbol{q}}(\boldsymbol{s}|\boldsymbol{p})\big|_{\boldsymbol{s}=\boldsymbol{s}^k} \approx \kappa \frac{f_{\boldsymbol{q}|\Omega}\left(\boldsymbol{p}|\boldsymbol{s}^k\right)}{\sum_{\widetilde{\boldsymbol{s}}\in\mathcal{S}} f_{\boldsymbol{q}|\Omega}\left(\boldsymbol{p}|\widetilde{\boldsymbol{s}}\right)} \; \forall \boldsymbol{s}^k \in \mathcal{S} \tag{4.5}$$

The density functions in the numerator and denominator of Eq. (4.5) are obtained from Eq. (4.1), which were determined in the training phase. It is noted that the nature of the density function $f_{\Omega|\boldsymbol{q}}\left(\boldsymbol{s}^k|\boldsymbol{p}\right)$ does not depend on the constant κ.

The probability mass functions are obtained by evaluating the probability density function in Eq. (3.8) at points $\boldsymbol{s}^k \in \mathcal{S}$.

$$\begin{aligned} P(\boldsymbol{s}^k|\boldsymbol{p}) &\triangleq \frac{f_{\Omega|\boldsymbol{q}}\left(\boldsymbol{s}^k|\boldsymbol{p}\right)}{\sum_{j=1}^{|\mathcal{S}|} f_{\Omega|\boldsymbol{q}}\left(\boldsymbol{s}^j|\boldsymbol{p}\right)} \\ &\approx \frac{f_{\boldsymbol{q}|\Omega}\left(\boldsymbol{p}|\boldsymbol{s}^k\right)}{\sum_{j=1}^{|\mathcal{S}|} f_{\boldsymbol{q}|\Omega}\left(\boldsymbol{p}|\boldsymbol{s}^j\right)} \end{aligned} \tag{4.6}$$

which is expressed in terms of Eq. (4.1) as

$$P(\boldsymbol{s}^k|\boldsymbol{p}) \approx \frac{\frac{1}{(2\pi)^{N/2}|\Gamma(\boldsymbol{s}^k)|^{1/2}} \exp\left(X(\boldsymbol{s}^k)\right)}{\sum_{l=1}^{|\mathcal{S}|} \frac{1}{(2\pi)^{N/2}|\Gamma(\boldsymbol{s}^l)|^{1/2}} \exp\left(X(\boldsymbol{s}^l)\right)} \tag{4.7}$$

where $X(\bullet) = -\frac{1}{2}[\boldsymbol{p} - \mu(\bullet)][\Gamma(\bullet)]^{-1}[\boldsymbol{p} - \mu(\bullet)]^T$

The above equation signifies a statistical pattern matching by calculating the Mahalnobis distance [48] between the test and the training patterns; therefore, smaller the Mahalnobis distance, better is the match between these two patterns.

It has been observed from experimental data that fluctuations of the pattern vectors are very weakly correlated among different symbols and different sensors. Therefore, the jointly Gaussian distribution of all $f_{\boldsymbol{q}|\Omega}\left(\boldsymbol{p}|\boldsymbol{s}^k\right)$'s can be reduced to

the product of individual Gaussian distributions $f_{\boldsymbol{q}|\Omega}\left(p_i^j|\boldsymbol{s}^k\right)$ of different symbols $\forall j \in \{1,2,..,|\mathcal{Y}|\}$ and $\forall i \in \{1,2,...,N_j-1\}$. Therefore, instead of using the multi-variate jointly Gaussian distribution, univariate Gaussian distribution is used for each symbol, (the variance being the corresponding diagonal element of the covariance matrix) to calculate $P(\boldsymbol{s}^k|\boldsymbol{p})$. Thus, Eq. (4.7) is expressed as follows.

$$P(\boldsymbol{s}^k|\boldsymbol{p}) \approx \frac{\prod_j \prod_i \frac{1}{(2\pi)^{1/2}(\gamma_{ii}^{jj}(\boldsymbol{s}^k))^{1/2}} \exp\left(X_i^j(\boldsymbol{s}^k)\right)}{\sum_{l=1}^{|\mathcal{S}|} \prod_j \prod_i \frac{1}{(2\pi)^{1/2}(\gamma_{ii}^{jj}(\boldsymbol{s}^l))^{1/2}} \exp\left(X_i^j(\boldsymbol{s}^l)\right)} \quad \begin{array}{c} \forall j \in \{1,2,..,|\mathcal{Y}|\} \text{ and } \forall i \in \{1,2,...,N_j-1\}, \\ \\ \end{array} \tag{4.8}$$

where $X_i^j(\bullet) \triangleq -\frac{1}{2}[p_i^j - m_i^j(\bullet)][\gamma_{ii}^{jj}(\bullet)]^{-1}[p_i^j - m_i^j(\bullet)]$

Once the probability mass function $P(\boldsymbol{s}^k|\boldsymbol{p})$ is obtained, there can be different estimates $\hat{\boldsymbol{s}} \in \Omega$ depending upon the cost function of estimation. For example, the median of the distribution yields the estimated value by minimizing the root mean square value of the deviations. Again, most likely parameter value can be obtained from the mode of the distribution. In this paper, estimated mean is considered which minimizes the average of the square of the absolute deviations around the estimated point. Estimated mean $\hat{\boldsymbol{s}}$ and estimated covariance matrix $\hat{\mathbf{C}}_{\boldsymbol{s}}$ of the parameter (column) vector $\boldsymbol{s}$ are obtained directly from $P(\boldsymbol{s}^k|\boldsymbol{p})$ as

$$\hat{\boldsymbol{s}}(\boldsymbol{p}) \triangleq \sum_{k=1}^{|\mathcal{S}|} \boldsymbol{s}^k \; P(\boldsymbol{s}^k|\boldsymbol{p}) \tag{4.9}$$

$$\hat{\mathbf{C}}_s(\boldsymbol{p}) \triangleq \sum_{k=1}^{|\mathcal{S}|} \left(\boldsymbol{s}^k - \hat{\boldsymbol{s}}(\boldsymbol{p})\right) \; P(\boldsymbol{s}^k|\boldsymbol{p}) \; \left(\boldsymbol{s}^k - \hat{\boldsymbol{s}}(\boldsymbol{p})\right)^T \tag{4.10}$$

Since the statistical information is available in the form of probability mass functions, the third and higher moments of the parameter vector can be estimated in a similar way; however, third and higher moments are redundant because the inherent distribution is assumed to have a Gaussian structure that carries full statistical information in the first two moments.

4.3 Sensor selection framework

Chapter 5

Description of Test Beds

5.1 Description of Duffing Experiment

This chapter provides a description of the electronic circuit for simulating the Duffing Equation. Experiments were conducted on the test bed [72] described in Appendix B. For the purpose of the experiment, a parameter is allowed to vary continuously by a small amount. This parametric change simulates a change in the health condition of the dynamical system. The objective of the experiment is to distinguish these small variations and estimate the value of the parameter. The experiments are repeated a number of times, by replacing a component (resistor) in the experimental setup each time with a similar component. This is done to establish robustness of the estimate with respect to component uncertainties.

5.1.1 Duffing System Analysis

The Duffing equation [73] is a second-order forced differential equation with a cubic non-linearity . It is given by

$$\frac{d^2x(t)}{dt^2} + \beta\frac{dx}{dt} + x(t) + x^3(t) = A\cos(\omega t) \tag{5.1}$$

5.1.2 Single Parameter experiment

The dissipation parameter $\beta(t_s)$, realized as a resistance in the circuit, varies with the slow time t_s and is treated as a constant in the fast time scale at which the

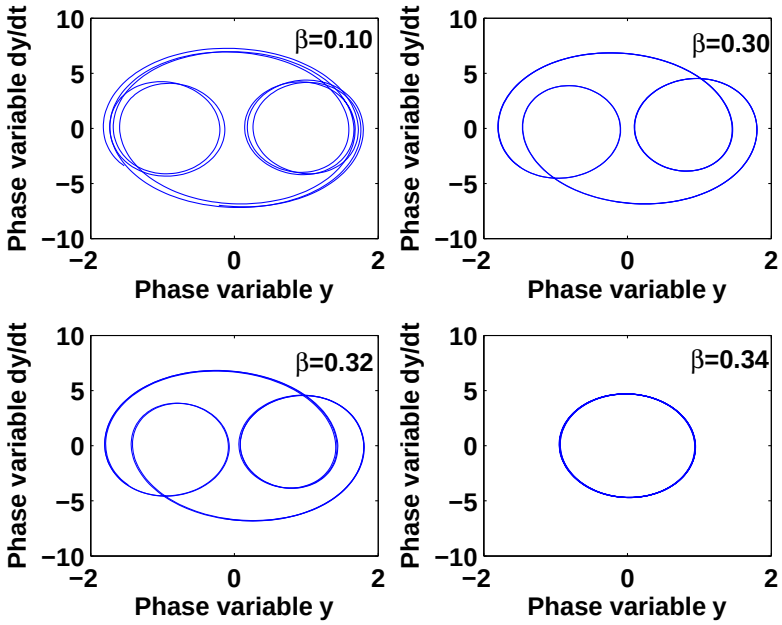


Figure 5.1. Phase Plots for the Electronic Circuit

dynamical system is excited. The goal is to detect, at an early stage, changes in $\beta(t_s)$, which are associated with the anomaly. The first task is selection of appropriate input stimuli. For illustration purposes, we show the response of a stimulus with amplitude $A = 22$ and frequency $\omega = 5$. Changes in the stationary behavior of the electronic circuit in Figure 5.1 take place starting from $\beta(t_s) = 0.10$, with a significant disruption occurring in the narrow range of 0.32 to 0.34. The stationary behavior of the system response for this input stimulus is obtained for several values of in the range of 0.10 to 0.40. The four plates in Figure 5.1 exhibit four phase plots for the values of the parameter at $0.10, 0.30, 0.32$, and 0.34, respectively. Each plate in Figure 5.1 relates the phase variable of electrical charge that is proportional to the voltage across one of the capacitors in the electronic circuit, with its time derivative (i.e., the instantaneous current). While a small difference between the phase plots for $\beta = 0.1$ and $\beta = 0.3$ is noticeable, there is no clearly visible difference between the plots for $\beta = 0.30$ and $\beta = 0.32$ in Figure 5.1. However, the phase plots for $\beta = 0.32$ and $\beta = 0.34$ display a very large difference, indicating period doubling possibly due to onset of bifurcation.

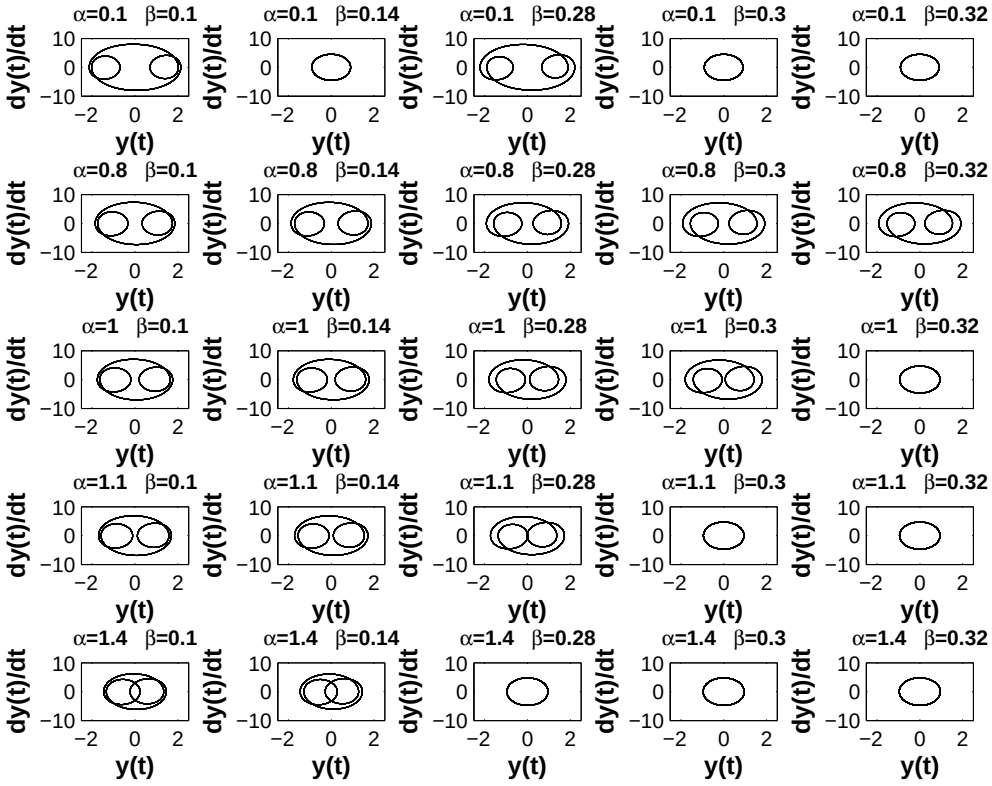


Figure 5.2. Phase Plots for the Multi-Parameter Experiment

5.1.3 Multiple Parameter Experiment

In this experiment, the dissipation parameters are chosen as β and α_1. The input stimulus are chosen as $A = 5$ and $\omega = 5$. The stationary behavior of the system is obtained with several combinations of values, with β ranging from 0.10 to 0.40, and α_1 ranging from 0.10 to 1.50. The plots are shown in Figure 5.2. The third line of plots corresponds to a value of $\alpha_1 = 1.0$ which is exactly the same as considered in Section 5.1.2. It can be seen that increasing (decreasing) the value of α_1 causes an early (late) onset of bifurcation. Also, a bifurcation is associated simply with a rise in α_1 for low values of β as can be seen in the first row of plots. It is also clear that there is very little visible difference in most plots before the onset of bifurcation. The challenge of this experiment is to determine the values of *both* α_1 and β by looking at data gathered at a slow time scale.

5.2 Description of Aircraft Engine Simulation Test Bed

The aircraft engine simulation test bed consists of three networked computers using the client/server concept. One of the three computers will host the propulsion-driven system for health monitoring of the engines. The other two computers execute separate copies (which may or may not be different depending on the health of the individual engines) of the gas turbine engine model including its continuously varying gain-scheduled feedback control system. These models are described by ordinary differential equations and supporting algebraic equations and look-up tables [74].

The test bed is capable of simulating steady-state and dynamical conditions for individual engines under different operating conditions. Each of the engine simulation models integrates the time-driven continuous dynamics and communicates through continuous-to-discrete interfaces. This software architecture is flexible to adapt different models and controller designs for other types of engine systems. Each major function in the simulation program has a modular structure as implemented on the three networked computers of the simulation test bed. The $C++$ code is superimposed on the existing FORTRAN simulation code of the turbofan engine model to be able to utilize $C++$ network communication routines. Specifically, the $C++$ wrapper program interfaces the major inputs and outputs of the FORTRAN simulation code and works as a data acquisition module. This approach takes advantage of the available FORTRAN models as individual parts of the integrated $C++$ program without making any significant changes.

The FORTRAN code of the turbofan engine simulation program, which consists of high-order nonlinear differential and difference equations and supporting algebraic equations, has been designed for both steady-state and transient operations of a generic jet engine*. This simulation code is a stand-alone program with a gain-scheduled feedback controller. The engine simulation model provides various sensor data (e.g., combustion chamber temperature and high-pressure and low-pressure turbine speeds) together with other critical information (e.g., simulation step size and simulation cycle number), which are collected by the $C++$ wrapper program and exchanged with the FDI computer through a typical message

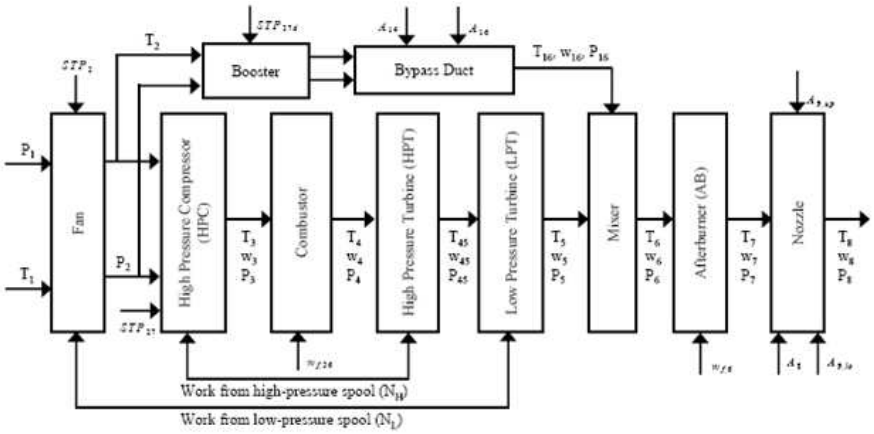


Figure 5.3. Schematic of turbofan engine model with labeled actuators (italics) and sensors

application protocol interface (API) communication routine. This communication protocol sends and receives message packages through TCP and/or UDP networks. The delay in this protocol interface is mainly due to the network communications and the typical value is found to be less than a fraction of millisecond. Since engine simulations use integration step sizes in the order of $20ms$, the communication delays do not have a significant bearing on performance of the FDI algorithms that are implemented in the MATLAB 7.1 environment.

5.2.1 Dynamic Model of the Turbofan Engine

Details of the mathematical model of the two-spool, low bypass turbofan engine is reported in the cited references [75, 76]. The structure of the engine model, simulated as a FORTRAN program, is depicted in Figure 5.3 that, along with a diagram of the implemented engine model, lists the actuators. The actuators are: (i) Fan variable inlet stator vane angle (STP_2), (ii) Forward blocker door area (A_{14}), (iii) High-pressure compressor stator vane angle (STP_{27}), (iv) High-pressure booster hub stator vane angle (STP_{27d}), (v) Combustor fuel flow (w_{f36}), (vi) Aft variable bypass area (A_{16}), (vii) Afterburner fuel flow (w_{f6}), (viii) Nozzle throat area (A_8), (ix) Upper nozzle exit area ($A_{9,hi}$), and (x) Lower nozzle exit area ($A_{9,lo}$).

Given the appropriate inputs of throttle position, also known as power lever angle (PLA), and ambient conditions (e.g., altitude (h), Mach number (M), ambi-

ent temperature (T_a)), nonlinear dynamics of real-time turbofan engine operation are represented as a component level model in the simulation test bed. Both steady-state and transient operations of the gas turbine engine are simulated in the continuous-time setting. Overall performance maps are used to provide steady-state representations of the engine's rotating components. Fluid momentum in the bypass duct and the afterburner, mass and energy storage within control volumes, and rotor inertias are also included to provide capability for simulating transient operations. The components of the engine model consist of a single stage high-pressure ratio fan with variable inlet stator vanes, booster with independent hub and tip stator vanes, high-pressure mixed flow compressor, double-annular combustor, high-pressure and low-pressure turbines, afterburner, and nozzle components. The components of the engine model and station numbering are provided in Figure 5.3. The stations are numbered at the exit condition of each component starting from the flight conditions and inlet as the first station. The health of the engine is described by eleven quality parameters that include the efficiency scalar of the combustor (η_4) and the flow scalars and efficiency scalars of the fan (ζ_2 and η_2), the compressor (ζ_{27} and η_{27}), the booster (ζ_{27d} and η_{27d}), and the high pressure and low pressure turbines (ζ_{41}, η_{41}, ζ_{49}, and η_{49}). The open-loop engine model has three state variables, which are the low-pressure and the high-pressure rotor speeds, as well as the average metal (wall) temperature. Together with its ten actuators, each of which is modeled by a second order differential equation, total number of states associated with the augmented plant model is *twenty three*.

5.3 Description of the Simulation Test Bed of a Permanent Magnet Synchrnous Motor

This section describes the simulation test bed that is a representation of an inverter-driven permanent magnet synchronous motor ($PMSM$) [77], as depicted in Fig. 5.4. The simulation model of a generic $PMSM$, without a damper winding, is similar to that of a wound-rotor synchronous machine under the following simplifying assumptions:

- Negligible magnetic field saturation;

- Negligible eddy current loss and hysteresis loss;
- Negligible field current dynamics;
- Sinusoidal induced electromotive force (EMF);

In rotor reference frame, the governing equations of the stator voltage are given as:

$$v_q = Ri_q + \frac{d\lambda_q}{dt} + \omega_s \lambda_d \tag{5.2}$$

$$v_d = Ri_d + \frac{d\lambda_d}{dt} - \omega_s \lambda_q \tag{5.3}$$

where the subscripts q and d have their usual significance of quadrature and direct axes in the equivalent 2-phase representation; and

$$\lambda_q = L_q i_q \qquad \text{and} \qquad \lambda_d = L_d i_d + \lambda_{af} \tag{5.4}$$

with v, i, and L being the corresponding axis voltages, stator currents and inductances; R and ω_s are the stator resistance and inverter frequency, respectively, while λ_{af} is the flux linkage of the rotor magnets with the stator.

The generated electromagnetic torque is expressed as:

$$T_e = 1.5P\left[\lambda_{af} i_q + (L_d - L_q) i_d i_q\right] \tag{5.5}$$

and the equation of motor dynamics is given by:

$$T_e = T_L + B\omega_r + J\frac{d\omega_r}{dt} \tag{5.6}$$

where P is the number of pole pairs, T_L is the load torque, B is the damping coefficient, ω_r is the rotor speed, and J is the moment of inertia. The rotor speed $\omega_r = \omega_s/P$.

In state-space setting, the governing equations of the $PMSM$ take the following form:

$$\frac{di_q}{dt} = \left(v_q - Ri_q - \omega_s L_d i_d - \omega_s \lambda_{af}\right)/L_q \tag{5.7}$$

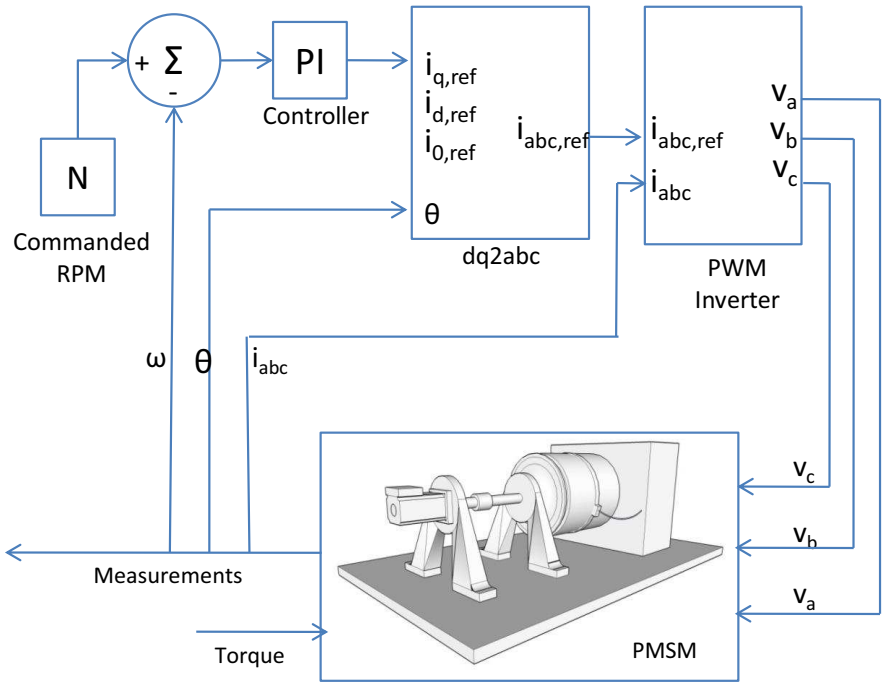


Figure 5.4. Inverter-driven permanent magnet synchronous motor ($PMSM$) system

$$\frac{di_d}{dt} = \left(v_d - Ri_d + \omega_s L_q i_q\right)/L_d \tag{5.8}$$

$$\frac{d\omega_r}{dt} = \left(T_e - T_L - B\omega_r\right)/J \tag{5.9}$$

In the control scheme shown in Fig. 5.4, i_d is forced to be zero. Consequently,

$$\lambda_d = \lambda_{af} \quad \text{and} \quad T_e = 1.5P\lambda_{af} i_q \tag{5.10}$$

In the above equation, the torque T_e is proportional to the quadrature axis current because the magnetic flux linkage λ_{af} is constant.

In the simulation test bed, the motor model is a three-phase four-pole device rated at 1.1 kW, 220 V, 3000 rpm and is fed by a pulse-width-modulated (PWM) inverter. The stator resistance of the motor is $R_s = 0.05\ \Omega$; the quadrature-axis and direct-axis inductances are: $L_q = L_d = 6.35 \times 10^{-4} H$; the nominal flux linkage $\lambda_{af} = 0.192\ Wb$; the rotor inertia $J = 0.011\, kg\ m^2$; and the friction factor is $B = 0.001889\ kgm^2\ s$.

A simple hysteresis current controller has been employed for controlling the power circuit that drives the $PMSM$, as seen in Fig. 5.4. Two control loops have been employed. The inner loop regulates the motor's stator currents, while the outer loop uses a proportional-integral controller to regulate the motor's speed. In this control scheme, the line currents i_a, i_b and i_c are measured. The reference values are compared with the actual values of the currents, and the error signal, thus constructed is used for generating the gate turn on/off commands. In the present scenario a hysteresis band of $0.25A$ on either side of the reference current i is employed.

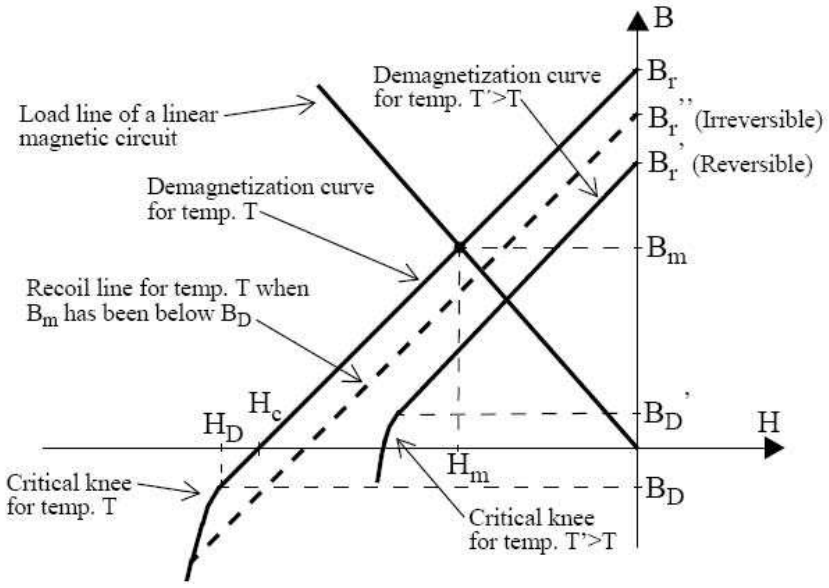


Figure 5.5. Demagnetization property of Neodymium-Iron-Boron (Nd-Fe-B) [1]

Chapter 6

Results

6.1 Superiority of Symbolic Dynamic Filtering over other methods

Fig. 6.2(a) compares the performance of SDF with Bayesian filter-based methods (i.e., particle filter (PF) and unscented filter (UKF)). These filters are calibrated to the nominal condition of $\beta = 0.1$, and the filter is designed to track both states (e.g., $y(t)$ and $\dot{y}(t)$), where 50 particles are used for the particle filter, as a tradeoff between tracking performance in the nominal conditions and CPU execution time and memory requirements. For unscented filtering, the parameter κ is set equal to 3 (see A), which is reported to be optimal for Gaussian priors [78]. For both PF and UKF, the variance of the zero-mean Gaussian process noise is set to 0.01 and the variance for zero-mean Gaussian measurement noise is 0.05. The Monte Carlo Markov Chain ($MCMC$) analysis has been carried out on $10,000$ data points, sampled at a rate of $T_s = 0.01$.

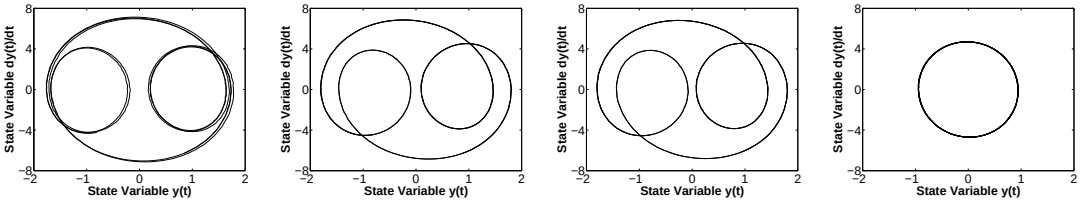
Figure 6.2(b) compares SDF for detection of anomaly patterns to MLPNN and RBFNN neural networks as well as Principal Component Analysis (PCA). and Bayesian Filter-based methods (PF and UKF). The Multilayer Percepron Neural Network ($MLPNN$) consists of three hidden layers with 50 neurons in each one of them and an output layer with one neuron (as the number of output is one). Tangent sigmoid functions have been used in the hidden layers as transfer functions, while the output layer uses a linear function. On the other hand, the Radial Basis

Function Neural Network ($RBFNN$) uses only one hidden layer and one output layer (with one neuron) as described earlier. Optimal training was obtained using 100 neurons in the hidden layer. The hidden layer uses radial basis function, whereas the output layer uses linear function as transfer functions. For training of the network, two thousand data points are chosen from the input-output time-series data set of the nominal system, i.e., with $\beta = 0.1$ at steady state.

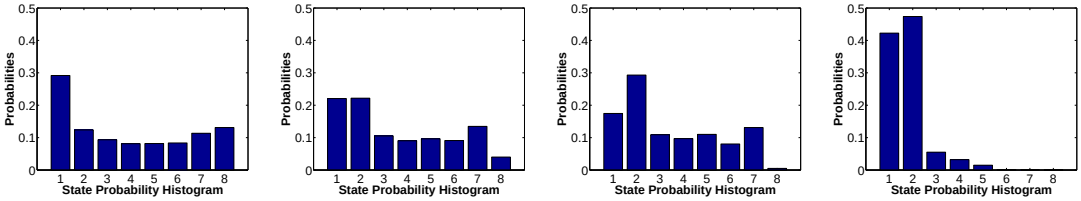
The same input and target vectors are used for training of $MLPNN$ and $RBFNN$. In both cases, the error goals are chosen so that the network could follow the target with reasonable accuracy. In the estimation step of the networks, four thousand data points have been chosen from the steady-state input-output time series data of the system. The error sequence is generated by taking point by point difference between the system output and the output generated from the neural networks. The anomaly measure μ is calculated as described in Section 2.3.

Figures 6.2(a) and Fig. 6.2(a) exhibit a family of normalized profiles of anomaly measure μ versus the dissipation parameter β, where each profile show gradual increase in μ until the bifurcation at $\beta \approx 0.33$. Changes in the value of μ, its slope (i.e., $\frac{\partial \mu}{\partial \beta}$), and its curvature (i.e., $\frac{\partial^2 \mu}{\partial \beta^2}$) provide early warnings for a forthcoming major change in the system dynamics. From this perspective, the performance of SDF is superior to that of Bayesian filtering, both types of Neural networks, and other statistical methods (i.e., PCA and KRA). It is also noted that the profile of SDF is smoother than those of PF and UKF. The smoothness of SDF reduces false alarms particularly for small changes in β from the nominal condition. Similarly, SDF outperforms $RBFNN$, $MLPNN$, PCA, and KRA.

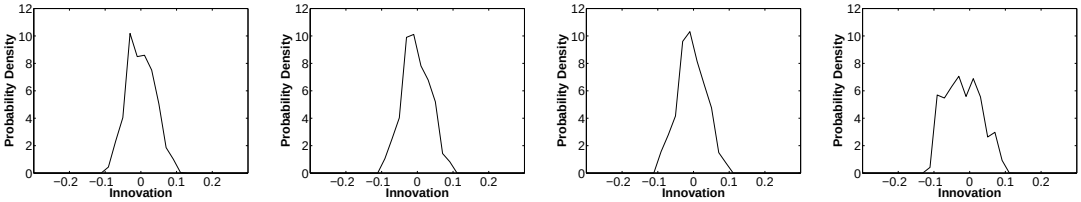
Table 6.1 provides a comparison of execution time and memory requirement of the afore-mentioned seven methods for computation of the anomaly measure μ. In each case, the CPU time for a single operation cycle at a time epoch, listed in Table 6.1, is obtained from the average of execution time for operation cycles at 16 consecutive slow time epochs on a 3.40 GHz Pentium 4 processor in the Matlab 7.0.1 environment. As seen in Table I, the execution time varies from a fraction of millisecond for KRA to hundreds of seconds for PF. Execution time for Neural Network-based methods and SDF are comparable although RBFNN is faster than MLPNN and SDF. However, Bayesian filters UKF and PF are one and two orders of magnitude slower than SDF, respectively. The requirement of (random access)



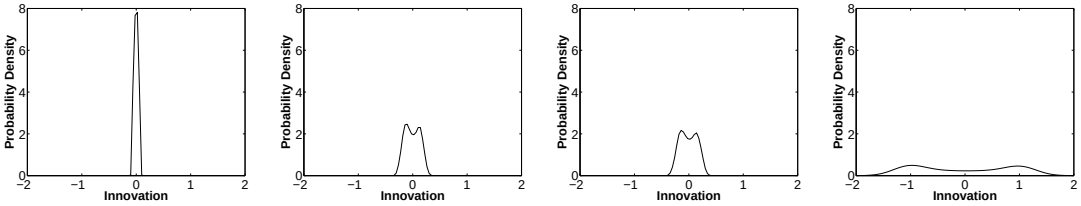
(a) Phase plot for $\beta = 0.10$ (b) Phase plot for $\beta = 0.30$ (c) Phase plot for $\beta = 0.32$ (d) Phase plot for $\beta = 0.34$



(e) SDF Histogram for $\beta = 0.10$ (f) SDF Histogram for $\beta = 0.30$ (g) SDF Histogram for $\beta = 0.32$ (h) SDF Histogram for $\beta = 0.34$



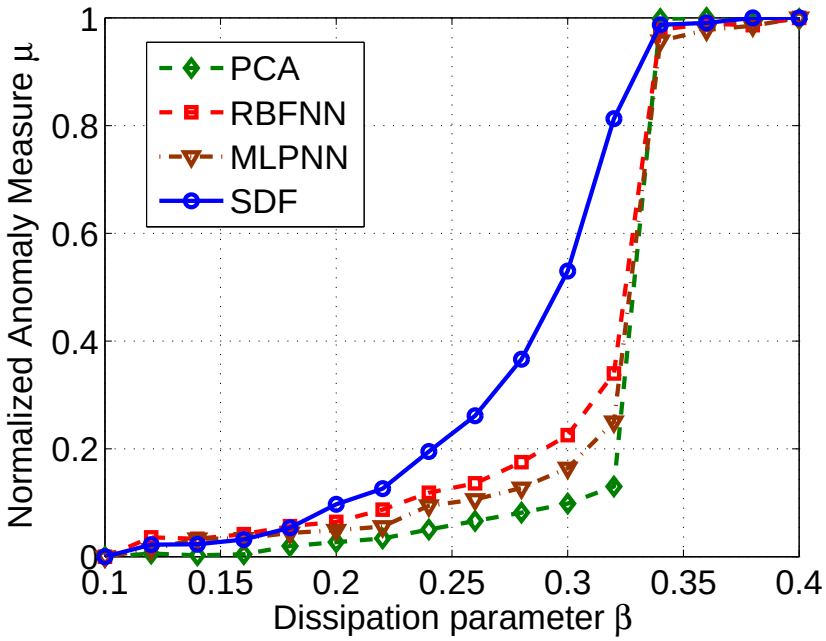
(i) RBFNN Density for $\beta = 0.10$ (j) RBFNN Density for $\beta = 0.30$ (k) RBFNN Density for $\beta = 0.32$ (l) RBFNN Density for $\beta = 0.34$

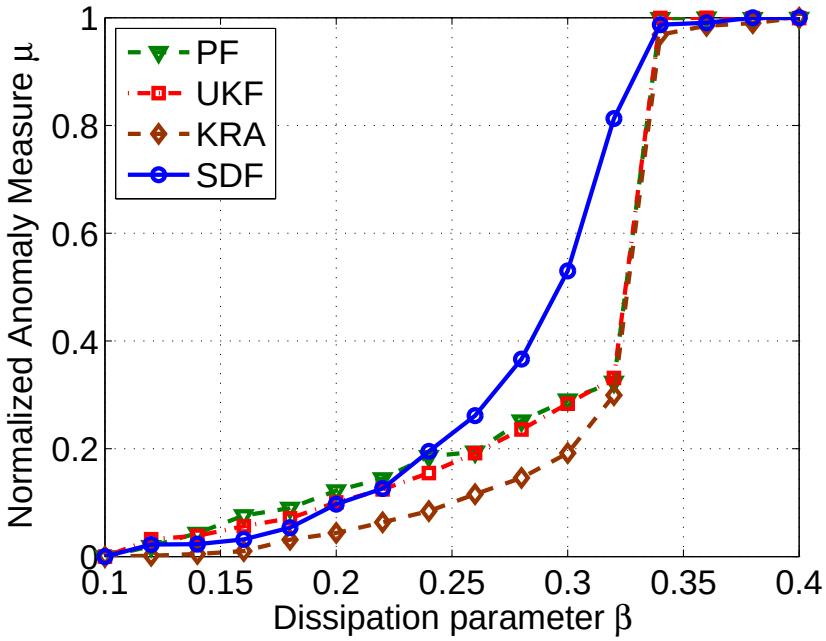


(m) PF Density for $\beta = 0.10$ (n) PF Density for $\beta = 0.30$ (o) PF Density for $\beta = 0.32$ (p) PF Density for $\beta = 0.34$

Figure 6.1. Evolution of anomaly patterns for changes in system dynamics

memory in each case is more or less similar (less than 5MB), which is insignificant for a commercially available laptop computer. However, for RBFNN and MLPNN, the training phase requires 45MB to 60 MB of memory, which is also reasonable.





(b) Comparison of SDF, Bayesian filtering, and KRA

Figure 6.2. Evaluation of Gradually Evolving Anomaly Patterns

Table 6.1. Comparison of execution time

Anomaly detection method	Execution time	Memory requirement
KRA	2.23×10^{-3} sec	2.95 MB
PCA	4.30×10^{-2} sec	2.88 MB
RBFNN	8.09×10^{-1} sec	4.05 MB
MLPNN	4.60×10^{0} sec	4.15 MB
SDF	4.65×10^{0} sec	2.94 MB
UKF	5.10×10^{1} sec	4.19 MB
PF	2.74×10^{2} sec	4.69 MB

6.2 Results for Statistical Estimation of multiple parameters

This section presents the test results of multiple-parameter estimation on two electronic circuits, namely the externally excited Duffing system [73] and the unforced van der Pol system [79], on the test apparatus described in a previous publication [67].

6.2.1 Results on Duffing system

This subsection analyzes and presents the experimental results for multiple parameter estimation in the Duffing system described by Eq. (5.1). For the forward problem/training (see Subsection 3.5), training data sets were generated with α_1 ranging from 0.10 to 1.50 in steps of 0.05, and β ranging from 0.10 to 0.40 in steps of 0.02, and; the nominal condition was chosen as $\alpha_1 = 1.0$ and $\beta = 0.1$; and an *SDF* was constructed with the number of states in the automaton $|\Sigma| = 8$. This information on time series data was then fed into the *SDF* to compute the components p_j of pattern vectors $\mathbf{p}$ at different values of the parameter pair $(\alpha_1,\ \beta)$. As the dynamics of the Duffing system changed due to variations in the parameters α_1 and β, the statistics of the symbol sequences were altered and so were the pattern vectors.

For the inverse problem/testing (see Subsection 3.5.2), experiments were conducted at the assigned values of the parameters that were different from those in the forward problem of *SDF* but within the range of α_1 and β where the training

was conducted. The components p_j of pattern vectors $\mathbf{p}$ at different values of the parameter pair $(\alpha_1,\ \beta)$ were computed from the data sets that were generated with these assigned values of parameters. For a typical test at $\alpha_1 = 0.75$ and $\beta = 0.23$, the 3-dimensional plot in Fig. 6.3 shows the bivariate probability distribution, followed by a close-up view of the contour plots in Fig. 6.4. The parameter pair $(\alpha_1,\ \beta)$ is crisply identified by a single, sharp spike in the probability distribution plot of Fig. 6.3, where the estimates $\hat{\alpha}_1$ and $\hat{\beta}$ lie in the ranges of $(0.745,\ 0.755)$ and $(0.235,\ 0.240)$, respectively, as seen in Fig. 6.4. Table 6.2 shows the results for mean, standard deviation, and confidence intervals of the parameter estimates, $\hat{\alpha_1}$ and $\hat{\beta}$ for test runs with four different pairs of α_1^{test} and β^{test} that do not belong to the set $\mathcal{S}$ of training data. It is seen that the estimated mean values of both α_1 and β are orders of magnitude greater than their respective standard deviations $\hat{\sigma}_{\alpha_1} \triangleq \sqrt{\hat{C}_{\alpha_1\alpha_1}}$ and $\hat{\sigma}_{\beta} \triangleq \sqrt{\hat{C}_{\beta\beta}}$. This observation suggests that the estimates are relatively close to the true values of the parameters. It is also seen in Table 6.2 that the correlation coefficient $\frac{\hat{C}_{\alpha_1\beta}}{\hat{\sigma}_{\alpha_1}\hat{\sigma}_{\beta}}$ is a positive fraction, which implies that the parameters α_1 and β are positively correlated. The rationale for this correlation is that variations even in a single component of a dynamical system may cause simultaneous variations in several parameters of its governing equations. In the Duffing system, usage of different but identically manufactured electronic cards caused simultaneous variations of both parameters α_1 and β in Eq. (5.1).

Table 6.2. Predicted values of $\left(\hat{\alpha_1}, \hat{\beta}\right)$ for the Duffing Equation

Test Num.	Estimates						
	Parameter α_1				Parameter β		
	α_1^{test}	$\hat{\alpha}_1$	$\hat{\sigma}_{\alpha_1}$	$\hat{C}_{\alpha_1\beta}$	β^{test}	$\hat{\beta}$	$\hat{\sigma}_{\beta}$
1	0.30	0.30	$8.4e-4$	$2.67e^{-7}$	0.10	0.10	$4.0e-4$
2	0.45	0.46	0.015	$5.15e^{-4}$	0.20	0.20	0.057
3	0.15	0.15	$3.3e-3$	$2.465e^{-5}$	0.14	0.14	$8.0e-3$
4	0.65	0.65	$3.0e-3$	$8.16e^{-6}$	0.35	0.36	$7.0e-3$

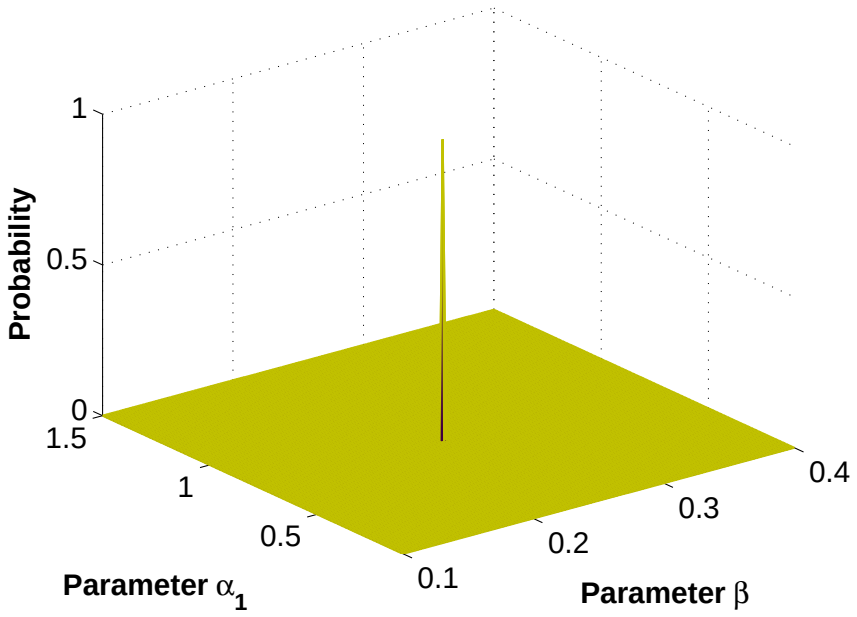


Figure 6.3. Joint probability distribution of the parameter pair α_1 and β

Table 6.3. Confidence intervals for the Duffing Equation

Test num	Estimates			
	95 % Confidence Interval		90 % Confidence Interval	
	$(\alpha_{1_{\min}}, \alpha_{1_{\max}})$	$(\beta_{\min}, \beta_{\max})$	$(\alpha_{1_{\min}}, \alpha_{1_{\max}})$	$(\beta_{\min}, \beta_{\max})$
1	$(0.30, 0.30)$	$(0.30, 0.30)$	$(0.10, 0.10)$	$(0.10, 0.10)$
2	$(0.45, 0.46)$	$(0.20, 0.20)$	$(0.45, 0.46)$	$(0.20, 0.20)$
3	$(0.15, 0.15)$	$(0.14, 0.14)$	$(0.15, 0.15)$	$(0.14, 0.14)$
4	$(0.65, 0.65)$	$(0.36, 0.36)$	$(0.65, 0.65)$	$(0.36, 0.36)$

6.3 Results on van der Pol System

This subsection analyzes and presents experimental results for multiple-parameter estimation in the van der Pol system described by Eq. (B.4). For the forward problem/training (see Subsection 3.5), training data sets were generated with both parameters μ and ω ranging from from 0.5 to 4.0 in increments of 0.5; and an *SDF* was constructed with the number of states in the automaton $|\Sigma| = 8$. This information on time series data was then fed into the *SDF* to compute pattern

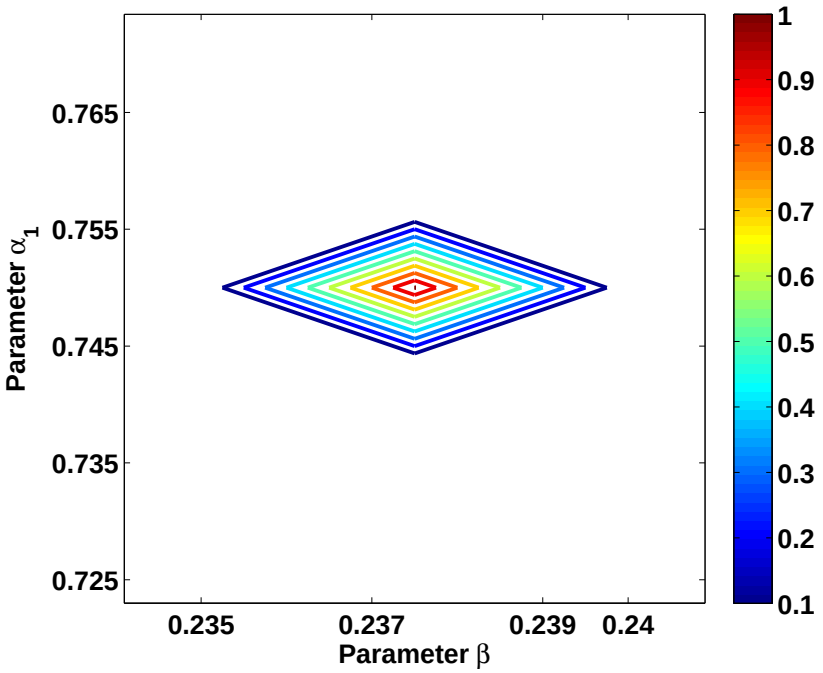


Figure 6.4. Zoomed-in contour plots of the parameter pair α_1 and β

vectors $\mathbf{p}$ and deviation measures $\mathcal{M}$ at different values of the parameter pair $(\mu,\ \omega)$. As the dynamics of the van der Pol system changed due to variations in the parameters μ and ω, the statistics of the symbol sequences were altered and so were the pattern vectors. However, unlike the Duffing system, no abrupt change (e.g., bifurcation) in the dynamic behavior was observed as μ and ω were varied from the nominal condition.

For the inverse problem/testing (see Subsection 3.5.2), experiments were conducted at the assigned values of the parameters that were different from those in the forward problem of *SDF*, i.e., they do not belong to the set $\mathcal{S}$ of training data. Pattern vectors $\mathbf{p}$ and the associated deviation measures $\mathcal{M}$ were estimated from of the data sets generated with these assigned values of parameters. Table 6.4 shows the results for mean, standard deviation, and confidence intervals of the parameter estimates, $\hat{\mu}$ and $\hat{\omega}$ for test runs with four different pairs of μ^{test} and ω^{test}. It is seen that the estimated mean values of both μ and ω are orders of magnitude greater than their respective standard deviations $\hat{\sigma}_\mu \triangleq \sqrt{\hat{C}_{\mu\mu}}$ and $\hat{\sigma}_\omega \triangleq \sqrt{\hat{C}_{\omega\omega}}$. This observation suggests that the estimates are relatively close to the true values

of the parameters. It is also seen in Table 6.4 that the correlation coefficient $\frac{\hat{C}_{\mu\omega}}{\hat{\sigma}_\mu \hat{\sigma}_\omega}$ is close to 0, implying that the parameters μ and ω are very weakly correlated.

Table 6.4. Predicted values of $(\hat{\mu}, \hat{\omega})$ for the Van der Pol Equation

Test Num.	Estimates						
	Parameter μ				Parameter ω		
	μ^{test}	$\hat{\mu}$	$\hat{\sigma}_\mu$	$\hat{C}_{\mu\omega}$	ω^{test}	$\hat{\omega}$	$\hat{\sigma}_\omega$
1	2.50	2.50	0.006	$-6.7e^{-6}$	1.00	1.00	0.028
2	3.30	3.32	0.059	$3.0e^{-4}$	3.40	3.42	0.035
3	4.00	3.99	0.069	$1.1e^{-5}$	2.50	2.49	0.087
4	3.50	3.51	0.016	$2.5e^{-4}$	4.00	3.99	0.220

Table 6.5. Confidence intervals for the Van der Pol Equation

Test num	Estimates			
	95 % Confidence Interval		90 % Confidence Interval	
	$(\mu_{\min}, \mu_{\max})$	$(\omega_{\min}, \omega_{\max})$	$(\mu_{\min}, \mu_{\max})$	$(\omega_{\min}, \omega_{\max})$
1	$(2.49, 2.51)$	$(0.99, 1.01)$	$(2.49, 2.50)$	$(0.99, 1.00)$
2	$(3.29, 3.35)$	$(3.39, 3.43)$	$(3.30, 3.33)$	$(3.40, 3.42)$
3	$(3.98, 4.02)$	$(2.46, 2.52)$	$(3.99, 4.00)$	$(2.48, 2.50)$
4	$(3.48, 3.52)$	$(3.75, 4.12)$	$(3.49, 3.51)$	$(3.89, 4.08)$

6.4 Estimation of multiple parameters for the PMSM

6.4.1 Failure Modes

Failure due to demagnetization of the permanent magnet in both surface-mounted and buried-magnet $PMSM$s have been widely studied in literature [1]. Demagnetization can occur due to several reasons, notable among which are demagnetization due to a strong opposing magnetic field, and also due to high temperature.

A strong opposing magnetic flux can be created in the event of a short circuit between one terminal of the machine and the (normally) isolated neutral point of the machine, short circuit between two or three terminals of the machine and short circuit in one of the diodes or electronic valves of the inverter, giving rise to a direct current (DC) in the machine even in short circuit steady state.

Risk of irreversible demagnetization is present if the counter-acting flux lowers the flux density in the magnet to a point (H_D, B_D) that is just above the so-called critical knee of the magnet's BH-curve, which has been illustrated in Fig. 5.5. The common method to check the demagnetization of the permanent magnets due to armature reaction is described in [80]. The disadvantage of this method is the assumption that the permanent magnet pole has uniform saturation. A more accurate way to check the demagnetization is with the finite element method.

Partial or complete demagnetization can also result from high temperature of the magnets and the winding insulation. The temperature increases the resistance of the winding wires and the increased resistance affects the applied current to the motor. At higher temperatures ($\sim 100^0C$), an appreciable deterioration in acceleration might be noticed, as the torque generated by a reduced magnetic flux drops below its nominal value.

Table 6.6. Predicted values of $\left(\hat{\lambda}_{af}, \hat{B}\right)$ for the $PMSM$

Test Num.	Estimates					
	Parameter λ_{af}			Parameter B		
	$\lambda_{af}^{\text{test}}$	$\hat{\lambda}_{af}$	$\hat{\sigma}_{\lambda_{af}}$	B^{test}	$\hat{B}$	$\hat{\sigma}_B$
1	0.075	0.075	$1.6e-7$	0.60	0.60	$1.6e-8$
2	0.09	0.09	$6.2e-7$	0.80	0.80	1.2e-8
3	0.11	0.11	$1.5e-9$	0.70	0.70	$2.3e-8$
4	0.10	0.10	$3.8e-9$	0.90	0.90	$7.0e-9$

Fatigue failure of bearings is quite common even under normal operating conditions with balanced load. Factors which affect smooth operation of the bearing are normal internal operating stresses caused by vibration, inherent eccentricity,

Table 6.7. Predicted values of $\left(\hat{\lambda}_{af}, \hat{B}\right)$ and confidence intervals for the $PMSM$

Test No.	Estimates			
	95 % Confidence Interval		90 % Confidence Interval	
	$\left(\lambda_{af_{\min}}, \lambda_{af_{\max}}\right)$	$(B_{\min}, B_{\max})$	$\left(\lambda_{af_{\min}}, \lambda_{af_{\max}}\right)$	$(B_{\min}, B_{\max})$
1	$(0.60, 0.60)$	$(0.075, 0.075)$	$(0.60, 0.60)$	$(0.075, 0.075)$
2	$(0.09, 0.09)$	$(0.80, 0.80)$	$(0.09, 0.09)$	$(0.80, 0.80)$
3	$(0.11, 0.11)$	$(0.70, 0.70)$	$(0.11, 0.11)$	$(0.70, 0.70)$
4	$(0.10, 0.10)$	$(0.90, 0.90)$	$(0.10, 0.10)$	$(0.90, 0.90)$

and bearing currents due to solid state drives [81], as well as external causes, such as abnormal mounting. Ball bearing related defects manifest themselves as outer bearing race defect, inner bearing race defect, ball defect, and train defect. Specific information concerning the bearing construction is indispensable for predicting the exact failure characteristics. However it may be safely assumed, that all such performance deteriorations principally manifest themselves as increase in the bearing friction.

6.4.2 Results on simulation model

In this dissertation, health-monitoring of permanent magnet synchronous motors has been proposed by a nonlinear time series analysis technique called symbolic dynamic filtering in conjunction with Hilbert Transform Space Partitioning [64]. The motor described in Section 5.3 is assumed to undergo a steady deterioration in terms of permanent magnet flux linkage, due to either of the two reasons mentioned above. The flux linkage λ_{af} drops from its nominal value of $0.192Wb$ to $0.007Wb$. At the same time the friction in the bearing increases from the nominal value of $B = 0.5Nms$ to a value of $B = 1Nms$. The line current signals i_a, i_b, i_c, which in principle reflect both these deteriorations are collected from the motor output at each of these off-nominal partially-demagnetized state of the rotor running on bearings with gradually increasing friction.

This data is then Hilbert Transformed and converted into discrete symbols.

Maximum Entropy partitioning is employed in the radial direction while the data is uniformly partitioned in the angular direction. The discretization procedure has been briefly discussed in Section A and follows the procedure described in Section 2.4. Here an alphabet size of $|\Sigma| = 15$ and a depth of $D = 1$ has been employed. This information on time series data was then fed into the *SDF* to compute the components p_j of pattern vectors $\mathbf{p}$ at different values of the parameter pair λ_{af}, B. The pattern vector obtained by constructing the D-Markov machine representation of the motor characterizes the health condition of the motor in general. As the dynamics of the PMSM system changed due to deterioration in flux linkage, as well as friction coefficient, the statistics of the symbol sequences were altered and so were the pattern vectors.

The angle-measure described in [2] is used to obtain the departure of the system from its nominal operating condition. The resulting measure has been plotted in Fig. 6.5 against the slowly deteriorating permanent magnet flux linkage. The concave nature of the anomaly measure curve at incipient fault conditions is highly desirable to suppress excessive false warnings, while the convex nature in the later part, when the system is near-critical condition guarantees high sensitivity. the flattening out of the curve after the flux linkage drops below the threshold indicates that hereafter the motor is almost inoperable.

For the inverse problem, experiments were conducted at several random predetermined values of the parameters that were different from those in the forward problem/training of *SDF*. The components p_j of pattern vectors $\mathbf{p}$ at different values of the parameter pair were computed from the data sets that were generated with these assigned values of parameters. The parameter pair is crisply identified by a single, sharp spike in the probability distribution plot. Table 6.2 shows the results for mean and confidence intervals of the parameter estimates for four different test runs that did not belong to the set of training data. It is seen that the estimated mean values of the flux linkage (λ_{af}) and friction coefficient (B) are very close to their true values and are orders of magnitude greater than the respective standard deviations $\hat{\sigma}_{\lambda_{af}}$ and $\hat{\sigma}_B$.

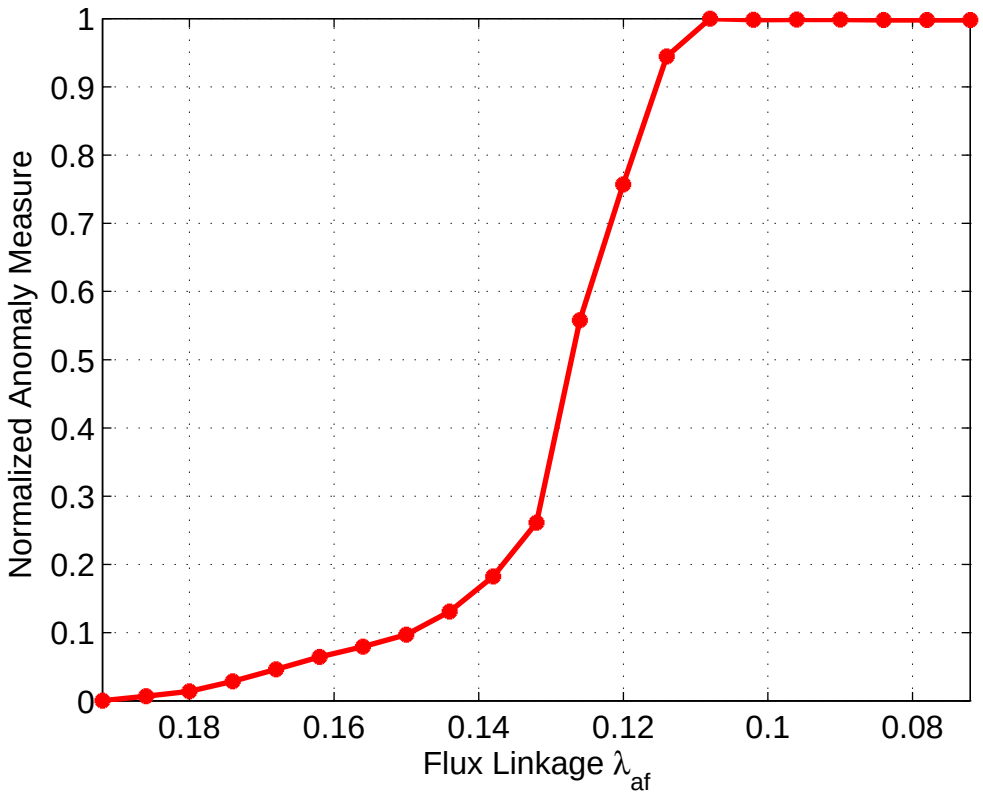


Figure 6.5. Anomaly measure in a permanent magnet synchronous motor

6.5 Results and discussion on using Sensor Fusion for the estimation of multiple parameters

The above formulation of the inverse problem uses information from all the sensors y_j for $j = 1, 2, ..., |\mathcal{Y}|$, i.e., the information from all sensors are fused together to estimate the fault level in the engine test case. However, this fusion technique allows the user to choose the number and the combination of sensors to be used for the multiple fault estimation. For example, let us assume that the user just wants to use only one sensor, say, Sensor y_1. Then, only blocks pertaining to y_1 are to be selected from the forward problem pattern database. Thus, elements $m_i^j(\boldsymbol{s}^k)$ for $j = 1$ and $\forall i \in \{1, 2, ..., N_1 - 1\}$ are selected from $\mu(\boldsymbol{s}^k)$ and elements $\gamma_{xy}^{mn}(\boldsymbol{s}^k)$ for $m, n = 1$ and $\forall x, y \in \{1, 2, ..., N_1 - 1\}$ are selected from $\Gamma(\boldsymbol{s}^k)$ $\forall \boldsymbol{s}^k \in \mathcal{S}$. Note,

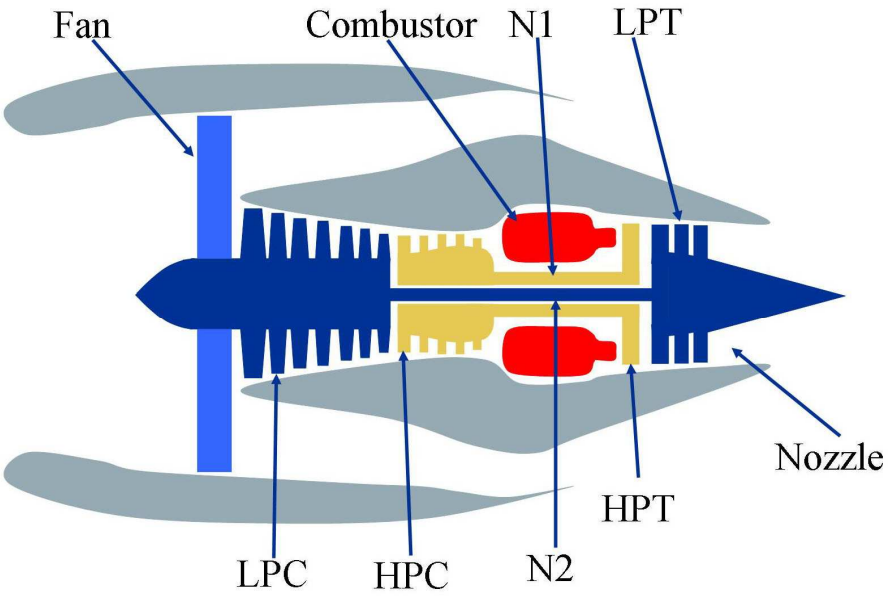
that the selected block of the covariance matrices signify the correlation among symbols for data from s_1. However, to use more than one sensor information, correlation among sensors also have to be considered along with correlation among symbols, which is explained in the following example. In another example, if both sensors s_1 and s_3 are to be used. Then, elements $m_i^j(\boldsymbol{s}^k)$, $\forall j \in \{1,3\}$ and $\forall i \in \{1,2,...,N_j-1\}$ are selected from $\mu(\boldsymbol{s}^k)$ and elements $\gamma_{xy}^{mn}(\boldsymbol{s}^k)$, $\forall m,n \in \{1,3\}$ and $\forall x \in \{1,2,...,N_m-1\}$, $\forall y \in \{1,2,...,N_n-1\}$ are selected from $\Gamma(\boldsymbol{s}^k)$ $\forall \boldsymbol{s}^k \in \mathcal{S}$. It follows from the above two examples that the elements of the test patterns need to be selected corresponding to the sensors under consideration.

Remark 6.5.1. *The current framework attempts to fuse information from different sensors at feature level as opposed to the frameworks of data level or decision level fusion. The advantages of the present sensor information fusion framework are delineated below.*

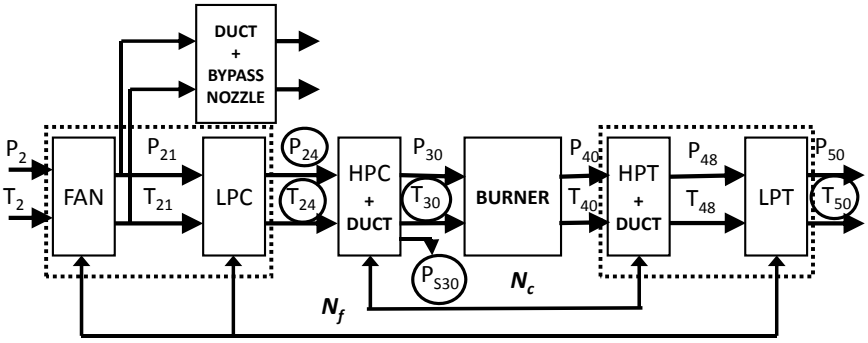
- *Data level fusion techniques often encounter scaling problem while fusing information from sensors of different modality. However, the present technique fuses the probability vector patterns, which does not have any scaling issue.*
- *Decision level fusion generally provides too coarse diagnosis of faults and also requires in depth understanding of the physical system.*
- *Multi-dimensionality of the parameter space has been taken care of without the use of product automata which leads to state explosion.*

6.6 Validation on the C-MAPSS test-bed

The C-MAPSS simulation test-bed, developed at NASA, is built upon the model of a commercial-scale two-spool turbofan engine and its control system. The engine under consideration produces a thrust of approximately 400,000 N and is designed for operation at (i) altitudes from sea level up to 12,200 m, (ii) Mach numbers from 0 to 0.90, and (iii) sea-level temperatures from approximately -50^oC to 50^oC. The throttle resolving angle (TRA) can be set to any value in the range between 0° (minimum power) and 100° (maximum power).



(a) Gas turbine engine schematic



(b) Gas turbine engine model configuration

Figure 6.6. C-MAPSS engine simulation test-bed

As seen in Figures 6.6(a) and 6.6(b), the simulation test-bed of the gas turbine engine system consists of high pressure compressor (HPC), combustor, and high pressure turbine (HPT), which form the core of the engine model; this subsystem is also referred to as the gas generator. In the turbofan engine, the engine core is surrounded by the fan and Low pressure compressure (LPC) in the front and an additional low pressure turbine (LPT) at the rear; and fan, LPC and LPT are mechanically connected by an additional shaft. The fan shaft passes through the core shaft and, due to this type of arrangement, the engine is called a two spool engine. In contrast to gas turbine engines for military aircraft [11], a relatively small part of the incoming air at the engine inlet passes through the fan and continues on

into the core compressor and then into the combustor, where it is mixed with fuel and combustion occurs; therefore, this type of engine is known as a high-bypass engine. The hot exhaust gas, called the core airflow, passes through the core and LPT and then exits through the nozzle; and the rest of the incoming air passes through the fan and bypasses, or flows around the engine. A gain-scheduled control system is incorporated in the engine system, which consists of (i) a fan-speed controller for a specified throttle-resolver angle (TRA), (ii) three high-limit regulators that prevent the engine from exceeding its design limits for core-spool speed, engine-pressure ratio, and HPT exit temperature, (iii) the fourth limit regulator that attempts to prevent the static pressure at the HPC exit from dropping too low, (iv) acceleration and deceleration limiters for the core-spool speed, and (v) a comprehensive logic structure that integrates these control-system components in a manner similar to that used in real engine controllers such that integrator-windup problems are avoided. To achieve fast execution of simulation runs, the sensors and actuators are approximated to have instantaneous response, no computational time delays, and no drift and or bias. Given the inputs of TRA, altitude (a) and Mach number (M), the interactively controlled component models at the simulation test-bed compute nonlinear dynamics of real-time turbofan engine operation. Both steady-state and transient operations are simulated in the continuous-time setting. Performance maps are used to provide steady-state representations of the engine's rotating components. Fluid momentum in the bypass duct and the augmentor, mass and energy storage within control volumes, and rotor inertias are also included to model transient operations. The entire test-bed code is written on Matlab and Simulink platform.

As indicated earlier, this dissertation addresses estimation of those faults that cause efficiency degradation in engine components. In the current configuration of the C-MAPSS simulation test-bed, there are 13 health parameter inputs, namely, efficiency health parameters (ψ), flow health parameters (ζ) and pressure ratio modifiers, that simulates the effects of faults and/or degradation in the engine components. Ten, out of these 13 health parameters, are selected to modify efficiency (η) and flow (ϕ) which are defined [82] as

- $\eta \triangleq$ the ratio of actual enthalpy change and ideal enthalpy change

- $\phi \triangleq$ the ratio of tip rotor velocity and axial fluid flow velocity

For the engine's five rotating components (i.e., Fan, LPC, HPC, HPT and LPT), the ten health parameters are: (a) fan (ψ_F, ζ_F), (b) low pressure compressure (ψ_{LPC}, ζ_{LPC}), (c) high pressure compressor (ψ_{HPC}, ζ_{HPC}), (d) high pressure turbine (ψ_{HPT}, ζ_{HPT}), and (e) low pressure turbine (ψ_{LPT}, ζ_{LPT}). Table 6.8 lists the (commercially available) sensors and their locations (see Fig. 6.6(b)) that have been used for multiple fault estimation in C-MAPSS engine test-bed.

Table 6.8. Required Engine System Sensors

Sensors	Description
P_{24}	LPC exit/ HPC inlet pressure
T_{24}	LPC exit/ HPC inlet temperature
Ps_{30}	HPC exit static pressure
T_{30}	HPC exit/ Burner inlet temperature
T_{50}	LPT exit temperature

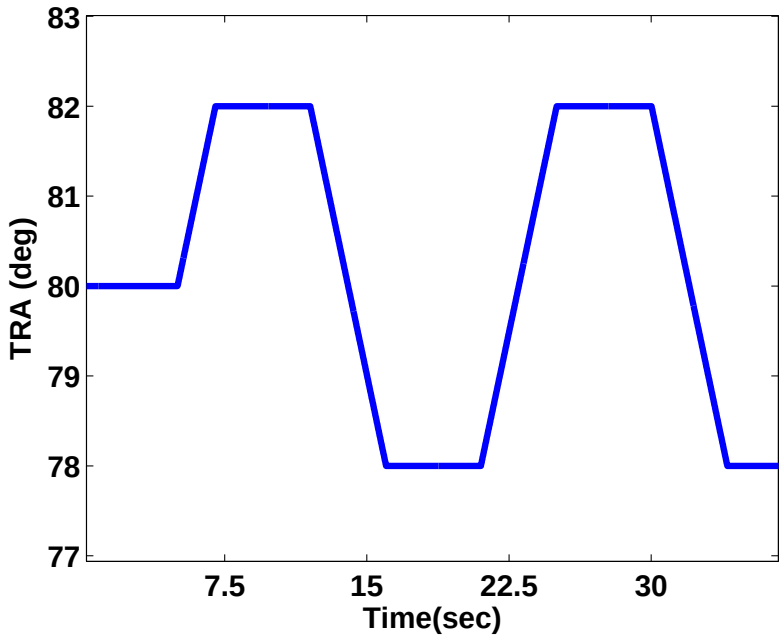


Figure 6.7. Throttle resolving angle (TRA) profile

6.6.1 Discussion

Time series data have been collected for different sensors under persistent excitation of TRA inputs that have truncated triangular profiles with the mean value of 80°, fluctuations within ±2° and frequency of 0.056 Hz as shown in Fig. 6.7. The ambient conditions are chosen to be at the sea level (i.e. altitude $a = 0.0$, Mach number $M = 0.0$) when the engine is on the ground for fault monitoring and maintenance by the engineering personnel. The engine simulation is conducted at a frequency of 66.67 Hz (i.e., inter-sample time of $15ms$) and the length of the simulation time window is 150 seconds, which generate $10,000$ data points for each training or test case.

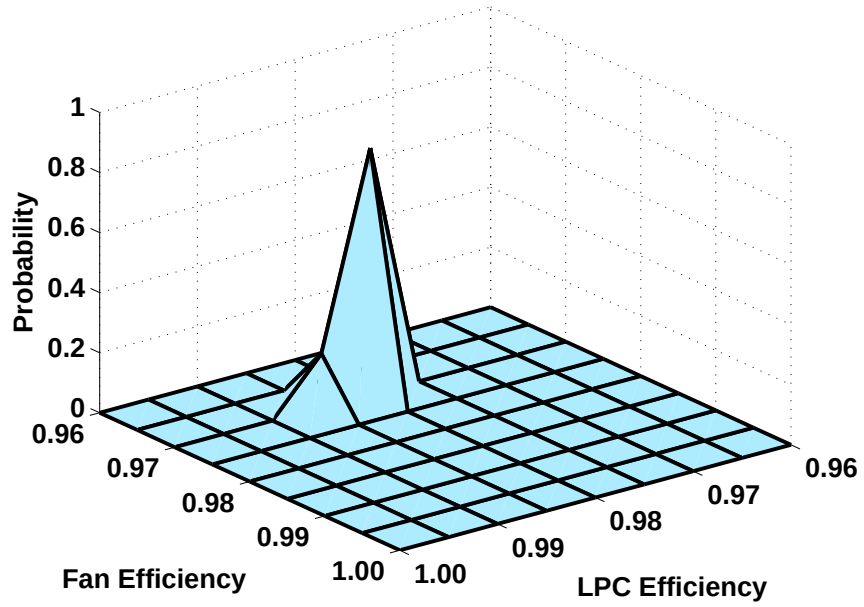
An engine component C is considered in nominal condition when both ψ_C and ζ_C are equal to 1. Fault is injected in the component C by simultaneously reducing both ψ_C and ζ_C by same amount in the results reported here. Although the algorithm described above, does not have any restriction on the dimension of the parameter space, the result presented here considers simultaneous degradation of two different components. Subsection 6.6.1.1 describes a fault condition, where Fan and LPC are degraded simultaneously, whereas Subsection 6.6.1.2 analyzes simultaneous degradation in HPT and LPT . For both training (i.e., forward problem) and testing (i.e., inverse problem), time series data from all sensors, listed in Table 6.8, are generated with ψ and ζ ranging from 1.0 to 0.96 (i.e., 4% relative loss in efficiency) in steps of 0.005 for the engine components under consideration. For SDF analysis, the number of states in the PFSA is selected to be 15 for each sensor after pre-processing the time series data by Hilbert transform and pattern vectors are generated for each of the possible fault conditions. For example, in the first case, patterns are generated for all possible combinations of the Fan and LPC efficiency values with points in the square grid made by efficiency parameter values ranging from 1.0 to 0.96 in steps of 0.005. Fifty repetitions of each simulation have been conducted to generate pattern vector statistics with injected process and sensor noise. For testing (i.e., inverse problem), fault conditions are chosen within the range of training data such that they do not coincide with the training grid points.. The results of multiple-fault estimation are presented in the following two subsections along with discussions on sensor fusion.

6.6.1.1 Fault estimation in Fan and LPC

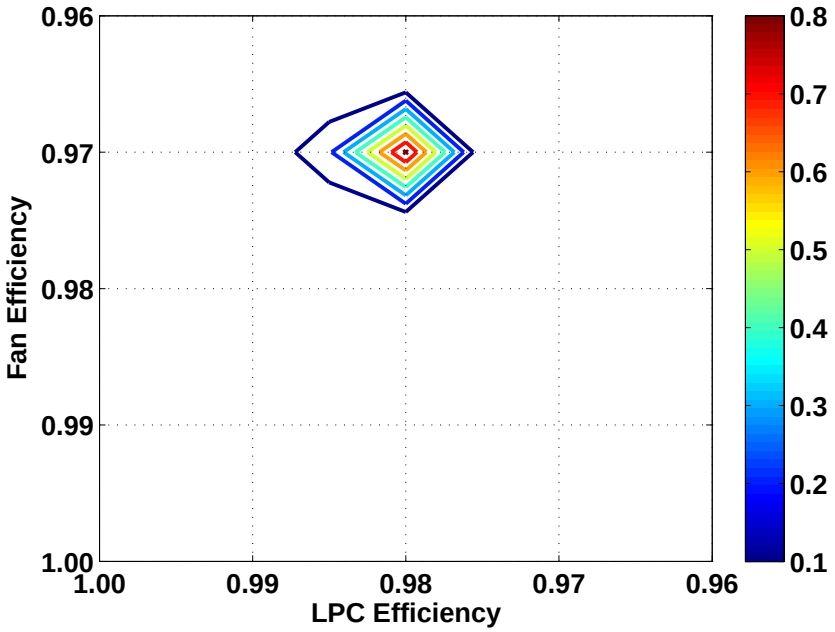
A test pattern is generated for a given fault condition, $\psi_F = \zeta_F = 0.973$ and $\psi_{LPC} = \zeta_{LPC} = 0.981$. The 3-dimensional plot in Fig. 6.8(a) shows the bivariate probability distribution of the estimated fault condition, followed by a close-up view of the contour plots in Fig. 6.8(b), where the results are generated from time series of a single sensor, namely, Ps_{30}. The estimates lie within the $\pm 3\sigma$ bound around the estimated mean (see Eq. (4.9)), where the variance σ^2 is obtained as a diagonal element of the estimated covariance matrix $\hat{C}_{\underline{s}}$ (see Eq. (4.10)). In this case, the estimates range from 0.9606 to 0.9704 for ψ_F and ζ_F, and from 0.9805 to 0.9813 for ψ_{LPC} and ζ_{LPC}, respectively. This indicates that the correct region is located in the parameter space, which assigns highest probability to the nearest training grid point.

6.6.1.2 Fault estimation in HPT-LPT

This example shows the result for a fault condition, $\psi_{HPT} = \zeta_{HPT} = 0.977$ and $\psi_{LPT} = \zeta_{LPT} = 0.985$. In contrast to the previous example of fan and LPC, the plots in Fig. 6.9(a) and Fig. 6.9(b) show that there is an ambiguity in estimation when using information from only one sensor, namely Ps_{30}. Although, it identifies the correct region with significant probability, another fault condition is seen to be identified with higher probability. Similar is the result if sensor T_{24} is used as seen in Fig. 6.10(a) and Fig. 6.10(b). To resolve this ambiguity, the sensor information fusion framework makes use of both Ps_{30} and T_{24} to correctly identify the fault in the parameter space without any ambiguity, as seen in Fig. 6.11(a) and Fig. 6.11(b). The estimates lie in the ranges ($\pm 3\sigma$ bound) of 0.9747 to 0.9753 for ψ_{HPT} and ζ_{HPT} and 0.9847 to 0.9853 for ψ_{LPT} and ζ_{LPT}, respectively; in this case, highest probability is assigned to the training grid point that is nearest to the test point.

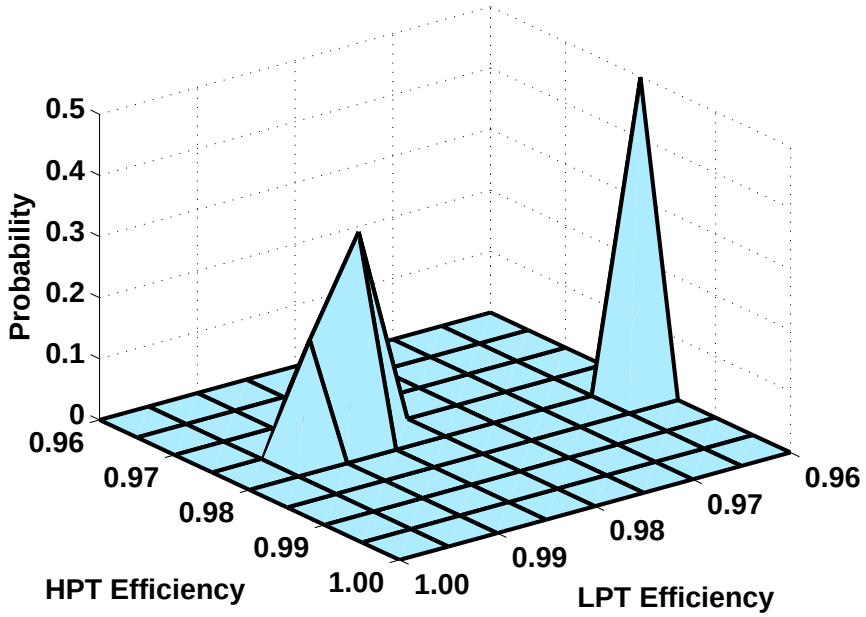


(a) Surface plot of fault estimation

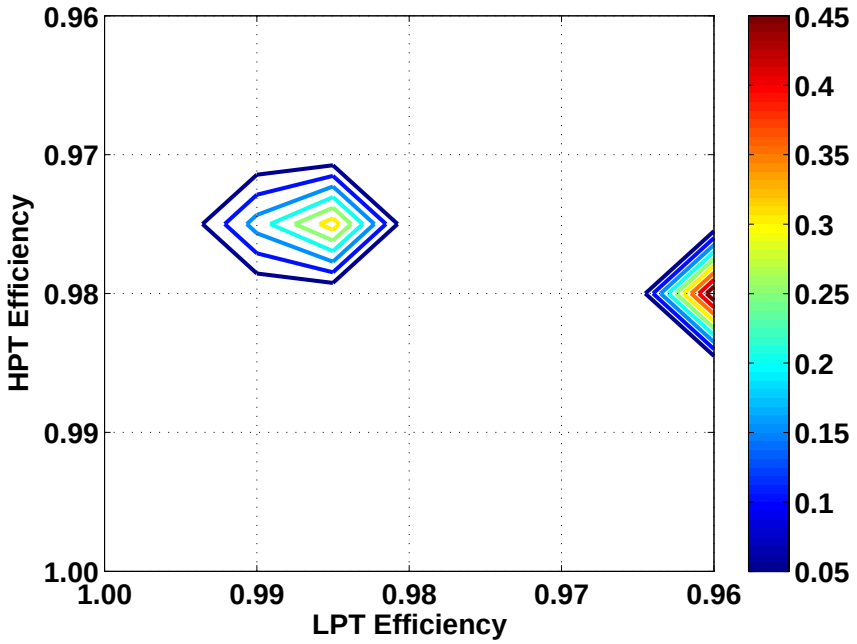


(b) Contour plot of fault estimation

Figure 6.8. Fault estimation in Fan-LPC based on Ps_{30} sensor

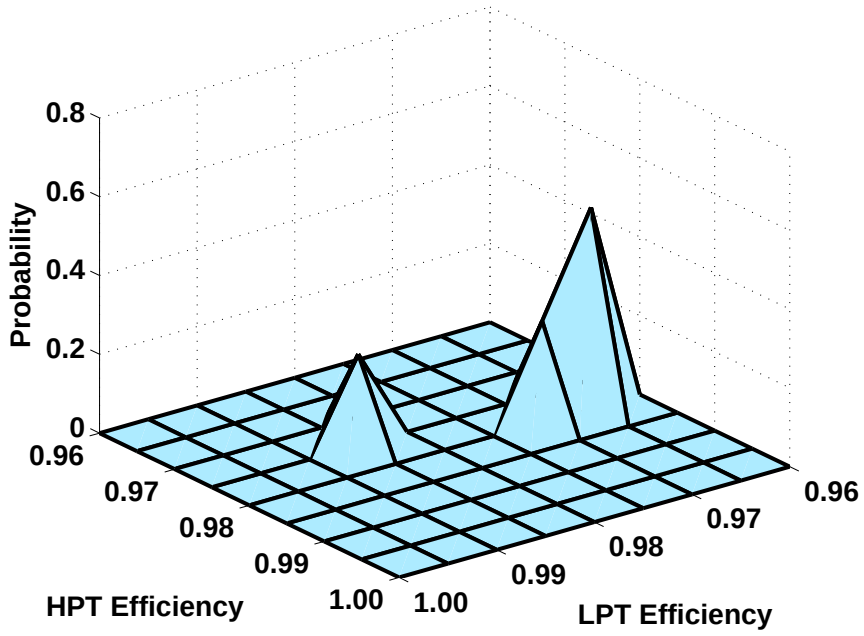


(a) Surface plot of fault estimation

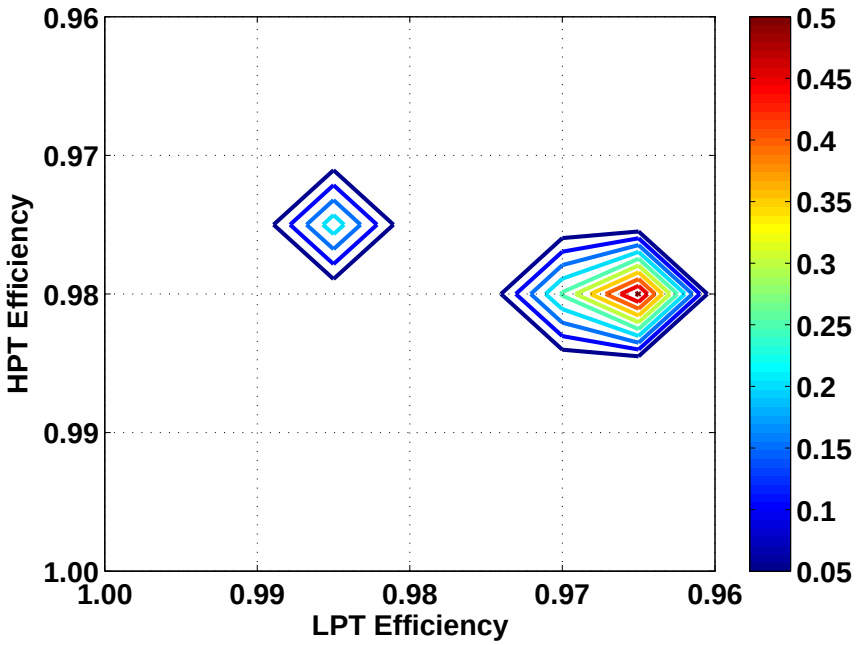


(b) Contour plot of fault estimation

Figure 6.9. Fault estimation in HPT-LPT based on Ps_{30} sensor

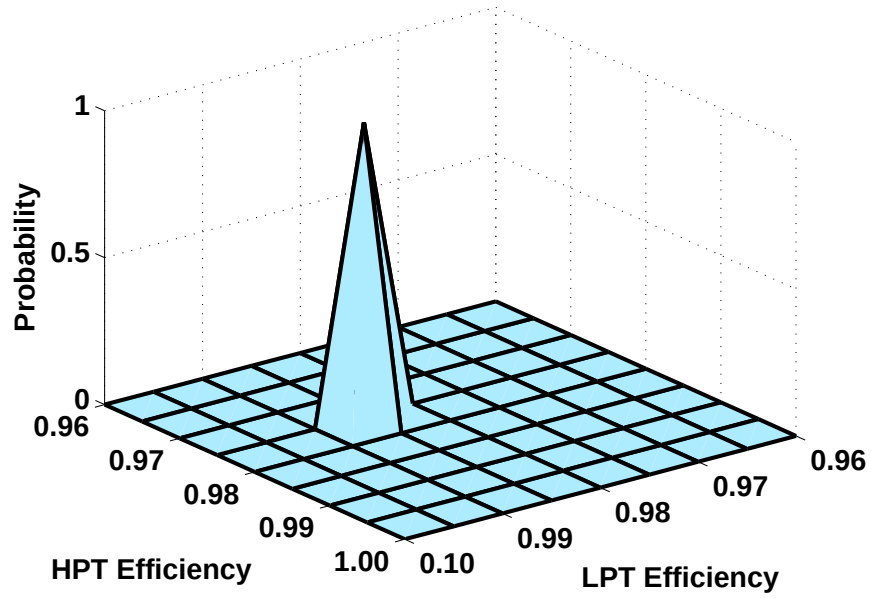


(a) Surface plot of fault estimation

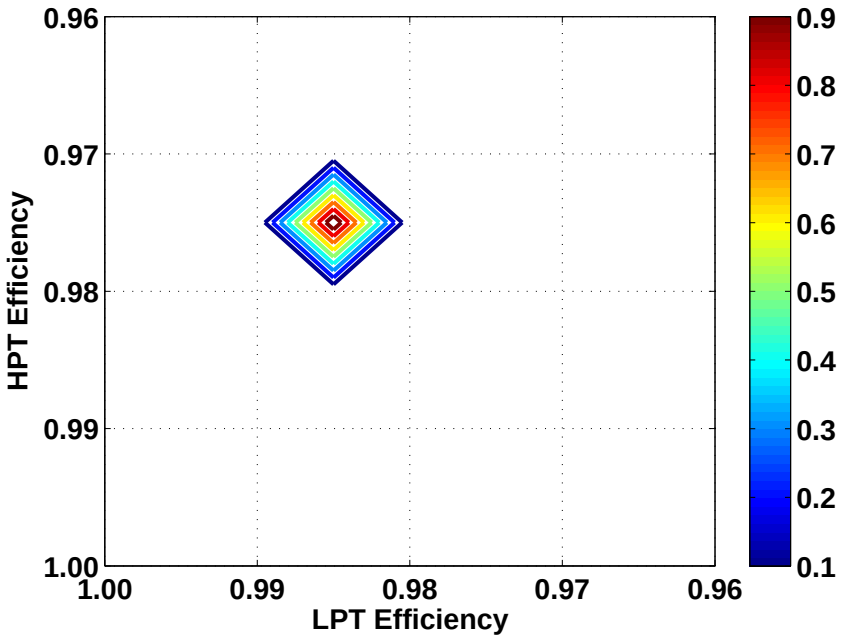


(b) Contour plot of fault estimation

Figure 6.10. Fault estimation in HPT-LPT based on T_{24} sensor



(a) Surface plot of fault estimation



(b) Contour plot of fault estimation

Figure 6.11. Fault estimation in HPT-LPT based on Ps_{30} and T_{24} sensors

Chapter 7

Summary and conclusions

This dissertation presents an extension of Symbolic Dynamic Filtering (*SDF*) for the statistical estimation of multiple parameters in nonlinear dynamical systems. It also presents a methodology to perform parameter estimation when the system is tapped by multiple sensors. The parameter estimation tool is sensor-data-driven and is suitable for applications such as early detection of parametric faults for prognosis of catastrophic failures in human-engineered systems. It has been shown that the training process of the *SDF*-based parameter estimation method is significantly less time-consuming than those of multi-layer-perceptron and radial-basis-function neural networks. This is so because the underlying algorithm of *SDF* makes use of a stopping rule [59,61,66] to limit the length of the time series and then compresses the pertinent information into pattern vectors of low dimension.

A method is described in the dissertation that provides a closed form solution of the estimated expected value and estimated covariance matrix of the parameter vector. This parameter estimation method can be implemented in a sensor network for real time execution on limited-memory small microprocessors, and also on the sensor itself, in the case of smart sensors.

This dissertation also presents a methodology for multiple-fault estimation in aircraft Gas turbine engines. It also proposes and validates a sensor-information-fusion framework to alleviate certain problems of data-level (i.e., related to scaling) and decision-level (i.e., resolution of detection) techniques for information fusion. The proposed fault estimation tool is sensor-data-driven and is apparently applicable for early detection of multiple faults for prognosis of catastrophic failures

in aircraft gas turbine engines. The underlying algorithm enables compression of information into pattern vectors of low dimension for real-time execution on limited-memory platforms.

The major contributions of this dissertation are listed below.

- Demonstration of the superiority of symbolic dynamic filtering to other methods commonly used such as Bayesian filtering, neural networks and statistical methods.
- Information fusion of observed evidence to obtain a statistical estimate of simultaneously varying parameters.
- Demonstration of possible non-uniqueness in parameter estimation, which may result due to selection of a scalar deviation measure as a cost functional in the multiple-parameter estimation problem.
- Using an underlying stopping rule to limit the length of the time series data used, to obtain real time parameter estimation
- Robustness to process noise, sensor noise and small fluctuations in parameter values, as discussed extensively in [3]
- Closed form solutions of estimated statistical parameters allowing for real-time execution on limited-memory platforms.

7.1 Directions for future work

While there are many other issues that need to be addressed before the proposed estimation method can be considered for industrial applications. The following research topics are being currently pursued.

7.1.1 Extensions to the parameter estimation methodology

Several extensions are possible for the parameter estimation methodology presented in this dissertation. The technique can be extended for different types of nonlinearities with structured and unstructured uncertainties. For example, only

one chaotic regime of the Duffing equation is presented. The technique can be extended to estimate parameters over a much larger range of parameter values of such a system. It is possible that different partitioning schemes would be required for different regions of the parameter space. Different fault types, such as sensor faults and actuator faults can be incorporated. Also, in this dissertation only the state probability vector has been used for estimation. This only measures the state occupation probabilities and does not take into account the transition probabilities between the different states. Using this information can possibly improve the robustness of the parameter identification framework.

An important extension would be to investigate the distribution used for the state probability vector. In this dissertation, the feature vector is fitted only to a Gaussian distribution. It is expected that the Dirichlet distribution [71] would provide a better fit for a probability vector. The Dirichlet distribution is the multivariate generalization of the beta distribution. It would also require fewer parameters to fit a Dirichlet distribution to a state probability vector than to fit a Gaussian dsitribution. For an n-dimensional feature vector, the Dirichlet distribution would require n independent parameters, while a full Gaussian distribution would require $\frac{(n-1)(n+2)}{2}$ parameters. However, it is a challenge to determine the multi-variate goodness of fit for a given Dirichlet distribution.

7.1.2 Theoretical Extension: Non-extensive thermodynamics and escort probabilities

Constantino Tsallis replaces Gibbs entropy with a non-extensive quantity, popularly known as Tsallis Entropy. For long range interactions, such as gravity, it is shown that energy is not extensive. This formalism is believed to be a natural framework for studying systems with fractal structure. In this framework, entropy is defined as:

$$S_q = \frac{1 - \sum_i p_i^q}{q - 1} = \sum_i \beta \left[b_i \right] p_i^q \tag{7.1}$$

where q is the degree of non-extensivity, and $[b_i]$ is defined as the bit-variance. Similarly, the escort probabilities are defined as:

$$P_i = \frac{\{1-(1-q)\beta\,[b_i]\}^{1/(1-q)}}{\sum_i \{1-(1-q)\beta\,[b_i]\}^{1/(1-q)}} \tag{7.2}$$

Now, these escort probabilities, defined in a non-extensive sense can provide a stronger basis for estimation than the normal case when $q = 1$. The reason for this is that differences in the Kullback distance are amplified by taking higher values of q, and the manifolds can be spread out in the higher dimensional space, thus making it easier to make estimates.

7.1.3 Fisher information and Application to Sensor Networks

The function $g_{\mu,\nu}$ described in Section D.3 supplies a local coordinate in the n-dimensional submanifold of the functional space of distributions. The Fisher metric is merely an induced metric on this manifold. Now, we can regard the order of the escort distribution q as a parameter in a one family distribution $\left(P_i^{(q)}\right)$. Now, measure the Fisher distance between $\left(P_i^{(q)}\right)$ and $\left(P_i^{(q+dq)}\right)$ to obtain:

$$\frac{\partial P_i^{(q)}}{\partial q} = \left(\mathbb{F}_q I - I_i\right) P_i^{(q)} \tag{7.3}$$

$\mathbb{F}_q I$ is the expected value of the information content, and $I_i = -\ln p_i$ is the information content. In the one dimensional case, we find:

$$D\left[P^{(q)}, P^{(q+dq)}\right] = (\Delta_q I)^2 dq^2 \tag{7.4}$$

The Fisher metric $(\Delta_q I)^2$ is the generalized variance of he bit content.

$$(\Delta_q I)^2 = < I^2 >_q - < I >_q^2 \tag{7.5}$$

The ordinary bit variance is recovered in the limit as $q \to 1$. The Fisher information is related to the problem of statistical parameter estimation. For an unbiased estimator of q, the error δq obeys the Cramer Rao inequality:

$$(\delta q)^2 \times (\Delta_q I)^2 \geq 1 \tag{7.6}$$

Hence, the generalized bit variance as the metric gives the fundamental limit for the precision of estimate of the order of the escort distribution.

There are two major challenges in sensor networking for information fusion

- The sensors generate more time series data than the network is capable of transmitting to the base station.
- Relative significance of data generated at a particular location could be highly variable

The first issue is addressed through the use of the symbolic domain, and a comprehensive evaluation has been provided in this proposal showing the benefits of working with symbols. The second issue is being addressed through thermodynamic quantities such as Fisher information and bit variance. The Cramer Rao inequality can be interpreted as follows: For each probability, there is a trade-off between sensitivity of the thermodynamic state to the system parameters, and the sensitivity of the probability to parameters such as sensor noise. Resource allocation can be determined using the formalism outlined above.

7.1.4 Sensor fusion using Cross D-Markov methods

There is an inherent loss of information when moving from a symbol sequence to a probabilistic finite state automaton (PFSA). The D-Markov machine is one method that is used in this dissertation, for both single and multiple sensors. However, several hierarchical fusion schemes can provide tractable solutions. One such method is the Cross D-Markov method being developed for co-dependence aware sensor fusion. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the $PFSA$s corresponding to symbol streams ω_1 and ω_2 respectively. The cross D-Markov machine $\mathcal{A}_{12}$ consists of:

- The states $\mathcal{Q}_1$ from $\mathcal{A}_1$ and the alphabet set Σ_2 from $\mathcal{A}_2$.
- The symbol generation matrix $\tilde{\Pi}_{12}$ is of size $|\mathcal{Q}_1| \times |\Sigma_2|$.
- The distance measure can be defined as $\mu = d(\tilde{\Pi}, \tilde{\Pi}_0)$ with respect to a reference symbol generation matrix.

This method can exploit access to symbol level co-dependence between different modalities to reduce information loss.

Appendix A

Construction of Anomaly Detection Algorithms

This appendix briefly reviews the rudimentary principles of commonly used pattern recognition tools that have been compared with SDF in Chapter 2.

A.1 Bayesian Filters

Bayesian filters provide a framework for state estimation for both linear and non-linear problems [46]. It is assumed that the states evolve according to a generalized model and generate discrete-time observations as:

$$x_{k+1} = f(x_k, w_k, u_k); \quad y_k = g(x_k, v_k, u_k) \tag{A.1}$$

where x_k is the state vector; y_k is the observation vector; u_k is the deterministic input; v_k is the observation noise; and w_k is the process noise. Given noisy observations, the state estimation problem involves determining a probability distribution for the system states, i.e., to determine $p(x_k|y_k)$. The following information is assumed to be available:

1. Initial probability distribution of the states $p(x_0)$
2. Functional form of the probability density $p(x_k|y_k)$
3. Probability distribution of the observation noise

Bayesian techniques largely follow a recursive predictor/corrector determination of the probability density function (*pdf*). The predictor stage forms the estimate $p(x_k|y_{k-1})$ while the corrector stage uses the most current observation and yields $p(x_k|y_k)$. The recursive determination implies that $p(x_k|y_k)$ is constructed from $p(x_{k-1}|y_{k-1})$. In a non-Markov setting, multiple observations generate the estimate, and construct $p(x_k|Y_k)$ where Y_k represents all observations from time 1 to k i.e. $Y_k = y_{1:k}$. Generalized recursive Bayesian state estimation is executed as:

$$p(x_k|Y_k) = \frac{p(y_k|x_k)p(x_k|Y_{k-1})}{p(y_k|Y_{k-1})} \tag{A.2}$$

where

$$p(x_k|Y_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|Y_{k-1})dx_{k-1} \tag{A.3}$$

depends on the density function $p(y_k|x_k)$ defined by measurement model and the known statistics of v_k. In the corrector stage, the measurement y_k is used to modify the prior density to obtain the required posterior density of the current state.

The recurrence relations in Eqs. (A.2) and (A.3) constitute a formal solution to the Bayesian state estimation problem. In general, these equations are not analytically solvable; hence, numerical approximations are sought. Note that an analytic solution exists for f and h being linear functions with additive white Gaussian noise v and w. This issue has been addressed by numerical approaches that include Particle Filtering and Sigma Point Kalman Filtering. These two methods are succinctly described below.

Particle Filtering attempts to approximate each distribution using a series of particles, and updates the distribution at each step, depending on the observation. Sigma Point Filtering makes a Gaussian assumption, and tries to model how the mean and covariance travel through a non-linear process using a series of points known as Sigma Points and an Unscented Transform. Due to this, the process is also known as Unscented Kalman Filtering. The steps needed to solve the Bayesian state estimation problem are succinctly described below.

A.1.1 Particle Filter (*PF*)

The particle filter [83] is a commonly used model based approach for anomaly detection. Two forms of the Particle Filter have been investigated - Sampling Importance Resampling (*SIR*), and Sampling Importance Sampling (*SIS*) [45]. Both algorithms involve generating a number of particles according to an initial distribution, and then passing these particles through an initial model of the system. After the first observation, the particles are weighted according to their Euclidean distance from the true observation. *SIS* and *SIR* filters differ in the stage where the particles are resampled. In *SIR* filtering, the particles are redistributed with particles of greater weight being given higher probabilities. In *SIS* filtering, the distribution is allowed to evolve without the effect of these weights. The histogram of these particles represents a multi-point approximation of the density function of the physical process evolving with time, and the mean and confidence intervals for the state estimates can be determined from this distribution. The particle filter algorithm is presented below.

1. Initialize time at $t = 0$ and sample N particles $\left\{x(t)^{(i)}\right\}_{i=1}^{N}$ from an initial distribution can be assumed to be Gaussian.

2. Generate N observations $\left\{y(t+1)^{(i)}\right\}_{i=1}^{N}$using the system and observation model.

3. Obtain the true observation $y(t+1)$ and compute weights $q(t)^{(i)} = p(y(t)|x(t|t-1))^{(i)}$ according to the distribution of the measurement noise, and normalize the weights: $\tilde{q}(t) = \frac{q(t)^{(i)}}{\sum q(t)^{(i)}}$

4. Resample the particles according to a new distribution that is specified by the normalized importance weights: $Pr(x(t|t))^{(i)} = Pr(x(t|t-1)^{(i)}) = \tilde{q}(t)^{(i)}$

5. Generate a new set of updated particles according to the distribution $p(x(t+1|t)|x(t|t)^{(i)}, y(t))$.

6. Increase t by 1 and repeat from step 2.

A.1.2 Unscented Kalman Filter (UKF)

The unscented Kalman filter [35] takes a different approach from the extended Kalman filter [44] in state estimation. In general, it is more convenient to approximate a probability distribution than it is to approximate an arbitrary nonlinear function or transformation. Following this argument, it is possible to generate a set of points whose sample mean and sample covariance are $\hat{x}_{k|k}$ and $P_{k|k}$ respectively. The nonlinear function is applied to each of these points in turn to yield a transformed sample, and the predicted mean and covariance are calculated from the transformed sample. The objective is to determine the output distribution by passing a few deterministically chosen points through the system, rather than a large number of stochastically chosen particles. Although this approach apparently resembles a Monte Carlo method, the samples are not drawn at random. Rather, the samples are deterministically chosen so that they capture specific information about the distribution. The system provides a Gaussian assumption of the output distribution, and hence is a form of a Kalman filter. One point is chosen for the mean of the system, and two points are chosen for calculating the variance in each dimension. These points are known as Sigma Points, and the principle involved is known as the Unscented Transform. Given the dimension n of the state space of the process, the equations for the Unscented Kalman Filter (UKF)are presented below.

1. Initialize:

$$\hat{x}_0 = E[x_0] \tag{A.4a}$$

$$P_0 = E[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T] \tag{A.4b}$$

2. For $k \in \{1, 2, \ldots, \infty\}$, calculate the sigma points

   (a) Sigma points update: $\chi_{k-1} = \left[\hat{x}_{k-1} \quad \hat{x}_{k-1} \pm \sqrt{(n+\kappa)P_{k-1}}\right]$

   (b) Time update: $\chi_{k|k-1} = f(\chi_{k-1}, u_{k-1}, k)$

(c) Predicted mean: $\hat{\chi}_{k|k-1} = W\chi_{k|k-1}$

(d) Predicted covariance update: $P_{k|k-1} = W.(\chi_{k|k-1} - \hat{x}_{k|k-1})$

where κ is the scale factor for output state dimension. This is set to be 3 for a two state system.

A.2 Neural Network Based Methods

Neural Network based fault detection and isolation techniques have been extensively investigated for last two decades. Generally neural networks are used to create a black-box model of the nominal system [84] to capture the possible fault signature(s) in the system [85]. An adaptive neural-network-based fault detection technique in nonlinear systems is presented in [86]. Liu and Scherpen presented a different fault detection technique for nonlinear systems based on probabilistic neural network filtering [49]. This paper investigates two different types of Neural Network algorithms, namely Multilayer Perceptron Neural Network (MLPNN) and Radial Basis Function Neural Network (RBFNN), for the detection of anomaly patterns.

A.2.1 Multi Layer Perceptron NN

A Multilayer Perceptron Neural Network is one of the simplest implementations of the back-propagation algorithm. It consists of a finite number of hidden layers of interconnected neurons and an output layer. The number of neurons in the output layer is same as the number of outputs from the network. Multilayer networks typically use sigmoid transfer functions, such as the logarithmic and tangent sigmoid functions, in the hidden layers. In the case of a neuron model with logarithmic sigmoid (LS) transfer function, LS takes an input vector $\{p_i\}$ with associated weights vector $\{w_i\}$ for$i = 1, 2 \ldots n$, then the output, a from the neuron will be, $a = LS\left(\sum_{i=1}^{n} w_i p_i + b\right)$ where bias to the neuron is b.

Output layer generally uses linear transfer functions. Networks with biases, sigmoid hidden layers, and a linear output layer are capable of approximating any

function with a finite number of discontinuities. Input vectors and targeted output vectors are used to train the network until it estimates the functional relation between the input and output up to a required accuracy. The training starts with an initial guess of weights and biases for each neuron. There are several types of back-propagation training algorithm. In each case, the network weights and biases are updated in the direction where the performance function decreases most rapidly, i.e. in the direction of the negative of the gradient. Normally, the performance function for feed-forward networks is the mean square error (MSE), i.e., the average squared error between the network outputs and the target outputs. For example, let x_j be the vector of weights and biases for one neuron layer, g_j be the gradient and α_j be the learning rate after the j^{th} iteration. Then, the $(j+1)^{th}$ iterated value of the weights and bias vector are: $x_{j+1} = x_j - \alpha_j g_j$.

Among different types of back-propagation algorithms, gradient descent or gradient descent with momentum are slow for certain problems [87]. In this paper, a relatively fast resilient back-propagation algorithm has been chosen, which uses only the sign of the gradient to determine the direction of the weight update. The advantage is that if the magnitude of the gradient becomes very small, the updating process continues in the correct direction until the weights and biases reach their optimal values [88].

A.2.2 Radial Basis Function NN

A Radial Basis Function Neural Network uses only one hidden layer and one output layer. It may require more number of neurons for the hidden layer compared to that required by a standard feed-forward network. However, it usually takes less time for training. Linear transfer function is used in the output layer and Radial Basis Function is used for the hidden layer. The transfer function for radial basis neuron is [50]: $f(x) = e^{-x^2}$. In this neural network algorithm the input to the radial basis transfer function is the vector distance between its weight vector w and the input vector p, multiplied by the bias b, i.e $x = d(w, p) \times p$. Sufficient number of neurons are added in the hidden layer to bring the sum-squared error below a threshold.

A.3 Statistical Pattern Recognition Techniques

There are many statistical pattern recognition techniques, of which two most common methods, namely, Principal component analysis (PCA) and Kernel Regression Analysis (KRA), have been investigated in this paper.

A.3.1 Principal Component Analysis

Principal component analysis (PCA), also known as proper orthogonal decomposition, is commonly used to reduce the dimensionality of a data set with a large number of interdependent variables [51]; PCA is the optimal linear transformation with respect to minimizing the mean square reconstruction error but it only considers second-order statistics [89].

Given a set of observed n-dimensional data points $\{x_1, x_2 \dots x_n\}$, the goal of PCA is to reduce the dimensionality of the observed vector x. This is realized by finding m principal axes $\{(p)^1, (p)^2 \dots (p)^m\}$ onto which the retained variance under projection is maximal.

The best known linear feature extraction technique, Principal Component Analysis (PCA) that makes use of Karhunen-Loéve transformation [89] to compute the m largest eigenvectors of the covariance matrix of the N patterns, each of which is m-dimensional. Since PCA uses the most expressive features (eigenvectors with the largest eigenvalues), it effectively approximates the data on a linear subspace using the mean squared error criterion.

A.3.2 Kernel Regression

Kernel regression method has been used to estimate the posteriori density function of the measurements and is proven to be superior to histogram density appraisers [90]. The kernel estimators are unbiased and smooth functions which are differentiable. Kernel functions are also used in conjunction with neural networks [90] and principal component analysis [51] in various fault detection and isolation applications.

A function $K(x) : \mathbb{R}^d \to \mathbb{R}$ is called a kernel function if $K(x)$ is limited and Borel measurable , i.e. if for $|x| \to \infty$, the following relation holds:

$$\left| \int_{\mathbb{R}^d} K(x) \right| < \infty \tag{A.5}$$

$K(x)$ is called normalized kernel function if K is unimodal, symmetric and non-negative i.e. $K(x) \geq 0 \forall x \in \mathbb{R}^d$ and the following condition holds : $\int_{\mathbb{R}^d} K(x)dx = 1$. Therefore, a normalized kernel function has the necessary properties of a density function. The following kernel functions are common:

1. Boxcar:

$$K(x) = \begin{cases} 0.5 & |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \qquad x \in \mathbb{R} \tag{A.6}$$

2. Cosine:

$$K(x) = \begin{cases} \frac{\pi}{4} cos\left(x\frac{\pi}{2}\right) & |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \qquad x \in \mathbb{R} \tag{A.7}$$

3. Gaussian:

$$K(x) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{x^2}{2}\right) \qquad x \in \mathbb{R} \tag{A.8}$$

Let $\{x_1, \ldots x_n\}$ be sampled from a distribution with density $f : \mathbb{R}^d \to \mathbb{R}$ and $K(x)$ be a normalized kernel function. Then, the univariate kernel estimator for the density $f(x)$ is defined as:

$$\hat{f}(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right) \tag{A.9}$$

where h is known as the window width or smoothing parameter and plays a similar role to the bin width in the case of a histogram.

Kernel regression technique provides a suite of nonlinear functions for density estimation [52]. They are also implicitly equivalent to radial basis functions in neural network literature. In the case of fault detection and isolation, kernel functions can essentially be used as a nearest neighbor type of classifier, where the activation of a cluster is determined by the distance between the input vector and the prototype vector. The kernel regression technique used herein for the fault

detection is extended to form a statistical model of the nominal data in terms of probability density estimate of the measurements.

As new data enters into the anomaly detection system, it is compared with the kernel regression of the density function of the nominal data. If it falls within the boundaries defined by the kernel estimator model, then it is considered as a nominal data; otherwise, the data is considered as faulty. A key requirement for kernel regression technique is appropriate selection of the kernel function and the order of the statistics of the model. From this perspective, a kernel function for fault detection is chosen as:

$$K(x) = exp\left(-\frac{1}{\theta_\alpha}\sum_i |x_i - \mu|^\alpha\right) \qquad x \in \mathbb{R} \tag{A.10}$$

where the parameters $\alpha \in (0, \infty)$ and (μ, θ_α) are the center and $\alpha^t h$ central moment of the data set, respectively., and are computed as:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{A.11a}$$

$$\theta_\alpha = \sum_{i=1}^{N} |x_i - \mu|^\alpha \tag{A.11b}$$

For $\alpha = 1, K(.)$ becomes logistic; and for $\alpha = 2, K(.)$ becomes Gaussian.

Appendix B

Experimental setup for Duffing and Vanderpol equations

This appendix provides a description of the electronic circuit for simulating the Duffing Equation. Experiments were conducted on the test bed [72] described in Section B.0.3. For the purpose of the experiment, a parameter is allowed to vary continuously by a small amount. This parametric change simulates a change in the health condition of the dynamical system. The objective of the experiment is to distinguish these small variations and estimate the value of the parameter. The experiments are repeated a number of times, by replacing a component (resistor) in the experimental setup each time with a similar component. This is done to establish robustness of the estimate with respect to component uncertainties.

B.0.3 Description of Experimental Apparatus

The experimental setup is a combination of electronic circuit designed with resistors (R), capacitors (C) and operational amplifiers and a computer interfaced with the circuit. The circuit consists of R-C networks which model the linear dynamics of the process, adders and voltage amplifiers. The nonlinearity is generated in the computer. The adder sums up the input signal and the terms generated by the computer, thereby making the overall system nonlinear.

Linear IC LM-741 chips were chosen to build the voltage amplifier and adder circuits since the operating frequencies are low (< 10 rad/sec). The R-C values of

both the networks were chosen to be equal. The resistance R was chosen to be 500 Ω and the capacitance C to be 1.00 mF. The software controlling the setup is coded in MATLAB. The software utilizes functions from the Data Acquisition toolbox of MATLAB. A single module of the test bed can model differential equations of order four. Multiple modules in cascade can be used to realize higher order equations.

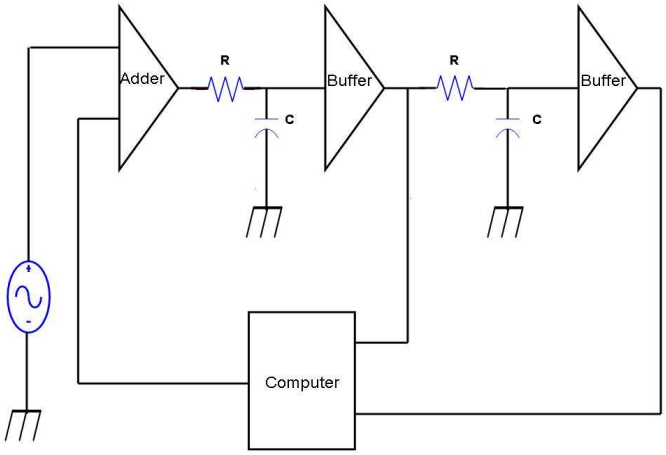


Figure B.1. Schematic of Experimental Setup

Keithley $KPCI1801 - HC$ plug-in board was used to interface the computer with the circuit. This board has the maximum sampling rate of 333 KSamples/sec, 64 A/D channels (12 bit), 2 D/A channels and operates in the range of -5V to $+5$V. The 12 bit A/D converters provide quantization levels of approximately 2.5mV. The process dynamics in the experiments were of the order of hertz(< 5 Hz). The process is sampled at 2000 Hz. The derivative and other terms were computed numerically using such samples. This plug-in board can be used in conjunction with the Data Acquisition toolbox of MATLAB. Figure B.1 provides the schematic of the test bed. Figures B.2 and B.3 provide the circuit diagrams.

B.0.4 Implementation of Duffing Equation System

The linear circuit is described by Equation (B.1)

$$\tau^2 \frac{d^2 v(t)}{dt^2} + 2\tau \frac{dv(t)}{dt} + v(t) = \tilde{A}\cos(\omega t) \tag{B.1}$$

where $v(t)$ is the voltage across the second capacitor, $\tau = RC$ is the time constant and $\tilde{A} = 5.5$.

The terms generated in the computer are:

1. $(2\tau - \beta\tau^2)\frac{dv(t)}{dt}$
2. $(1.0 - \tau^2)v$
3. $-\tau^2 v^3$

By adding these terms to the input,we get

$$\tau^2\frac{d^2v(t)}{dt^2} + 2\tau\frac{dv(t)}{dt} + v(t) = \tilde{A}\cos(\omega t) + (1.0 - \tau^2)v + (2\tau - \beta\tau^2)\frac{dv(t)}{dt} - \tau^2 v^3 \quad \text{(B.2)}$$

Rearranging the terms and dividing by τ^2,we obtain the duffing equation

$$\frac{d^2v(t)}{dt^2} + \beta\frac{dv}{dt} + v(t) + v^3(t) = A\cos(\omega t) \quad \text{(B.3)}$$

B.0.5 Implementation of van der Pol Equation System

The same circuit described above is also used to generate terms for the van der Pol equation. The equation for this is given below:

$$\frac{d^2x(t)}{dt^2} - \mu(1 - x^2(t))\frac{dx(t)}{dt} + \omega^2 x(t) = 0 \quad \text{(B.4)}$$

where nominal values of the parameters, to be estimated, are: $\mu = 1.0$ and $\omega = 1.0$.

The linear circuit is described by Equation (B.5)

$$\tau^2\frac{d^2v(t)}{dt^2} + 2\tau\frac{dv(t)}{dt} + v(t) = \tilde{A}\cos(\omega t) \quad \text{(B.5)}$$

where $v(t)$ is the voltage across the second capacitor, $\tau = RC$ is the time constant and $\tilde{A} = 5.5$.

The terms generated in the computer are:

1. $(2\tau - \beta\tau^2)\frac{dv(t)}{dt}$
2. $(1.0 - \tau^2)v$

3. $-\tau^2 v^3$

By adding these terms to the input,we get

$$\tau^2\frac{d^2v(t)}{dt^2}+2\tau\frac{dv(t)}{dt}+v(t)=\tilde{A}\cos(\omega t)+(1.0-\tau^2)v+(2\tau-\beta\tau^2)\frac{dv(t)}{dt}-\tau^2v^3 \quad \text{(B.6)}$$

Rearranging the terms and dividing by τ^2,we obtain the duffing equation

$$\frac{d^2v(t)}{dt^2}+\beta\frac{dv}{dt}+v(t)+v^3(t)=A\cos(\omega t) \quad \text{(B.7)}$$

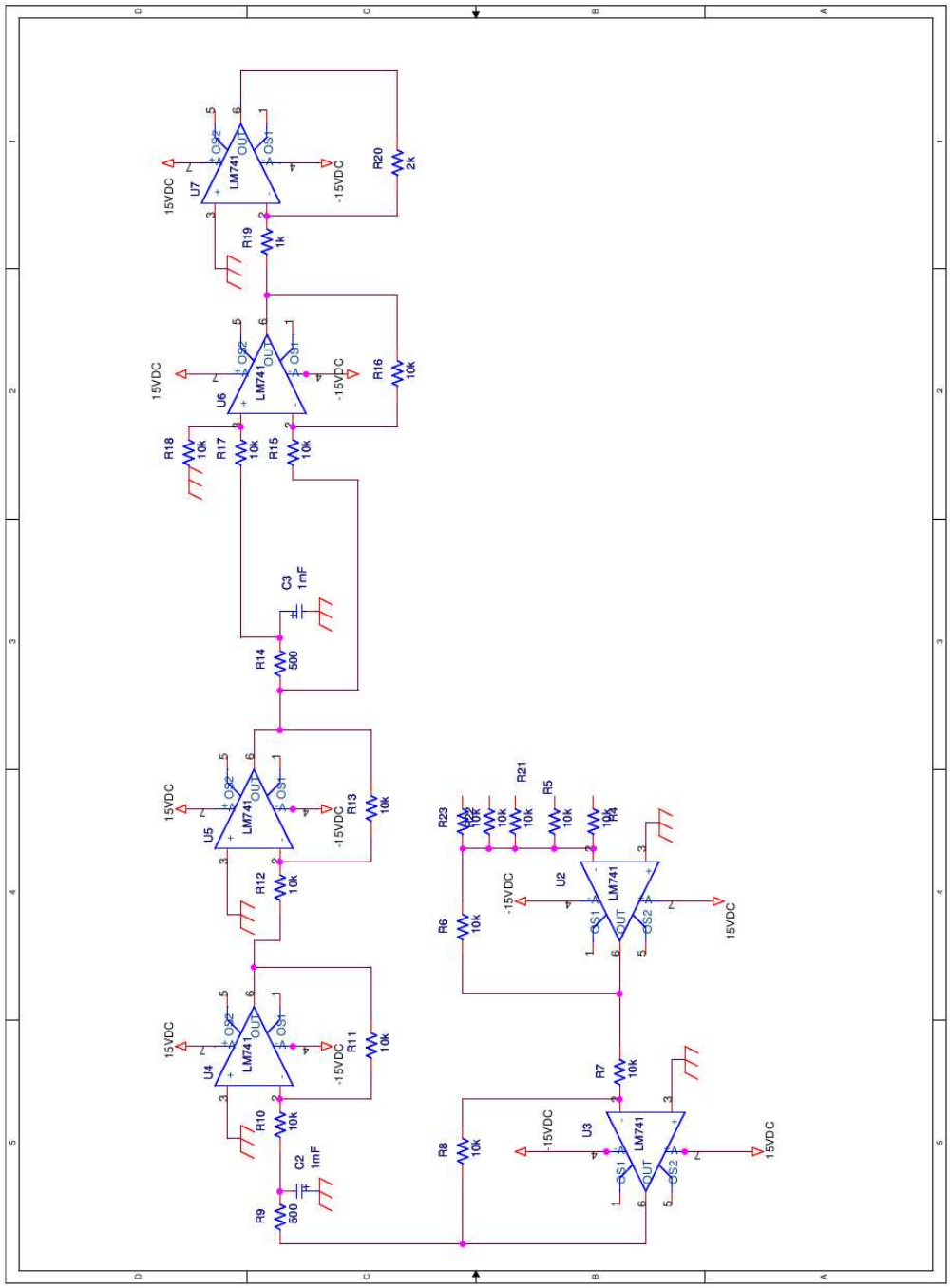


Figure B.2. Circuit for 2nd order systems using op-amps as integrators

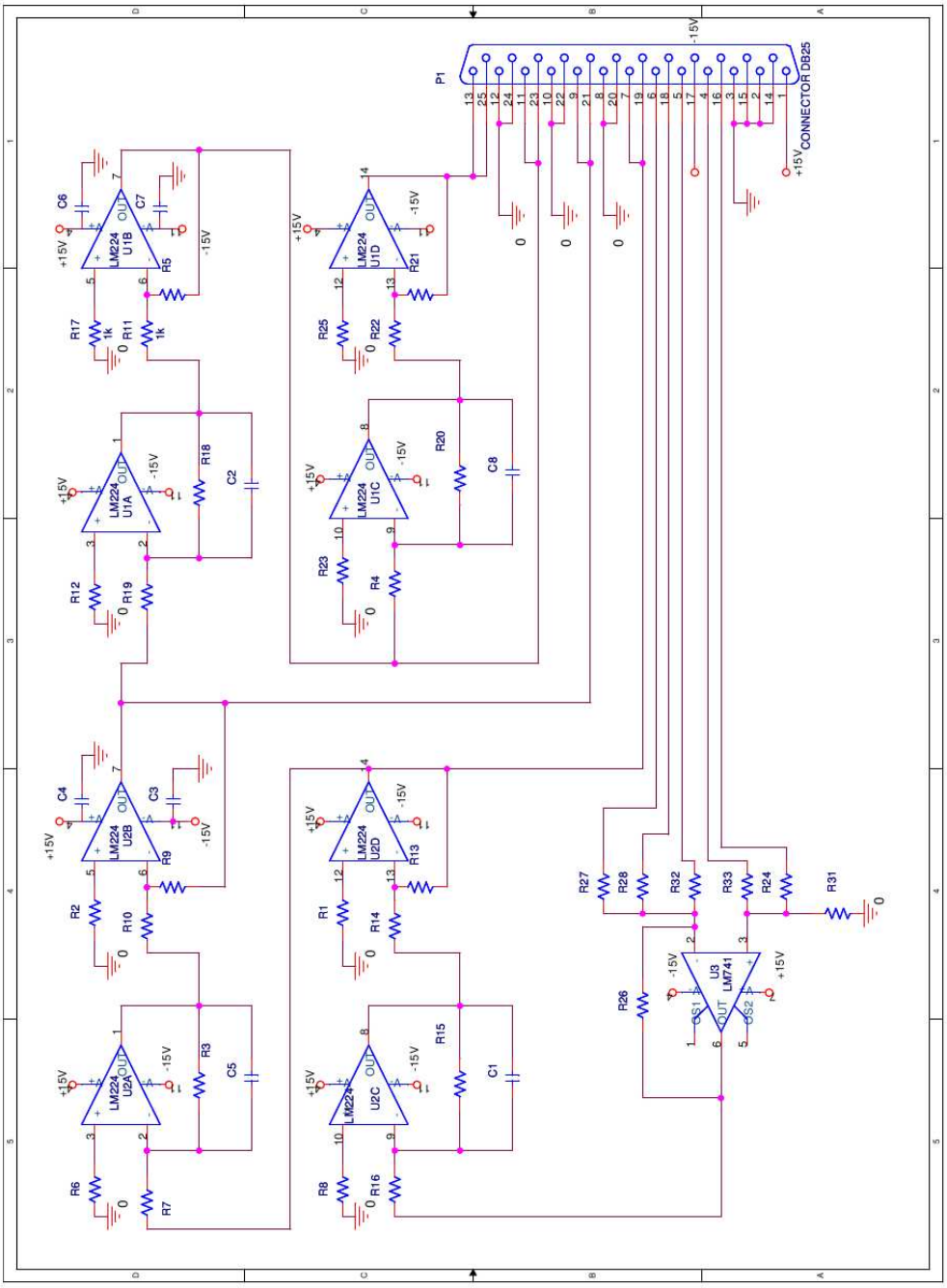


Figure B.3. Circuit for 2nd order systems using op-amps as integrators

Appendix C

Review of multi-class classification techniques

This appendix provides a review of multi-class classification techniques. Multi-class classification is a problem in machine learning which studies the assignment of one of several class labels to an input object. Unlike a better understood problem of binary classification, which requres discerning between the two given classes, the multiclass one is a more complex and less researched problem. Two broad approaches to the problem are direct approaches, or generalization of binary classification approaches. Direct approaches include nearest/k-nearest neighbor methods, naive bayes methods, and linear approaches such as perceptrons, polychotomous regression and support vector machines. Generalized binary classification approaches include "One versus All" (OvA) and "All versus All" (AvA) methods.

C.1 Direct Approaches

C.1.1 k-Nearest Neighbors Algorithm

In pattern recognition, the *k-nearest neighbours algorithm (k-NN)* is a method for classifying objects based on closest training examples in the feature space. *k-NN* is a type of instance based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification. The k-nearest neighbor algorithm is amongst the simplest of all machine learning

algorithms: an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of its nearest neighbor.

The *k-NN* algorithm can also be adapted for use in estimating continuous variables. One such implementation uses an inverse distance weighted average of the k-nearest multivariate neighbors. This algorithm functions as follows:

1. Compute Euclidean or Mahalanobis distance from target plot to those that were sampled.

2. Order samples taking for account calculated distances.

3. Choose heuristically optimal k nearest neighbor based on RMSE done by cross validation technique.

4. Calculate an inverse distance weighted average with the k-nearest multivariate neighbors.

The optimal k for most datasets is 10 or more. That produces much better results than *1-NN*. Using a weighted *k-NN*, where the weights by which each of the k nearest points class (or value in regression problems) is multiplied are proportional to the inverse of the distance between that point and the point for which the class is to be predicted also significantly improves the results.

C.1.2 Naive Bayes Classifier

A Bayes classifier is a simple probabilistic classifier based on applying Bayes theorem with strong independence assumptions. A more descriptive term for the underlying probability model would be independent feature model. That is, the naive Bayes classifier assumes that the presence or absence of a particular feature of a class is unrelated to the presence or absence of any other feature.

Depending on the precise nature of the probability model, naive Bayes classifiers can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for naive Bayes models uses the method of maximum likelihood; in other words, one can work with the naive Bayes model

without believing in Bayesian probability or using any Bayesian methods. An advantage of the naive Bayes classifier is that it requires a small amount of training data to estimate the parameters i.e. the means and variances of the variables necessary for classification. Because independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.

Assume that the dependent class variable is denoted as C with a small number of outcomes or classes. The observed features F are denoted as F_1, $F_2 \cdots F_n$. The probability model for the classifier is then the conditional model $P(C|F_1, \ldots, F_n)$. The problem is that if the number of features n is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. Bayes rule is used to reformulate this model.

$$P(C|F_1, \ldots, F_n) = \frac{P(C)P(F_1, \ldots, F_n|C)}{P(F_1, \ldots, F_n)} \tag{C.1}$$

The denominator does not depend on C and the values of the features F_i are given, so the denominator is effectively constant. The numerator is equivalent to the joint probability model $P(C, F_1, \ldots, F_n)$. This can be rewritten as follows, using repeated applications of the definition of conditional probability.

$$P(C, F_1, \ldots, F_n) = P(C)P(F_1|C)P(F_2|C, F_1) \cdots P(F_n|C, F_1, F_2 \cdots, F_{n-1}) \tag{C.2}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$. This can be stated as follows:

$$P(F_i|C, F_j) = P(F_i|C) \tag{C.3}$$

So, the joint model simplifies to

$$P(C, F_1, \cdots, F_n) = P(C)P(F_1|C)P(F_2|C)P(F_3|C) \cdots \tag{C.4}$$

This means that under the above independence assumptions, the conditional

distribution over the class variable C can be expressed like this:

$$P(C|F_1, \cdots, F_n) = \frac{1}{Z} P(C) \prod_{i=1}^{n} P(F_i|C) \tag{C.5}$$

where Z is the evidence which is a scaling factor dependent only on $F_1, \ldots, F_n$, i.e., a constant if the values of the feature variables are known. The total number of parameters of the naive Bayes model is $2n + 1$, where n is the number of binary features used for classification and prediction.

The above equations represent the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the maximum a-posteriori or MAP decision rule. The corresponding classifier is the function defined as follows:

$$\text{classify}(f_1, \ldots, f_n) = \underset{c}{\text{argmax}}\ P(C = c) \prod_{i=1}^{n} P(F_i = f_i|C = c). \tag{C.6}$$

C.1.3 Linear methods

The goal of classification is to use an object's features to identify which class it belongs to. A linear classifier achieves this by making a classification decision based on the value of a score, which is simply a linear combination of the underlying features. An object's characteristics are typically presented to the machine in a vector called a feature vector. If the input feature vector to the classifier is a real vector $\overrightarrow{x}$, then the output score is:

$$y = f(\overrightarrow{w} \cdot \overrightarrow{x}) = f\left(\sum_j w_j x_j\right) \tag{C.7}$$

where $\mathbf{w}$ is a real vector of weights and f is a function that converts the dot product of the two vectors into the desired output. The weight vector $\overrightarrow{w}$ is learned from a set of labeled training samples. Often f is a simple function that maps all values above a certain threshold to the first class and all other values to the second class. A more complex f might give the probability that an item belongs to a certain class. For a two-class classification problem, one can visualize the operation

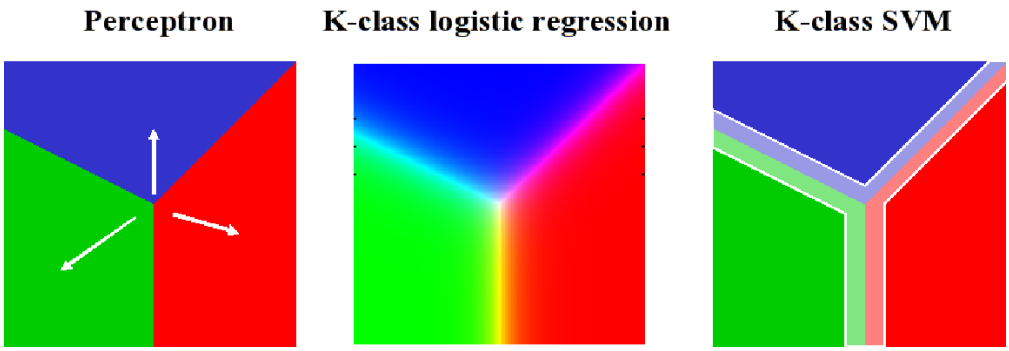


Figure C.1. Linear classification methods

of a linear classifier as splitting a high-dimensional input space with a hyperplane: all points on one side of the hyperplane are classified as "yes", while the others are classified as "no". A linear classifier is often used in situations where the speed of classification is an issue, since it is often the fastest classifier, especially when $\overrightarrow{x}$ is sparse. The different types of linear classifiers are summarized in Figure C.1.

Perceptron based methods and support vector machines are further elaborated in the following subsections.

C.1.3.1 Perceptron based linear methods

In this method, first the feature vector is multiplied by a weight vector $\overrightarrow{w}$, but now the resulting score is used to choose among many possible outputs.

$$\hat{y}_i = \underset{y}{\operatorname{argmax}}\ w_y^T x^i \tag{C.8}$$

Learning again iterates over the examples, predicting an output for each, leaving the weights unchanged when the predicted output matches the target, and changing them when it does not. The update becomes:

$$w_{t+1} = w_t + \alpha\left(f(x, y) - f(x, \hat{y})\right) \tag{C.9}$$

α is the learning rate of the perceptron. The advantages of this method include extremely simple updates (i.e. there is no need to calculate a gradient). Also, it is not required to store all the data in memory. However, for data sets that are

not perfectly seperable, the value of α would need to be very small to get a well trained perceptron.

C.1.3.2 Support Vector Machines

A support vector machine (SVM) constructs a hyperplane or set of hyperplanes in a high dimensional space, which is then used for classification. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training datapoints of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. The training set consists of L training points, where each input x_i is D dimensional and is in one of two classes $y_i = 1$ or $+1$.

The aim is to find the maximum margin hyperplane that divides the points having $y_i = 1$ from those having $y_i = 1$. Any hyperplane can be written as the set of points satisfying

$$\overrightarrow{w}.\overrightarrow{x} - b = 0 \tag{C.10}$$

The vector $\overrightarrow{w}$ is a normal vector, that is, it is perpendicular to the hyperplane. The parameter $\frac{b}{||\overrightarrow{w}||}$ determines the offset of the hyperplane from the origin along the normal vector $\overrightarrow{w}$. Support Vectors are the examples closest to the separating hyperplane and the aim of Support Vector Machines (SVM) is to orientate this hyperplane in such a way as to be as far as possible from the closest members of both classes.

Implementing a SVM boils down to selecting the variables $\overrightarrow{w}$ and b. If the training data points are linearly separable, the two hyperplanes of the margin can be selected in a way that there are no points between them and then their distance can be maximized.

C.2 Generalization of binary classification approaches

C.2.1 One vs. All approaches

One of the simplest multiclass classification schemes built on top of real-valued binary classifiers is to train N different binary classifiers, each one trained to

distinguish the examples in a single class from the examples in all remaining classes. When it is desired to classify a new example, the N classifiers are run, and the classifier which outputs the largest(most positive) value is chosen. This scheme will be referred to as the one-vs-all or OVA.

C.2.2 All vs. All approaches

Another scheme used for multiclass classification is the all-pairs, or AVA (all-vs- all) scheme. In this approach, $\binom{n}{2}$ binary classifiers are trained; each classifier separates a pair of classes. This scheme, like the OVA scheme, has a simple conceptual justification, and can be implemented to train faster and test as quickly as the OVA scheme.

Classification of new instances for one-versus-all case is done by a winner-takes-all strategy, in which the classifier with the highest output function assigns the class (it is important that the output functions be calibrated to produce comparable scores). For the AVA approach, classification is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with most votes determines the instance classification.

Appendix D

Motivation from Thermodynamics Principles for Multiple Parameter Estimation

In order to solve the problem of estimation of multiple parameters, the principles of thermodynamics and statistical mechanics [20] can be used. In statistical mechanics, a few macroscopic parameters (e.g. pressure and temperature) are used to describe the intrinsic dynamics of the entire system in terms of the estimates derived from the distribution of the elementary particles in various micro states. In the same fashion, the behavior of a dynamical system can be investigated both from microscopic and macroscopic points of view. In the study of a dynamical system, the measured time series data of the observable parameters on the fast time scale can be analyzed to generate the pattern vectors in terms of the probability distributions, which can be used to describe the macroscopic or global behavior of the system at a particular slow time epoch. The information derived from these pattern vectors can be further compressed into a few macroscopic parameters such as entropy, Kullback distance, and Euclidean norm. This analogy is further explained below.

Statistical Mechanics $\Rightarrow$ distribution of microstates $\rightarrow$ macroscopic properties
Dynamical System$\Rightarrow$ pattern vectors from time series data $\rightarrow$ System Parameters

A window of time series data points, chosen on a fast time scale at a particular

slow time epoch, can be considered to be analogous to a thermodynamic system consisting of the elementary particles. The activities of these individual particles in the thermodynamic system and their possible interactions with each other determine the behavior of the entire system which can be defined by a few macroscopic parameters (e.g., pressure and temperature). Similarly, in the case of a dynamical system, the macroscopic behavior can be derived from the configuration of these time series data points inside the window. At a particular slow time epoch, the properties of a dynamical system can be expressed by a few parameters which can be estimated by the statistical patterns in the time series data points.

The pattern identification procedure of a quasi-stationary dynamical process is recognized as a two-time-scale (i.e., fast and slow time scale) problem. The thermodynamic interpretation of this two-time-scale approach is that, on the fast time scale, the analogous thermodynamic system stays on the same energy surface, i.e. the characteristics of the system can be described by one equivalence class. Therefore, the macroscopic properties remain constant over the fast time scale. However, on the slow time scale, the macroscopic properties of the system change due to the evolution of multiple anomalies (if any) and therefore the behavior of the system is described by different equivalence classes.

D.1 Gibbs Canonical Distribution

A canonical ensemble in statistical mechanics is an ensemble (a large number of mental copies of a system, representing in effect a probability distribution for the exact microscopic state of the system), that is characterised by the proportion p_i of members of the ensemble occupying the state i being given by the Boltzmann distribution.

$$p(E) = exp(G - \beta E) \tag{D.1}$$

where G is a normalization constant, and β is the inverse temperature.

It can be shown that this is the distribution which is most likely, if each system in the ensemble can exchange energy with a heat bath, or alternatively with a large number of similar systems. Equivalently, it is the distribution which has maximum

entropy for a given average energy E_i.

It is also referred to as an NVT ensemble: the number of particles (N), the volume (V), of each system in the ensemble are the same, and the ensemble has a well defined temperature (T), given by the temperature of the heat bath with which it would be in equilibrium.

The quantity k is Boltzmann's constant, which relates the units of temperature to units of energy i.e. $\beta = 1/kT$.

The quantities G and Z are constants for a particular ensemble, which ensure that Σp_i is normalised to 1. Z is therefore given by:

$$Z = \sum \exp(-E_i/kT) = \sum \exp(-\beta E_i) \tag{D.2}$$

This is called the partition function of the canonical ensemble. Specifying this dependence of Z on the energies E_i conveys the same mathematical information as specifying the form of p_i above. The canonical ensemble (and its partition function) is widely used as a tool to calculate thermodynamic quantities of a system under a fixed temperature.

D.2 Escort Probabilities and Distributions

Escort distributions scan the attributes of the original distribution, while describing the features of a non-linear dynamical system. [91]. Let $\{p_i\}$ be the original distribution. Then its escort is given by:

$$P_i = \frac{\phi(p_i)}{\sum_j \phi(p_j)} \tag{D.3}$$

where ϕ is a positive function.

This equation comes about when we consider that the Renyi information is a monotonically increasing function of β [20]. An important case occurs if $\phi(s) = s^q$, for $0 < s \leq 1$ and $q > 0$ then $P_i^{(q)} \equiv P_i$ and

$$P_i^{(q)} = \frac{(p_i)^q}{\sum_j (p_j)^q} \tag{D.4}$$

Expectations with respect to the original distribution p are denoted as $\mathbb{E}_p$. Also,

expectations with respect to the escort distribution $P^{(q)}$ are denoted as $\mathbb{F}_q$. More formally,

$$\mathbb{E}_p f = \int_\Omega d\mu(x) p(x) f(x) \tag{D.5}$$

and

$$\mathbb{F}_q f = \int_\Omega d\mu(x) P^{(q)}(x) f(x) \tag{D.6}$$

Now, $p_i \overset{q}{\rightarrow} P_i^{(q)}$ can be regarded as a transformation.

$$P_i^{(q)} \overset{r}{\rightarrow} = \frac{(p_i)^{qr}}{\sum_j (p_j)^{qr}} = P_i^{(qr)} \tag{D.7}$$

This transformation forms a one-parameter Abelian group with the identity transformation corresponding to the order unity. The parameter is obviously, q.

In the context of Symbolic Dynamics, it is proposed that the State Probability Vector corresponds to the original distribution, $\{p_i\}$, while the escort distribution is $P_i^{(q)} = \frac{(p_i)^q}{\sum_j (p_j)^q}$ where q is defined as the Tsallis degree of non-extensivity of the complex dynamical system. In the two cases considered in this proposal, $q = 1$ and the original distribution and the escort distribution are the same.

D.3 Parameter Estimation using Escort Probabilities

The concept of Fisher information is introduced in a thermodynamic sense, which is then applied to the problem of parameter estimation. The discussion follows the principles embodied in [20,91]. In statistics and information theory, the Fisher information (denoted $\mathcal{I}(\beta)$) is the variance of the score. The Fisher information is the amount of information that an observable random variable X carries about an unknown parameter β upon which the likelihood function of X, $\mathcal{I}(\beta) = f(X;\beta)$, depends. The likelihood function is the joint probability of the data, the Xs, conditional on the value of β, as a function of β. Since the expectation of the score is zero, the variance is simply the second moment of the score, the derivative of

the log of the likelihood function with respect to β.

$$\mathcal{I}(\beta) = \mathrm{E}\left\{ \left[\frac{\partial}{\partial\beta} \ln f(X;\beta)\right]^2 \middle| \beta \right\}, \tag{D.8}$$

This implies $0 \leq \mathcal{I}(\beta) < \infty$.

To introduce a thermodynamic formalism based definition of the Fisher information, it is necessary to first define Kullback information and the Kullback Distance.

The Kullback Liebler relative entropy is defined for two distributions π and π^a as:

$$K\left[\pi \parallel \pi^a\right] = \sum_i \pi_i \ln \frac{\pi_i}{\pi_i^a} \tag{D.9}$$

This is positive definite and vanishes only if $\pi_i = \pi_i^a \forall i$.

The Kullback Liebler divergence is defined for the same two distributions as:

$$D\left[\pi, \pi^a\right] = K\left[\pi \parallel \pi^a\right] + K\left[\pi^a \parallel \pi\right] \tag{D.10}$$

Let π_i depend on a set of parameters $\mathbf{q}$. That is, let $\pi_i = \pi_i(\mathbf{q})$, where $\mathbf{q} = (q^1, q^2, \ldots, q^n)$.

Let π_i^a represent $\pi_i(\mathbf{q} + d\mathbf{q})$. $D[\pi, \pi^a]$ is calculated as:

$$D\left[\pi, \pi^a\right] = \sum_{\mu,\nu=1}^{n} g_{\mu\nu}(\mathbf{q}) dq_\mu dq_\nu \tag{D.11}$$

where $g_{\mu\nu}$ is given as:

$$g_{\mu\nu}(\mathbf{q}) = \sum_i \frac{\partial_\mu \pi_i(\mathbf{q}) \partial_\nu \pi_i(\mathbf{q})}{\pi_i(\mathbf{q})} \tag{D.12}$$

$g_{\mu\nu}$ is defined as the Fisher Information, or the Fisher metric, and $\partial_\mu = \frac{\partial}{\partial q_\mu}$. $\mathbf{q}$ supplies a local coordinate in the n-dimensional submanifold of the functional space of distributions. The Fisher metric is merely an induced metric on this manifold.

Bibliography

[1] Thelin, P. (2002) "Short circuit fault conditions of a buried PMSM investigated with FEM," in *NORPIE/2002, Stockholm, Sweden*.

[2] Ray, A. (2004) "Symbolic Dynamic Analysis of Complex Systems for Anomaly Detection," *Signal Processing*, **84**(7), pp. 1115–1130.

[3] Rajagopalan, V. and A.Ray (2006) "Symbolic Time Series Analysis via Wavelet-Based Partitioning," *Signal Processing*, **86**(11), pp. 3309–3320.

[4] Khatkhate, A., A. Ray, S. Chin, V. Rajagopalan, and E. Keller (June-July 2004) "Detection of Fatigue Crack Anomaly: A Symbolic Dynamic Approach," *Proceedings of American Control Conference, Boston, MA*, pp. 3741–3746.

[5] Ding, Y., Z. Wu, and Y. Zhang (2001) "Multi-fault diagnosis method based on a joint estimation of states and fault parameters," *Journal of Tsinghua University*, **41**(12), pp. 92–94.

[6] Isermann, R. (2005) "Model-based fault-detection and diagnosis - status and applications," *Annual Reviews in Control*, **29**(1), pp. 71–85.

[7] Aitouche, A., D. Maquin, and F. Busson (1-4 Sep 1998) "Multiple sensor fault detection in heat exchanger systems," *Proceedings of the 1998 IEEE International Conference on Control Applications*, **2**, pp. 741–745.

[8] Ghosh, A., V. Kumar, and B. Kulkarni (2001) "Parameter estimation in spatially extended systems: The Karhunen-Loeve and Galerkin multiple shooting approach," *Physical Review E*, **64**, p. 056222.

[9] Huang, H.-P., C.-C. Li, and J.-C. Jeng (2007) "Multiple Multiplicative Fault Diagnosis for Dynamic Processes via Parameter Similarity Measures," *Industrial & Engineering Chemistry Research*, **46**(13), pp. 4517–4530.

[10] Koh, C., J. Shi, W. Williams, and J. Ni (1999) "Multiple Fault Detection and Isolation Using the Haar Transform, Part 1: Theory," *Journal of Manufacturing Science and Engineering*, **121**(2), pp. 290–294.

[11] Gupta, S., A. Ray, S. Sarkar, and M. Yasar (2008) "Fault Detection and Isolation in Aircraft Gas Turbine Engines: Part I - The Underlying Concept," *Proceedings of the IMechE-Part G: Journal of Aerospace Engineering,*, **222**(3), pp. 307–318.

[12] Ott, E. (1993) *Chaos in dynamical systems*, Cambridge: Cambridge University Press, 1993.

[13] Abarbanel, H. D. I., R. Brown, J. J. Sidorowich, and L. S. Tsimring (1993) "The analysis of observed chaotic data in physical systems," *Reviews of Modern Physics*, **65**, pp. 1331–1392.

[14] Dragoman, D. (1999) "Complexity: Hierarchical Structures and Scaling in Physics, by R. Badii and A. Politi," *Optics & Photonics News*, **10**, pp. 55–+.

[15] Lind, D. and M. Marcus (1995) *An Introduction to Symbolic Dynamics and Coding*, Cambridge University Press.

[16] H.Kantz and T.Schreiber (1997) "Nonlinear Time Series Analysis," *Cambridge University Press*.

[17] Abarbanel, H. D. I. (1996) *The Analysis of Observed Chaotic Data*, Springer-Verlag, New York.

[18] Badii, R. and A. Politi (1997) *Complexity, Hierarchical Structures and Scaling in Physics*, Cambridge University Press, Cambridge, U.K.

[19] C.S.Daw, C.E.A.Finney, and E.R.Tracy (2003) "A review of symbolic analysis of experimental data," *Review of Scientific Instruments*, **74**(2), pp. 915–930.

[20] Beck, C. and F. Schögel (1993) *Thermodynamics of Chaotic Systems: An Introduction*, Cambridge University Press.

[21] N.Tufillaro (1999) "Symbolic Dynamics in Mathematics, Physics, and Engineering," *HP Labs Technical Reports*, **HPL-1999-28**.

[22] R.L.Davidchack, Y.C.Lai, E.M.Bolt, and H.Dhamala (2000) "Estimating generating partitions of chaotic systems by unstable periodic orbits," *Physical Review*, **75**, pp. 1353–1356.

[23] Kennel, M. B. and M. Buhl (2003) "Estimating Good Discrete Partitions from Observed Data: Symbolic False Nearest Neighbors," *Physical Review Letters*, **91**(8), p. 084102.

[24] J.P.Crutchfield and K.Young (1989) "Inferring Statistical Complexity," *Physical Review Letters*, **63**, pp. 105–108.

[25] C.R.Shalizi, K.L.Shalizi, and J.P.Crutchfield (2002) "An Algorithm for Pattern Discovery in Time Series," *SFI Working Paper 02-10-060*.

[26] P.M.T.Broersen (1976) "Estimation of Parameters of Non-linear Dynamical Systems," *Int. J. Non-linear Mechanics*, **9**, pp. 355–361.

[27] Van Lith, P., H. Witteveen, B. Betlem, and B. Roffel (2001) "Multiple nonlinear parameter estimation using PI feedback control," *Control Engineering Practice*, **9**, pp. 517–531.

[28] J.Ching, J.L.Beck, and K.A.Porter (2006) "Bayesian state and parameter estimation of uncertain dynamical systems," *Probabilistic Engineering Mechanics*, **21**(1), pp. 81–96.

[29] Gupta, S. and A. Ray (2007) *Symbolic Dynamic Filtering for Data-Driven Pattern Recognition*, Chapter 2 in Pattern Recognition: Theory and Application, Editor- E.A. Zoeller, Nova Science Publisher, Hauppage, NY, USA.

[30] Tang, X. Z., E. R. Tracy, and R. Brown (1997) "Symbol statistics and spatio-temporal systems," *Physica D: Nonlinear Phenomena*, **102**(3-4), pp. 253–261.

[31] Piccardi, C. (2006) "On parameter estimation of chaotic systems via symbolic time-series analysis," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, **16**(4), 043115.

[32] Boppana, V., I. Hartanto, and W. K. Fuchs (1996) "Fault diagnosis using state information," in *FTCS '96: Proceedings of the The Twenty-Sixth Annual International Symposium on Fault-Tolerant Computing (FTCS '96)*, IEEE Computer Society, Washington, DC, USA, p. 96.

[33] Broersen, P. (1974) "Estimation of parameters of non-linear dynamical systems," *International Journal of Non-linear Mechanics*, **9**, pp. 355–361.

[34] E.Wan and R. Merwe (2000) "The Unscented Kalman Filter for Nonlinear Estimation," *Proc. of IEEE Symposium 2000 (AS-SPCC), Lake Louise, Alberta, Canada*.

[35] Julier, S. and J. Uhlmann (1997) "A new extension of the Kalman filter to nonlinear systems," in *Proceedings of SPIE*, vol. 3068, pp. 182–193.

[36] C.L.Bremer and D.T.Kaplan (2001) "Markov chain Monte Carlo estimation of nonlinear dynamics from time series," *Physica D*, **160**, pp. 116–126.

[37] Wang, Y. and L. Geng (2006) "Bayesian network based fault section estimation in power systems," in *IEEE Region 10 Annual International Conference, TENCON*, Hong Kong, China.

[38] Hoffman, A. J. and N. T. van der Merwe (2002) "The application of neural networks to vibrational diagnostics for multiple fault conditions," *Comput. Stand. Interfaces*, **24**(2), pp. 139–149.

[39] David, B. and G. Bastin (2002) "Parameter estimation in nonlinear systems with auto and crosscorrelated noise," *Automatica*, **38**, pp. 81–90.

[40] R.Ghanem and F.Romeo (2001) "A wavelet-based approach for model and parameter identifcation of non-linear systems," *Int. J. Non-linear Mechanics*, **36**, pp. 835–859.

[41] L.Yao and W.A.Sethares (1994) "Nonlinear Parameter Estimation via the Genetic Algorithm," *IEEE Transactions on Signal Processing*, **42**(4), pp. 927–935.

[42] Gupta, S., A. Ray, and E. Keller (2007) "Symbolic time series analysis of ultrasonic data for early detection of fatigue damage," *Mechanical Systems and Signal Processing*, **21**(2), pp. 866–884.

[43] Sarkar, S., M. Yasar, S. Gupta, A. Ray, and K. Mukherjee (May 2008) "Fault Detection and Isolation in Aircraft Gas Turbine Engines: Part II - Validation on a Simulation Test Bed," *Proceedings of the I Mech E Part G: Journal of Aerospace Engineering*, **222**(3), pp. 319–330.

[44] Jazwinski, A. H. (1970) *Stochastic processes and filtering theory*, Academic Press, New York.

[45] Arulampalam, M. S., S. Maskell, N. Gordon, and T. Clapp (2002) "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, **50**(2), pp. 174–188.

[46] C.Andrieu, A.Doucet, S.S.Singh, and V.B.Tadic (2004) "Particle Methods for Change Detection, System Identification, and Control," *Invited Paper, Proceedings of the IEEE*, **92**(3), pp. 423–438.

[47] Li, P. and V. Kadirkamanathan (2001) "Particle filtering based likelihood ratio approach to fault diagnosis in nonlinear stochastic systems," *IEEE Transactions on Systems, Man and Cybernetics*, **31**(3), pp. 337–343.

[48] R.Duda, P.Hart, and D.Stork (2001) "Pattern Classification, 2/e," *John Wiley, New York*, pp. 915–930.

[49] Liu, J. and J. M. A. Scherpen (2002) "Fault detection method for nonlinear systems based on probabilistic neural network filtering," *International Journal of Systems Science*, **33**(13), p. 1039 1050.

[50] Haykin, S. (1999) *Neural Networks: A Comprehensive Foundation*, Prentice-Hall.

[51] Fukunaga, K. (1990) "Statistical Pattern Recognition, 2nd ed." .

[52] Shawe-Taylor, J. (2004) "Kernel Methods for Pattern Analysis," .

[53] Eckmann, J. P. and D. Ruelle (1985) "Ergodic Theory of Chaos and Strange Attractors," *Reviews of Modern Physics*, **57**(3), pp. 617–656.

[54] Cover, T. M. and J. A. Thomas (1991) *Elements of Information Theory*, John Wiley, New York.

[55] Gupta, S., A. Ray, and A. Mukhopadhyay (2006) "Anomaly detection in thermal pulse combustors using symbolic time series analysis," *Proc. IMechE Part I: Journal of Systems and Control Engineering*, **220**(5), pp. 339–351.

[56] Rajagopalan, V., A.Ray, R. Samsi, and J. Mayer (2007) *Pattern identification in dynamical systems via symbolic time series analysis*.

[57] Kantz, H. and T. Schreiber (2004) *Nonlinear Time Series Analysis, 2nd ed*, Cambridge University Press, United Kingdom.

[58] Mallat, S. (1998) *A Wavelet Tour of Signal Processing*, 2 ed., Academic Press, Boston, MA.

[59] A.Ray (2005) "Signed real measure of regular languages for discrete-event supervisory control," *Int. J. Control*, **78**(12), pp. 949–967.

[60] Bapat, R. and T. Raghavan (1997) *Nonnegative Matrices and Applications*, Cambridge University Press.

[61] Wen, Y. and A. Ray (2010) "A stopping rule for symbolic dynamic filtering," *Applied Mathematics Letters*, **23**(9), pp. 1125–1128.

[62] Nrgaard, M., O. Ravn, N. K. Poulsen, and L. K. Hansen (2000) *Neural Networks for Modelling and Control of Dynamic Systems*, Springer-Verlag, London.

[63] Cohen, L. (1995) *Time-Frequency Analysis*, Prentice Hall PTR.

[64] Subbu, A. and A. Ray (2008) "Space Partitioning via Hilbert Transform for Symbolic Time Series Analysis," *Applied Physics Letters*, **92**(8), p. 084107.

[65] Aminian, F. and M. Aminian (2001) *Fault Diagnosis of Nonlinear Analog Circuits Using Neural Networks with Wavelet and Fourier Transforms as Preprocessors*, vol. 17, Kluwer Academic Publishers, Norwell, MA, USA.

[66] Rao, C., A. Ray, S. Sarkar, and M. Yasar (2008) "Review and Comparative Evaluation of Symbolic Dynamic Filtering for Detection of Anomaly Patterns," DOI 10.1007/s11760-008-0061-8.

[67] Rajagopalan, V., S. Chakraborty, and A. Ray (2008) "Estimation of slowly varying parameters in nonlinear systems via symbolic dynamic filtering," *Signal Processing*, **88**(2), pp. 339–348.

[68] Brunk, H. (1995) *An introduction to mathematical statistics,3rd Edn.*, Xerox Publishing,Lexington, MA.

[69] H.D.Brunk (1975) "An Introduction to Mathemeatical Statistics, 3/e," *Xerox College Publishing, MA.*

[70] Pathria, R. (1996) *Statistical Mechanics*, Elsevier Science and Technology Books.

[71] Wilks, S. S. (1962) *Mathematical Statistics*, John Wiley and Sons,, New York, NY.

[72] V.Rajagopalan, R.Samsi, A.Ray, J.Mayer, and C.Lagoa (2004) "A Symbolic Dynamics Approach For Early Detection of Slowly Evolving Faults in NonLinear Systems," *IASTED-CSS, Clearwater, FL.*

[73] Thompson, J. and H. Stewart (1986) *Nonlinear Dynamics and Chaos*, John Wiley, Chichester, United Kingdom.

[74] Diao, Y. and K. Passino (2001) "Stable fault-tolerant adaptive fuzzy/neural control for a turbine engine," .

[75] Adibhatla, S. and K. Johnson (1993) "Evaluation of Nonlinear PSC Algorithm on a Variable Cycle Engine," in *AIAA 29th Joint Propulsion Conference and Exhibit*, Monterey, CA.

[76] Parker, K. I. and T. H. Guo (2002) "Development of a Turbofan Engine Simulation in a Graphical Simulation Environment," in *JANNAF Aero-Propulsion Subcommittee Meeting*, Destin, FL.

[77] Pillay, P. and R. Krishnan (1989) "Modeling, Simulation, and Analysis of Permanent-Magnet Motor Drives, Part I: The Permanent-Magnet Synchronous Motor Drive," *IEEE Transactions on Industry Applications*, **25**(2), pp. 265–273.

[78] Julier, S., J. Uhlmann, and H. F. Durrant-Whyte (2000) "A new method for the nonlinear transformation of means andcovariances in filters and estimators," *IEEE Transactions on Automatic Control*, **45**(3), pp. 477–482.

[79] M.Vidyasagar (1993) *Nonlinear Systems Analysis*.

[80] Hendershot, J. and T. Miller (1996) *Design of Brushless Permanent-Magnet Motors*, Oxford University Press.

[81] Chen, S. and T. A. Lipo (Sep./Oct. 1998) "Bearing currents and shaft voltages of an induction motor under hard- and soft-switching inverter excitation," *IEEE Trans. Ind. Appl*, **34**(5), pp. 1042–1048.

[82] Kobayashi, T. and D. L. Simon (2001) "A hybrid neural network-genetic algorithm technique for aircraft engine performance diagnostics," in *37th Joint Propulsion Conference and Exhibit cosponsored by the AIAA, ASME, SAE, and ASEE*, Salt Lake City, Utah.

[83] Doucet, A., S. Godsill, and C. Andrieu (2000) "On sequential Monte Carlo sampling methods for Bayesian Filtering," *Statistics and Computing*, **10**, pp. 197–208.

[84] Tan, K. K., S. Huang, and T. H. Lee (2006) "Fault detection and diagnosis using neural network design," in *Third International Symposium on Neural Networks, ISNN 2006, Proceedings - Part III*, pp. 364–369.

[85] Patton, R. J. and J. Chen (1996) "Neural networks in fault diagnosis of nonlinear dynamic systems," *Engineering Simulation*, **13**, pp. 905–924.

[86] Sreedhar, R., B. Fernandez, and G. Y. Masada (1995) "Neural network based adaptive fault detection scheme," in *Proceedings of the American Control Conference*, vol. 5, pp. 3259–3263.

[87] Hagan, M., H. Demuth, and M. Beale (1996) *Neural Network Design*, PWS Publishing.

[88] Riedmiller, M. and H. Braun (1993) "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," in *Proceedings of the IEEE International Conference on Neural Networks*.

[89] Kerschen, G. and J.-C. Golinval (2002) "Non-linear generalization of principal component analysis: From a global to a local approach," *Journal of Sound and Vibration*, **254**(5), pp. 867–876.

[90] Jakubek, S. M. and T. Strasser (2004) "Artificial neural networks for fault detection in large-scale data acquisition systems," *Engineering Applications of Artificial Intelligence*, **17**, pp. 233–248.

[91] Abe, S. (2003) "Geometry of escort distributions," *PHYSICAL REVIEW*, **68**(3), pp. 031101–+, `arXiv:cond-mat/0305231`.

Vita

Chinmay Rao

Chinmay Rao was born in Mumbai, India on August 18, 1981. He received his baccalaureate degree in Electronics Engineering from the Vivekanand Education Societies' Institute of Technology, Mumbai in the year 2003. He joined The Pennsylvania State University (UP) in 2003. He joined Prof. Asok Rays research group in 2006 and received two concurrent masters degrees in Electrical and Mechanical Engineering from Penn State. His general research interests include: signal processing, control systems theory and pattern recognition. His specific areas of interest include system identification, sensor selection and fusion and data-driven signal and image processing. He also enjoys reading and several sports such as badminton, squash, running and cycling.