In: Fault Detection: Classification, Techniques and Role in Industrial Systems ISBN xxx
Editor: xxxxx, pp. 1-72 © 2013 Nova Science Publishers, Inc.

Chapter X

DATA-ENABLED HEALTH MANAGEMENT OF COMPLEX INDUSTRIAL SYSTEMS*

***Soumik Sarkar*†, *Soumalya Sarkar*‡ *and Asok Ray*§**
Mechanical Engineering Department, The Pennsylvania State University
University Park, PA 16802, USA

Abstract

Complex industrial systems, such as gas turbine engines, have many heterogeneous subsystems (e.g., thermal, mechanical, hydraulic and electrical) with complex thermal-hydraulic, electro-mechanical, and electronic interactions. Schedule-based policies are largely followed in the industry for maintenance of complex systems. Since the degradation profiles of individual systems are usually different due to manufacturing and usage variations, a scheduled maintenance policy in most cases becomes either overly conservative or a source of serious safety concern. This calls for development of reliable and cost-effective health management for complex industrial systems. However, reliable first-principle modeling of such systems may often be very expensive (e.g., in terms of computational memory, execution time, and cost) and hence may become inappropriate for condition monitoring, fault detection, diagnostics and prognostics.

This chapter describes a data-driven framework to obtain abstract models of complex systems from multiple sensor observations to enable condition monitoring, diagnostics and supervisory control. The algorithms are formulated in the setting of symbolic dynamic filtering (SDF) that has been recently reported in literature. The underlying concept of SDF is built upon the principles of symbolic dynamics, statistical pattern recognition, probabilistic graphical models and information theory. SDF was initially developed for statistically quasi-stationary data and then it was successfully extended to analysis of transient data. The tool-chain involves data space abstraction for analyzing mixed data (i.e., continuous, discrete and categorical) without significant loss of information. Spatiotemporal pattern discovery & fusion algorithms have been built upon Markovian & Bayesian network concepts and health status classifications

*This work has been supported in part by the U.S. Army Research laboratory and the U.S. Army Research Office under Grant No. W911NF-07-1-0376 and by NASA under Cooperative Agreement No. NNX07AK49A.

†Currently with United Technologies Research Center, E-mail address: sarkars@utrc.utc.com
‡E-mail address: soumalya.still.dreaming@gmail.com
§E-mail address: axr2@psu.edu

by using machine learning techniques. The chapter also presents applications of the generic data-driven tool in heterogeneous areas, such as fault detection in aircraft gas turbine engines, fatigue damage prediction in structural materials, and health monitoring in nuclear power plants & ship-board auxiliary systems.

1. Introduction

With the advent of advanced sensing and computational resources, opportunities for data-driven technologies have become abundant in most of the industries dealing with complex human-engineered systems. Need for high-fidelity condition monitoring, fault detection, diagnostics and prognostics also grew due to the increasing inter-connectedness of such electro-mechanical systems. Apart from the physical components, their cyber counterparts added another dimension from the perspective of complexity, uncertainty and disturbance. Although model-based approaches have shown their merits to utilize the understanding of physical phenomena to detect and isolate faults in industrial systems, data-driven methods are receiving increased attention to address the underlying issues, such as scalability, robustness and cost of deployment. To this end, this chapter summarizes the decade-long development of a statistical time-series analysis tool called the Symbolic Dynamic Filtering (SDF) that has been shown to be extremely effective for numerous industrial systems, some of which will be discussed here.

The chapter is organized in six sections including the present one which briefly introduces the notion of nonlinear time series analysis and presents the two-time-scale problem formulation for anomaly detection using symbolic dynamic filtering. Section 1.1.2. provides a brief overview of symbolic dynamics and encoding of time series data. Before describing different aspects of the SDF tool-chain, Section 2. introduces the industrial applications that will be used to illustrate the capabilities and performance of the tool. This applications include Gas Turbine engines, Fatigue damage problems, Nuclear power plants and Ship-board auxiliary systems. The first step of the tool-chain is data space abstraction where raw time-series can be transformed into a more suitable domain via wavelet or Hilbert transform. After the pre-processing step time-series data is spatially abstracted via a discretization process known as partitioning. Details and varieties of the abstraction procedure are presented in Section 3.. While the standard steady-state symbolic modeling is discussed at length in 1.1., a recent development to encode short-length time-series motifs after partitioning is discussed in Section 4.. This section also highlights the classification scheme of such motifs. Today's industrial systems almost always are monitored by multiple sensors of heterogeneous modality and it is getting increasingly obvious to design health management systems with information fusion capability. However, most of the state-of-the art fusion technologies deployed today fuse information at the decision layer after extracting condition indicators from individual sensors. This strategy suffers from acute loss of information that may prevent early detection and prognostics. To this end, SDF has a unique advantage especially due to the data abstraction step in the ability of seamless fusion of multi-sensor information at the data and feature layers and Section 5. presents this capability in detail. Finally, Section 6. summarizes and concludes the chapter with recommendations for future research directions.

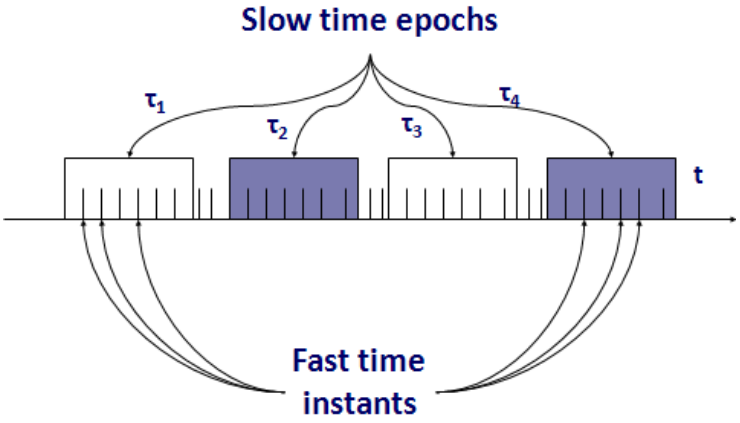


Figure 1. Pictorial view of the two time scales: (i) *Slow time scale* of system evolution and (ii) *Fast time scale* for data acquisition and signal conditioning

1.1. General Framework of Symbolic Dynamic Filtering (SDF)

Symbolic Dynamic Filtering (SDF) is built upon the concepts from multiple disciplines including Statistical Mechanics [1, 2], Symbolic Dynamics [3], Statistical Pattern Recognition [4], and Information Theory [5]. It is shown to be a fast, efficient and computationally inexpensive pattern recognition tool. These qualities are particularly important requirements for autonomous cyber-physical system applications. It has been shown by laboratory experimentation [6] that this method of pattern identification yields superior performance in terms of early detection of anomalies and robustness to measurement noise in comparison with other existing techniques such as Principal Component Analysis, (PCA) and Artificial Neural Networks (ANN).

1.1.1. A Brief Overview

The Symbolic Dynamic Filtering (SDF) technique is built upon the concept of two time-scales while analyzing time-series data from a given dynamical system. The *fast time scale* is related to the response time of the system dynamics. Over the span of a given time series data sequence, the dynamic behavior of the system is assumed to remain invariant, i.e., the process is quasi-stationary on the fast time scale. On the other hand, the *slow time scale* is related to the time span over which the critical parameters of the system may change and exhibit non-stationary dynamics. The concept of two time scales is illustrated in Fig. 1, where a long time span in the fast scale could be a tiny time interval in the slow scale.

1.1.2. Symbolic Dynamics Encoding, and State Machine

The basic concepts of *Symbolic Dynamics* include:

1. Encoding (possibly nonlinear) system dynamics from observed time series data for generation of symbol sequences.

2. Construction of probabilistic finite state automata ($PFSA$) from symbol sequences for generation of pattern vectors for representation of the environment's dynamical characteristics.

The continuously-varying finite-dimensional model of a dynamical system is usually formulated in the setting of an initial value problem as:

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \theta(t_s));\ \mathbf{x}(0) = \mathbf{x}_0, \tag{1}$$

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. Let $\Omega \subset \mathbb{R}^n$ be a compact (i.e., closed and bounded) region, within which the trajectory of the dynamical system, governed by Eq. (1), is circumscribed. 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: $\mathbb{O} \equiv \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \cdots\}$, where each $\mathbf{x}_i \in \Omega$, which passes through or touches one of the cells of the partition. Each initial state $\mathbf{x}_0 \in \Omega$ generates a sequence of symbols defined by a mapping from the phase space into the symbol space as:

$$\mathbf{x}_0 \rightarrow \sigma_{i_0}\sigma_{i_1}\sigma_{i_2}\ldots \tag{2}$$

where each $\sigma_{i_k},\ k = 0, 1, \cdots$ takes a symbol from the alphabet Σ. The mapping in Eq. (2) is called *Symbolic Dynamics* as it attributes a legal (i.e., physically admissible) symbol sequence to the system dynamics starting from an initial state. The partition is called a generating partition of the phase space Ω if every legal (i.e., physically admissible) symbol sequence uniquely determines a specific initial condition $\mathbf{x}_0$. In other words, every (semi-infinite) symbol sequence uniquely identifies one continuous space orbit [7].

In physics and signal processing literature, similar procedure is known as coarse-graining or quantization. However, in both cases, typically very fine and uniform quantization cells are considered to reduce loss of information. On the other hand, literature of dynamical systems in mathematics presents the concept of partitioning in the context of symbolic dynamics. Unlike coarse-graining or quantization, partitioning typically involves nonuniform and much coarser representation of the phase space, which entails a low order modeling of complex dynamical systems. Moreover, in case of a generating partition, there is no loss of information. Although symbolic dynamics is subjected to (possible) loss of information resulting from granular imprecision of partitioning boxes, the essential robust features (e.g., periodicity and chaotic behavior of an orbit) are expected to be preserved in the symbol sequences through an appropriate partitioning of the phase space [8].

Thus partitioning a finite region of the phase space leads to mapping from the partitioned space into the symbol alphabet [9]. This represents a spatial and temporal discretization of the system dynamics defined by the trajectories. Although the theory of phase-space partitioning is well developed for one-dimensional mappings [7], very few results are known for two and higher dimensional systems. Furthermore, the state trajectory of the system variables may be unknown in case of systems for which a model as in Eq. (1) is not known or is difficult to obtain. As such, as an alternative, the time series data set of

selected observable outputs can be used for partitioning and symbolic dynamic encoding. In general, the time series data can be generated from the available sensors and/or from analytically derived model variables. After partitioning, the symbol sequence is converted into a probabilistic finite-state machine, that is considered as the compressed abstract model of the system.

1.1.3. Phase Space Partitioning

Several partitioning techniques have been reported in literature for symbol generation [10][11], primarily based on symbolic false nearest neighbors ($SFNN$). These techniques rely on partitioning the phase space and 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 or Hilbert transforms largely alleviate the difficulties of phase-space partitioning and are particularly effective with noisy data from high-dimensional dynamical systems [12, 13]. A further theoretical development has been made to generalize Hilbert transform [14] that might prove to be a better choice as a pre-processing technique.

In both of these approaches, time series data are first converted to wavelet domain or analytic signal domain, and the transformed space is then partitioned with alphabet size $|\Sigma|$ into segments. 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 [12]. The partitioning can be done such that the regions are partitioned uniformly (equal width) or the regions with more information are partitioned finer and those with sparse information are partitioned coarser. This is achieved by maximizing the Shannon entropy [5], which is defined as:

$$S = -\sum_{i=1}^{|\Sigma|} p_i \, log(p_i) \tag{3}$$

where p_i is the probability of a data point to be in the i^{th} partition segment. In this case, the size of the cells is smaller for regions with higher density of data points to ensures an unbiased partition such that each cell is allocated equal number of visits at the nominal condition. Uniform probability distribution, i.e., $p_i = \frac{1}{|\Sigma|}$ for $i = 1, 2, \ldots |\Sigma|$, is a consequence of maximum entropy partitioning [12]. In the illustrative example of Fig. 2, the partitioning contains 4 cells (i.e., line intervals in this case).

Apart from this maximum entropy (equal frequency) or uniform (equal width) partitioning schemes, partitioning can be optimized based on other objective functions. This will be discussed in details later. Once the partitioning is done with alphabet size $|\Sigma|$ at a reference slow time scale condition, it is kept constant for all slow time epochs, i.e., the structure of the partition is fixed at the reference condition. Therefore, the partitioning structure generated at the reference condition serve as the reference frame for computation of pattern vectors from time series data at subsequent slow time epochs.

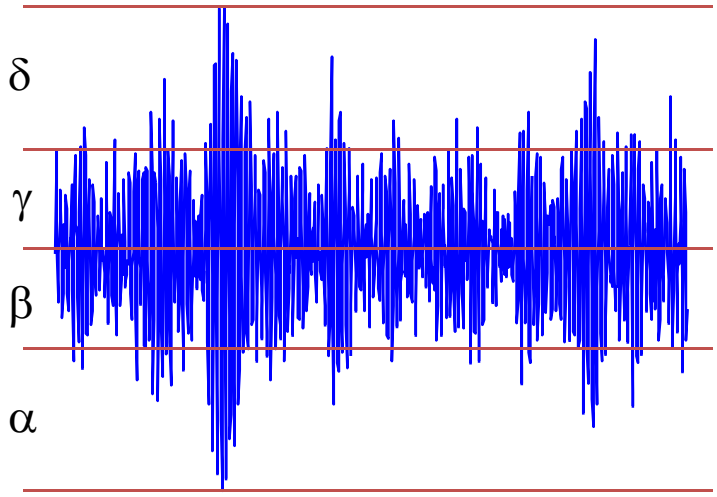


Figure 2. An Example of Partitioning

1.1.4. Probabilistic Finite State Automata ($PFSA$) Construction

Once the symbol sequence is obtained, the next step is the construction of a Probabilistic Finite State Automaton ($PFSA$) (as shown in Fig. 3) and calculation of the respective state probability vector. The partitioning (see Fig. 2) is performed at the reference slow time epoch. A $PFSA$ is then constructed, where the states of the machine are defined corresponding to a given *alphabet* set Σ and window length D. This explicit state construction method helps defining abstract states for systems without considerable domain knowledge. The alphabet size $|\Sigma|$ is the total number of partition segments while the window length D is the length of consecutive symbol words [15], 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, 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.

The choice of $|\Sigma|$ and D depends on specific applications and the noise level in the time series data as well as on the available computation power and memory availability. As stated earlier, 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 and increased length of the data sets. For the analysis of data sets in this research, the window length is set to D=1; consequently, the set of states Q is equivalent to the

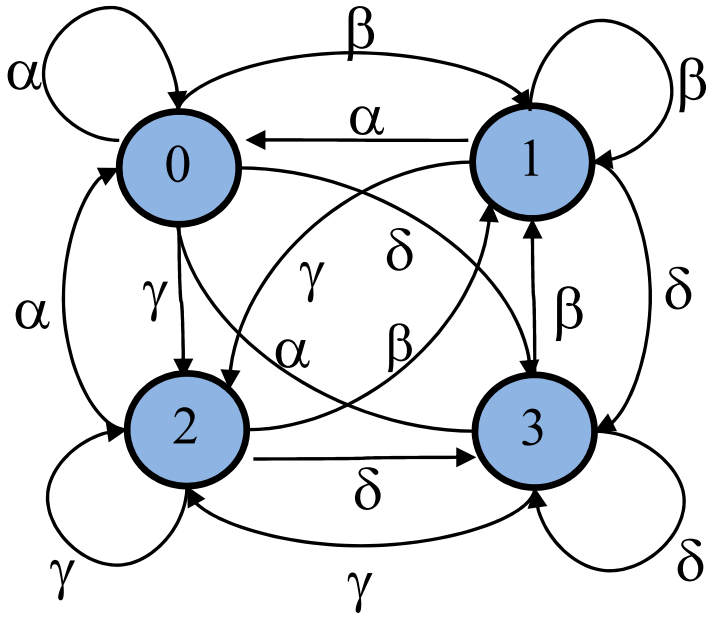


Figure 3. Example of Finite State Automaton

symbol alphabet Σ. With the selection of the parameters D=1 and $|\Sigma|$=8, the $PFSA$ has only 8 states, which yields very fast computation and low memory requirements; hence, the algorithm is capable of early detection of anomalies and incipient faults. However, other applications, such as two-dimensional image processing, may require larger values of the parameter D and hence possibly larger number of states in the $PFSA$.

Using the symbol sequence generated from the time series data, the state machine is constructed on the principle of sliding block codes [3]. 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 $PFSA$ constructed in this fashion is called the D-Markov machine [15], because of its Markov properties.

Definition 1..1. *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 other slow time epochs. That is, the partitioning and the state machine structure generated at the reference 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.

Definition 1..2. *The probability of transitions from state q_j to state q_k belonging to the set Q of states under a transition $\delta : Q \times \Sigma \to Q$ is defined as*

$$\pi_{jk} = P\left(\sigma \in \Sigma \mid \delta(q_j, \sigma) \to q_k\right); \sum_k \pi_{jk} = 1; \tag{4}$$

Thus, for a D-Markov machine, the irreducible stochastic matrix $\mathbf{\Pi} \equiv [\pi_{ij}]$ describes all transition probabilities between states such that it has at most $|\Sigma|^{D+1}$ nonzero entries. The left eigenvector $\mathbf{p}$ corresponding to the unit eigenvalue of $\mathbf{\Pi}$ is the state probability vector under the (fast time scale) stationary condition of the dynamical system [15].

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:

$$\begin{aligned} \pi_{jk} \equiv P(q_k|q_j) = \frac{P(q_k, q_j)}{P(q_j)} &= \frac{P(\sigma_{i_1}\cdots\sigma_{i_D}\sigma)}{P(\sigma_{i_1}\cdots\sigma_{i_D})} \\ \Rightarrow \pi_{jk} &\approx \frac{N(\sigma_{i_1}\cdots\sigma_{i_D}\sigma_)}{N(\sigma_{i_1}\cdots\sigma_{i_D})} \end{aligned} \tag{5}$$

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 time series data at the reference condition, set as a benchmark, generates the *state transition matrix* $\mathbf{\Pi}$ that, in turn, is used to obtain the *state probability vector* $\mathbf{q}$ whose elements are the stationary probabilities of the state vector, where $\mathbf{q}$ is the left eigenvector of $\mathbf{\Pi}$ corresponding to the (unique) unit eigenvalue. The state probability vector $\mathbf{p}$ is obtained from time series data at a (possibly) faulty condition. The partitioning of time series data and the state machine structure should be the same in both cases but the respective state transition matrices could be different.

Pattern changes may take place in dynamical systems over slow time epochs due to various reasons. The pattern changes are quantified as deviations from the reference pattern (i.e., the probability distribution at the reference condition). To identify variation of a single parameter in the dynamical system, the resulting anomalies (i.e., deviations of the evolving patterns from the reference pattern) are characterized by a scalar-valued function, called *anomaly measure* μ. The anomaly measures are obtained as:

$$\mu \equiv d\left(\mathbf{p}, \mathbf{q}\right) \tag{6}$$

where the $d(\bullet, \bullet)$ is an appropriately defined distance function. An extension of this method for identification of multiple parameters in dynamical systems has also been made in [16]. Instead of using the low dimensional state probability vector as a pattern, the entire Perron-Frobenius operator (state transition matrix) may be used as the pattern to reduce loss of information.

1.1.5. Stopping Rule for Determining Symbol Sequence Length

A stopping rule 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 [17] is based on the properties of irreducible stochastic matrices [18]. The state transition matrix, constructed at the r^{th} iteration (i.e., from a symbol sequence of length r), is denoted as $\mathbf{\Pi}(r)$ 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

$$p_i(r) = \frac{r_i}{\sum_{j=1}^{n} r_i} \tag{7}$$

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 [18] to establish a relation between the vector $\mathbf{p}(r)$ and the matrix $\mathbf{\Pi}(r)$. Since the matrix $\mathbf{\Pi}(r)$ 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, under the hypothetical arbitrarily long sequences, the following condition is assumed to hold.

$$\mathbf{p}(r+1) = \mathbf{p}(r)\mathbf{\Pi}(r) \ \ \Rightarrow \ \ \mathbf{p}(r) = \mathbf{p}(r)\mathbf{\Pi}(r) \ \text{ as } r \to \infty \tag{8}$$

Following Eq. (7), the absolute error between successive iterations is obtained such that

$$\| \ (\mathbf{p}(r) - \mathbf{p}(r+1)) \ \|_\infty = \| \ \mathbf{p}(r) \ (\mathbf{I} - \mathbf{\Pi}(r)) \ \|_\infty \leq \frac{1}{r} \tag{9}$$

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:

$$\frac{\| \ (\mathbf{p}(r) \ - \ \mathbf{p}(r+1)) \ \|_\infty}{\| \ (\mathbf{p}(r)) \ \|_\infty} \leq \eta \ \ \forall \ r \geq r_{stop} \tag{10}$$

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. (9) and (10) as:

$$r_{stop} \equiv int\left(\frac{n}{\eta}\right) \tag{11}$$

where $int(\bullet)$ is the integer part of the real number $\bullet$.

More advanced forms of stopping rules are reported by Wen and Ray [19, 20], where the parameter η in Eq. 10 is statistically determined.

2. Overview of Industrial System Testbeds

This section briefly introduces the industrial applications that will be used in the sequel to validate the effectiveness of the SDF tool. The applications include aircraft gas turbine engines, fatigue damage problems, nuclear power plants and ship-board auxiliary systems.

2.1. Aircraft Gas Turbine Engines

This section presents the C-MAPSS test bed along with the fault injection scheme. The C-MAPSS simulation test bed [21] was developed at NASA for a typical commercial-scale two-spool turbofan engine and its control system. Figure 4 shows the schematic diagram of a commercial aircraft gas turbine engine used in the C-MAPSS simulation test bed. The

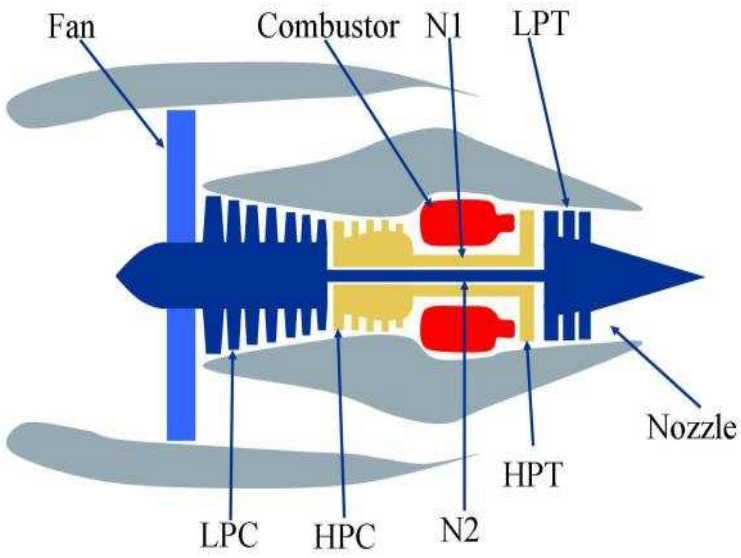


Figure 4. Gas turbine engine schematic [21]

engine under consideration produces a thrust of approximately 400,000 N and is designed for operation at altitude (A) from the sea level (i.e., 0 m) up to 12,200 m, Mach number (M) from 0 to 0.90, and temperatures from approximately $-50°$C to $50°$C. The throttle resolving angle (TRA) can be set to any value in the range between $0°$ at the minimum power level and $100°$ at the maximum power level. The gas turbine engine system consists of five major rotating components, namely, fan (F), low pressure compressor (LPC), high pressure compressor (HPC), high pressure turbine (HPT), and low pressure turbine (LPT), as seen in Figure 4. Apart from the rotating components, three actuators are modeled in the simulation test bed, namely, Variable Stator Vane (VSV), Variable Bleed Valve (VBV), and Fuel Pump that controls the fuel flow rate (Wf).

Given the inputs of TRA, A and M, the interactively controlled component models in the simulation test bed compute nonlinear dynamics of real-time turbofan engine operation. A gain-scheduled control system is incorporated in the engine system, which consists of speed controllers and limit regulators for engine components. Out of the different types of sensors (e.g., pressure, temperature, and shaft speed) used in the C-MAPSS simulation test bed, Table 1 lists those sensors that are commonly adopted in the Instrumentation & Control system of commercial aircraft engines, as seen in Figure 5.

In the current configuration of the C-MAPSS simulation test bed, there are 13 component level health parameter inputs, namely, efficiency parameters (ψ), flow parameters (ζ)

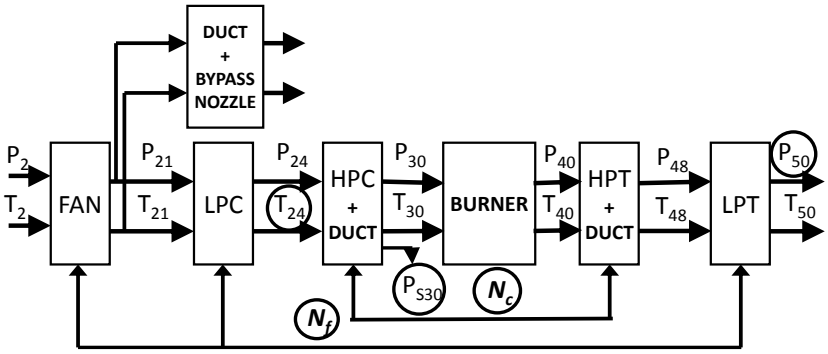


Figure 5. Schematic diagram of the C-MAPSS engine model with Sensors

Table 1. Sensor Suite for the Engine System

Sensors	**Description**
T_{24}	LPC exit/ HPC inlet temperature
Ps_{30}	HPC exit static pressure
T_{48}	HPT exit temperature
P_{50}	LPT exit pressure
N_f	Fan spool speed
N_c	Core spool speed

and pressure ratio modifiers, that simulate 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) that are defined as:

- $\eta \triangleq$ Ratio of actual enthalpy and ideal enthalpy changes.
- $\phi \triangleq$ Ratio of rotor tip and axial fluid flow velocities.

For the engine's five rotating components F, LPC, HPC, LPT, and HPT, the ten respective efficiency and flow health parameters are: (ψ_F, ζ_F), (ψ_{LPC}, ζ_{LPC}), (ψ_{HPC}, ζ_{HPC}), ((ψ_{HPT}, ζ_{HPT}), and (ψ_{LPT}, ζ_{LPT}). An engine component C is considered to be in nominal condition if both ψ_C and ζ_C are equal to 1 and fault can be injected in the component C by reducing the values of ψ_C and/or ζ_C. For example, $\psi_{HPC} = 0.98$ signifies a 2% relative loss in efficiency of HPC. Actuator faults can also be injected through the scale shift parameters for the three actuators, VSV, VBV and Wf.

2.2. Fatigue Damage in Polycrystalline Alloys

Fatigue damage is one of the most commonly encountered sources of structural degradation of mechanical structures, made of polycrystalline alloys. Therefore, analytical tools for online fatigue damage detection are critical for a wide range of engineering applications. The process of fatigue damage is broadly classified into two phases: crack initiation and

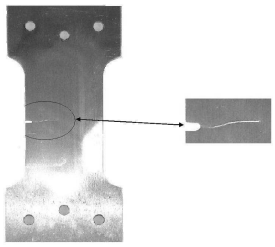


Figure 6. Cracked specimen with a side notch

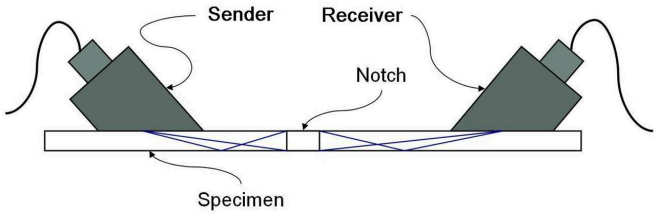


Figure 7. Ultrasonic flaw detection scheme

crack propagation. The damage mechanism of these two phases are significantly different and similar feature extraction policies may not work effectively to classify different damage levels in these two phases. For example, damage evolution in the crack initiation phase is much slower, resulting in smaller change in ultrasonic signals as compared to that in the crack propagation phase. The phase transition from crack initiation to crack propagation occurs when several small micro-cracks coalesce together to develop a single large crack that propagates under the oscillating load. Several crack propagation models have been developed based on the inherent stochastic nature of fatigue damage evolution for prediction of the remaining useful life. Due to stochastic nature of material microstuctures and operating conditions, a physics-based model would require the knowledge of the parameters associated with the material and geometry of the component. These parameters are often randomly varying and may not be accurately predicted a priori. Crack propagation rate is a function of crack length and, after a certain crack length, it becomes unstable.

Experimental Apparatus and Test Procedure: An experimental apparatus is designed to study the fatigue damage growth in mechanical structures. The apparatus consists of an MTS 831.10 Elastomer Test System that is integrated with Olympus BX Series microscope with a long working-distance objective. A camera, mounted on the microscope, takes images with a resolution of 2 micron per pixel at a distance of 20mm. The ultrasonic sensing device is triggered at a frequency of 5 MHz at each peak of the fluctuating load. Various components of the apparatus communicate over a TCP/IP network and post sensor, microscope and fatigue test data on the network in real time. This data can used by analysis algorithms for anomaly detection and health monitoring of the specimens in real-time.

Figure 6 shows a side-notched 7075-T6 aluminum alloy specimen used in the fatigue damage test apparatus. Each specimen is $3\ mm$ thick and $50\ mm$ wide, and has a slot of $1.58\ mm \times 4.57\ mm$ on one edge. The notch is made to increase the stress concentration factor that localizes crack initiation and propagation under the fluctuating load. Fatigue tests were conducted at a constant amplitude sinusoidal load for low-cycle fatigue, where the maximum and minimum loads were kept constant at 87MPa and 4.85MPa, respectively.

For low cycle fatigue studied in this study, the stress amplitude at the crack tip is sufficiently high to observe the elasto-plastic behavior in the specimens under cyclic loading. A significant amount of internal damage caused by multiple small cracks, dislocations and microstructural defects alters the ultrasonic impedance, which results in signal distortion and attenuation at the receiver end.

The optical images were collected automatically at every 200 cycles by the optical microscope which is always focussed in the crack tip. As soon as crack is visible by the microscope, crack length is noted down after every 200 cycles. Ultrasonic waves with a frequency of 5 MHz were triggered at each peak of the sinusoidal load to generate data points in each cycle. Since the ultrasonic frequency is much higher than the load frequency, data acquisition was done for a very short interval in the time scale of load cycling. Therefore, it can be implied that ultrasonic data were collected at the peak of each sinusoidal load cycle, where the stress is maximum and the crack is open causing maximum attenuation of the ultrasonic waves. The slow time epochs for data analysis were chosen to be 1000 load cycles (i.e., $\sim$80 sec) apart. To generate training and test data sample multiple experiments are conducted on different specimen. For each specimen all ultrasonic signals are labeled with crack length.

2.3. Nuclear Power Plants

The International Reactor Innovative & Secure (IRIS) simulator of nuclear power plants is based on the design of a next-generation nuclear reactor. It is a modular pressurized water reactor (PWR) with an integral configuration of all primary system components. Figure 8 shows the layout of the primary side of the IRIS system that is offered in configurations of single or multiple modules, each having a power rating of $1000MWt$ (about $335MWe$) [22]. The nominal reactor core inlet and outlet temperatures are $557.6^{\circ}F$ ($292^{\circ}C$) and $626^{\circ}F$ ($330^{\circ}C$), respectively. The pressurizer, eight steam generators, and the control rod mechanism are integrated into the pressure vessel with the reactor core. There is no huge pipe used to connect these components. This design avoids the large loss of coolant accident (LOCA). The entire control rod mechanism is mounted inside the pressure vessel to avoid the problem of penetrating the the pressure vessel head by the control rod. The test-bed is built using FORTRAN programming language. This FORTRAN model includes a reactor core model, a helical coil steam generator (HCSG) model.

The test-bed is implemented on a Quad Core 2.83 GHz CPU 8 GB RAM Workstation in the laboratory of Penn State. The IRIS simulator is operated in the *integrated control* mode through built-in PID controllers, which operates all three subsystems (i.e., turbine, feedwater flow and control rods) to combine the rapid response of a reactor-following system with the stability of a turbine-following system [23]. For a load increase in the turbine, the control system makes the control rods to be withdrawn and the feedwater flow rate to be increased. Simultaneously, the control system temporarily reduces the turbine header pressure set point. The turbine governor valves open to maintain the turbine speed and to reduce the main steam header pressure to the new set point. As reactor core power, primary to secondary heat transfer, and steam flow rate gradually increase, the turbine header pressure recovers and its set point is returned to the steady-state value. In this way, the control system achieves a rapid response and a smooth transition between turbine loads [23].

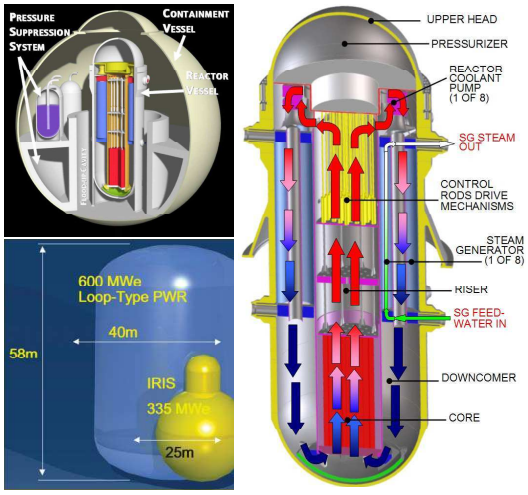


Figure 8. Layout of the primary side of the IRIS system [22]

The IRIS test-bed is capable of simulating both normal conditions at different operational modes (e.g., normal power maneuvers, reactor start-up, and turbine loading) and a variety of anomalous scenarios that include:

- Actuator anomalies, e.g., feedwater pump trip, malfunctions of reactor coolant pump and control rod mechanism;
- Sensor failures, e.g., malfunctions of temperature, pressure, and flow-rate sensors;
- Internal anomalies, e.g., uncertainties in the fuel temperature coefficient of reactivity, coolant heat capacity, and feedwater heat capacity.

In the IRIS test-bed, sensor degradations are realized as injected noise and disturbances. Depending on the location and modality of a sensor, there could be several different degradation levels. For example, the degradation levels in a pressure sensor have different characteristics from those of a temperature sensor. Furthermore, depending on the location and operating environment, even sensors of the same modality could have different degradation characteristics. In general, sensor degradation is categorized as the following [24, 25]:

- Constant bias and drift (i.e., slowly-varying bias);
- Change in sensor response time due to aging; and
- Change in the variance of sensor noise (e.g., due to large external electromagnetic radiation from electric motors).

Amongst the above sensor degradation types, only sensor degradation due to changes in the noise variance are investigated in this study. The rationale is that the sensors are assumed to be periodically tested and calibrated; hence, sensor degradation due to aging, bias, and drift is much less likely.

In a PWR plant, the primary coolant and feedwater temperatures are measured using resistance thermometer detectors (RTDs), and the temperature of the outlet water from the

reactor core is measured using thermocouples (TCs). The TCs are mainly used for temperature monitoring, whereas the primary coolant RTDs typically feed the plant's control and safety system [25]. A case study has been presented in this work to validate the anomaly detection methodology, in which the reactor coolant pump (RCP) is chosen to be the location of a component-level degradation and the primary coolant temperature (T_{HL}) sensor (RTD) in the hot leg piping is chosen for anomaly detection in the RCP.

Since the plant controller receives feedback signals from the T_{HL} sensor, any degradation in this sensor could pervade through the plant, which will potentially affect the outputs of the remaining components due to the inherent electro-mechanical and control feedback. Component-level anomaly detection under different sensor noise variance is posed as a multi-class classification problem in the sequel.

2.4. Navy Ship-board Auxiliary Unit

A simulation test bed of shipboard auxiliary systems has been developed for testing the proposed algorithm of sensor fusion in distributed physical processes. The test bed is built upon the model of a notational hydraulic system that is coupled with a notional electrical system under an adjustable thermal loading.

The simulation model is implemented in MATLAB/Simulink as seen in Fig. 9. This distributed notional system is driven by an external speed command ω_{ref} that serves as a set point for the speed, ω_e, of the permanent magnet synchronous motor (PMSM). A mechanical shaft coupling connects the fixed displacement pump (FDP) to the PMSM. The torque load of the PMSM, T_m, is obtained from the shaft model of the hydraulic system. In turn, the speed of the PMSM, ω_e, is an input to determine the angular speed of the shaft, ω_s, which drives the FDP and the cooling fan in the thermal system. In turn, the FDP drives the hydraulic motor (HM) with a dynamic load, which consists of the thermal load, T_t, and a time-varying mechanical torque load. The proportional-integral (PI) controller regulates the

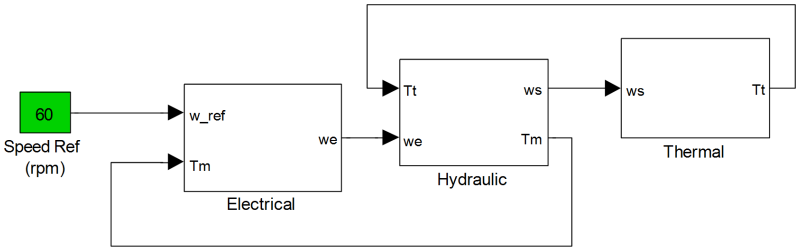


Figure 9. Notional coupled electrical, hydraulic and thermal systems.

PMSMs electrical frequency under dynamic loading conditions that arise due to fluctuations in hydraulic and thermal loading. There is a mechanical coupling between the PMSM and the FDP of the hydraulic system, which is modeled by a rotational spring-damper system applied to an inertial load. The mechanical system outputs the shaft velocity that drives the FDP and the cooling fan of the thermal system. The pump, in turn, drives the HM through the pipeline. The HM is subjected to a time-varying load with a profile defined by the user as well as the thermal load that varies with the fan efficiency of the cooling mechanism. The systems are further coupled with a feedback loop since the torque requirement of the HM

is input to the PMSM of the electrical system. The model has multiple parameters that can simulate various fault conditions. There are multiple sensors in each system with different modalities such as Hall Effect sensors, torque, speed, current, temperature and hydraulic pressure sensors that are explained further in Section 5.2..

The governing equations of the electrical component model are as follows:

$$\begin{aligned}
\frac{i_q}{dt} &= \frac{v_q - Ri_q - wL_d i_d - w\lambda_f}{L_q} \\
\frac{i_d}{dt} &= \frac{v_d - Ri_d + wL_q i_q}{L_d} \\
\frac{w_r}{dt} &= \frac{T_e - T_l - Bw_r}{J} \\
T_e &= \frac{3}{2}P[\lambda_f i_q + (L_d - L_q) i_d i_q] \\
w_r &= \frac{w}{P}
\end{aligned}$$

where subscripts d and q have their usual significance of direct and quadrature axes in the equivalent 2-pole representation; v, i, and L are the corresponding axis voltages, stator currents and inductances; R and ω are the stator resistance and inverter frequency, respectively; λ_f is the flux linkage of the rotor magnets with the stator; P is the number of pole pairs; T_e is the generated electromagnetic torque; 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 governing equations of the fixed displacement pump (FDP) model are as follows:

$$\begin{aligned}
q_p &= D_p \omega_p - k_{leak} P_p \\
T_p &= \frac{D_p P_p}{\eta_m} \\
k_{leak} &= \frac{k_{HP}}{\nu \rho} \\
k_{HP} &= \frac{D_p \omega_{nom}(1 - \eta_v)\nu_{nom}\rho}{P_{nom}}
\end{aligned}$$

The governing equations of the hydraulic motor (HM) model are as follows:

$$\begin{aligned}
\omega_m &= \frac{q_m - k_{leak} P_m}{D_m} \\
T_m &= D_m P_m \eta_m \\
k_{leak} &= \frac{k_{HP}}{\nu \rho} \\
k_{HP} &= \frac{D_m w_{nom}(1 - \eta_v)\nu_{nom}\rho}{P_{nom}}
\end{aligned} \tag{12}$$

where the subscripts p and m denote pump and motor parameters, respectively; the subscript nom denotes nominal values; q and P is the pressure differentials across delivery and terminal points; T is the shaft torque; D is the displacement; ω is the angular velocity; k_{leak} is the flow leakage coefficient; k_{HP} is the Hagen-Poiseuille coefficient; ν_v and ν_m are the volumetric and mechanical efficiencies, respectively; and ν and ρ are the fluid kinematic viscosity and density, respectively.

3. Data Space Abstraction

The first step of the SDF tool-chain involves abstraction of the data space that gives it a unique capability to fuse information from heterogeneous sensors. This also enables integration of mixed i.e., continuous, discrete and categorical data. While spatial partitioning is the major step of the abstraction process, often suitable domain transformation facilitates better feature extraction.

3.1. Data Preprocessing

Recently, it has been shown that Wavelet or Hilbert transformation of data before partitioning improves time-series feature extraction as both time and frequency domain information are considered. The wavelet space partitioning *WSP* is particulary effective for noisy data from high-dimensional dynamical systems. However, WSP has several other shortcomings such as identification of an appropriate basis function, selection of appropriate scales, and non-unique and lossy conversion of the two-dimensional scale-shift wavelet domain to a one-dimensional domain of scale-series sequences [12]. Subbu and Ray [13] have reported use of Hilbert transform to alleviate these problems. This method is called the analytic signal space partitioning (*ASSP*) as Hilbert transform converts data from time domain to analytic signal space. However, for noisy systems, it is expected that number of machine states typically increases largely, which in turn degrades computation efficiency (e.g., increased execution time and memory requirements) [6].

This section reviews a generalization of the classical Hilbert transform to modify *ASSP* for application to noisy systems. The objective here is to partition the transformed signal space such that *D*-Markov machines can be constructed with a small *D* without significant loss of information for noisy signals. The key idea is to provide a mathematical structure of the generalized Hilbert transform such that the low-frequency region is more heavily weighted than that in the classical Hilbert transform.

3.1.1. Generalization of Hilbert Transform

Hilbert transform and the associated concept of analytic signals, introduced by Gabor [26], have been widely adopted for time-frequency analysis in diverse applications of signal processing. Hilbert transform [27] of a real-valued signal $x(t)$ is defined as:

$$\widetilde{x}(t) \triangleq \mathcal{H}[x](t) = \frac{1}{\pi} \int_{\mathbb{R}} \frac{x(\tau)}{t-\tau} d\tau \tag{13}$$

That is, $\widetilde{x}(t)$ is the convolution of $x(t)$ with $\frac{1}{\pi t}$ over $\mathbb{R} \triangleq (-\infty, \infty)$, which is represented in the Fourier domain as:

$$\widehat{\widetilde{x}}(\omega) = -i \; sgn(\omega) \; \widehat{x}(\omega) \tag{14}$$

where $\widehat{x}(\omega) \triangleq \mathcal{F}[x](\omega)$ and $sgn(\omega) \triangleq \begin{cases} +1 & \text{if } \omega > 0 \\ -1 & \text{if } \omega < 0 \end{cases}$

Given the Hilbert transform of a real-valued signal $x(t)$, the complex-valued analytic signal [27] is defined as:

$$\mathcal{X}(t) \triangleq x(t) + i\,\widetilde{x}(t) \tag{15}$$

and the (real-valued) transfer function with input $\widehat{x}(\omega)$ and output $\widehat{\mathcal{X}}(\omega)$ is formulated as:

$$G(\omega) \triangleq \frac{\widehat{\mathcal{X}}(\omega)}{\widehat{x}(\omega)} = 1 + sgn(\omega) \tag{16}$$

Lohmann et al. [28] introduced the concept of a generalized Hilbert transform in the fractional Fourier space instead of the conventional Fourier space; a discrete version of this generalized Hilbert transform was developed later [29]. For geophysical applications, Luo et al. [30] proposed another type of generalized Hilbert transform that is essentially the windowed version of traditional Hilbert transform. However, the notion of generalization of Hilbert transform presented in the sequel is different from that in the previously published literature.

Let us define a generalized Hilbert transform as: $\mathcal{H}^\alpha$ of a real-valued signal $x(t)$ as the convolution:

$$\widetilde{x}^\alpha(t) \triangleq \mathcal{H}^\alpha[x](t) = x(t) * \left(\frac{sgn(t)}{\pi\ |t|^\alpha}\right) \text{ for } \alpha \in (0, 1] \tag{17}$$

It is shown in the sequel that, as $\alpha \uparrow 1$ (i.e., the values of α form an increasing sequence of positive real numbers with the limit equal to 1), $\mathcal{H}^\alpha$ converges to $\mathcal{H}$, where $\mathcal{H}$ is the classical Hilbert transform defined in Eq. (13); that is, $\mathcal{H}^1 \equiv \mathcal{H}$.

Two lemmas are presented, which are necessary for derivation of the main results in the Fourier space (detailed proofs can be found in [14]).

Lemma 3..1.

$$\int_{-\infty}^{\infty} \frac{e^{-i\omega t}}{\pi|t|^\alpha} sgn(t)dt = -i\ sgn(\omega)\frac{2}{\pi}\frac{\Gamma(1-\alpha)}{|\omega|^{1-\alpha}} sin\left(\frac{\pi}{2}(1-\alpha)\right) \tag{18}$$

where $\alpha \in (0, 1)$*; and* $\Gamma(1-\alpha) \triangleq \int_0^\infty \frac{e^{-y}}{y^\alpha} dy$*.*

Lemma 3..2. *As* $\alpha \uparrow 1$*, the integral* $\int_{-\infty}^{\infty} \frac{e^{-i\omega t}}{\pi|t|^\alpha} sgn(t)dt \to -i\ sgn(\omega)$*, i.e.,*

$$\lim_{\alpha\uparrow 1} \Gamma(1-\alpha)\ \left(\frac{2}{\pi} sin\frac{\pi}{2}\ (1-\alpha)\right) = 1. \tag{19}$$

Taking Fourier transform of the convolution in Eq. (17) and an application of Lemma 3..1 yield:

$$\begin{aligned}
\widehat{\widetilde{x}^\alpha}(\omega) &= \mathcal{F}\left(x(t) * \frac{sgn(t)}{\pi|t|^\alpha}\right) \\
&= \mathcal{F}(x(t)) \cdot \mathcal{F}\left(\frac{sgn(t)}{\pi|t|^\alpha}\right) \\
&= -i\ sgn(\omega)\ \frac{2}{\pi}\ \frac{\hat{x}(\omega)\Gamma(1-\alpha)}{|\omega|^{1-\alpha}}\ sin\left(\frac{\pi}{2}(1-\alpha)\right)
\end{aligned} \tag{20}$$

Since $\Gamma(1-\alpha) < \infty$ for $\alpha \in (0,1)$, the generalized Hilbert transform $\widetilde{x}^{\alpha}(t)$ can be evaluated by taking the inverse Fourier transform of $\widehat{\widetilde{x}^{\alpha}}(\omega)$.

The above formulation shows that reduced α puts more weight on the low frequency part of the signal $x(t)$ and hence more effectively attenuates the high-frequency noise than the classical Hilbert transform. Following Lemma 3..2, as $\alpha \uparrow 1$, Fourier transform of the signal $\frac{sgn(t)}{\pi|t|^{\alpha}}$ converges to $-i\ sgn(\omega)$. This leads to the fact, that as $\alpha \uparrow 1$, $\mathcal{H}^{\alpha}$ converges to $\mathcal{H}$, where $\mathcal{H}$ is the classical Hilbert transform defined in Eq. (13).

Analogous to the analytic signal in Eq. (15), the (complex-valued) generalized analytic signal of the real-valued signal $x(t)$ is defined as:

$$\mathcal{X}^{\alpha}(t) \triangleq x(t) + i\ \widetilde{x}^{\alpha}(t) \tag{21}$$

and the (real-valued) transfer function with input $\widehat{x}(\omega)$ and output $\widehat{\mathcal{X}}^{\alpha}(\omega)$ is formulated as:

$$G^{\alpha}(\omega) \triangleq \frac{\widehat{\mathcal{X}}^{\alpha}(\omega)}{\widehat{x}(\omega)} = 1 + sgn(\omega)\ \left(\frac{\omega_0(\alpha)}{|\omega|}\right)^{(1-\alpha)} \tag{22}$$

where $\omega_0(\alpha) \triangleq \left(\frac{2}{\pi\ \Gamma(1-\alpha)} sin\left(\frac{\pi}{2}(1-\alpha)\right)\right)^{\frac{1}{1-\alpha}}$.

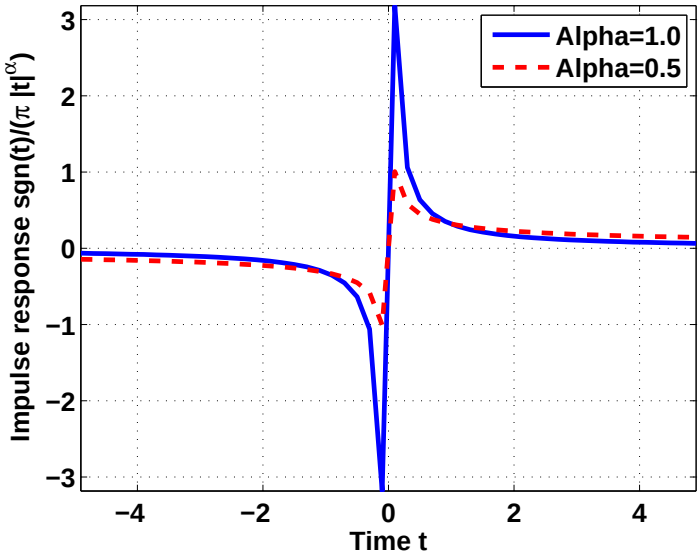


Figure 10. Impulse response $\left(\frac{sgn(t)}{\pi|t|^{\alpha}}\right)$ of the generalized Hilbert transform

Remark 3..1. *For $\alpha = 1$, it follows from Eq. (17) that the real-valued signal $x(t)$ is convoluted with $\frac{1}{\pi t}$. The implication is that the effects of memory in the signal $x(t)$ reduce as fast as $\frac{1}{\pi|t|}$. As α is decreased, the tail of the impulse response of the generalized Hilbert transform $\widetilde{x}^{\alpha}(t)$ becomes increasingly fat as seen in Fig. 10. Hence, for $0 < \alpha < 1$, the generalized analytic signal $\mathcal{X}^{\alpha}(t)$ captures more (low-frequency) information from time series data than that for $\alpha = 1$.*

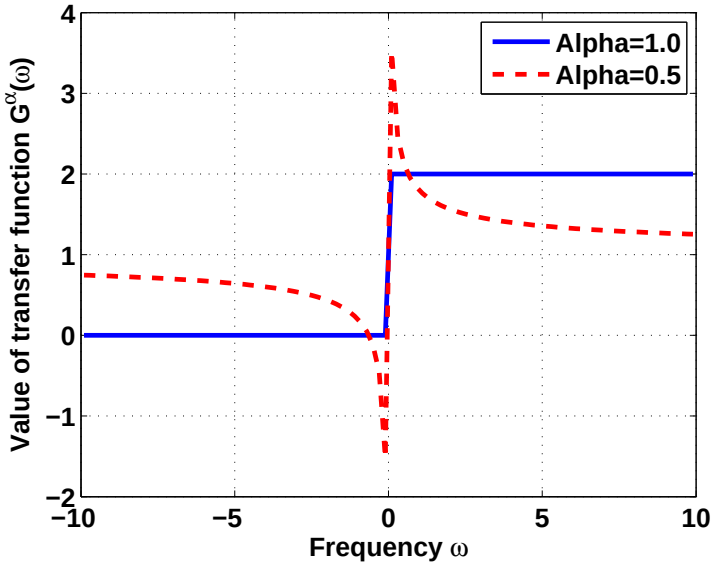


Figure 11. Transfer function $G^{\alpha}(\omega)$ of the generalized analytic signal

Remark 3..2. *Fourier transform of a real-valued signal does not contain any additional information beyond what is provided by the positive frequency components, because of the symmetry of its spectrum. Therefore, in the construction of an analytic signal in Eq. (15) and its transfer function in Eq. (16), Hilbert transform removes the negative frequency components while doubling the positive frequency components. For* $\alpha < 1$*, it follows from Fig. 11 that the negative frequency components of the transfer function* $G^{\alpha}(\omega)$ *of a generalized analytic signal are no longer zero. Therefore, the generalized analytic signal in Eq. (21) is not an analytic signal in the sense of Gabor [26] for* $\alpha < 1$*. However, the transfer functions of both analytic and generalized analytic signals are real-valued almost everywhere in the range* $\omega \in \mathbb{R}$*. The phase of the (real-valued) transfer function* $G^{\alpha}(\omega)$ *is either* 0 *or* $-\pi$ *as explained below.*

- *The phase of* $G(\omega)$ *(i.e.,* $G^{\alpha}(\omega)$ *for* $\alpha = 1$*) is* 0 *radians in the frequency range* $(0, \infty)$*. Its magnitude in the negative frequency range* $(-\infty, 0)$ *is identically equal to* 0*; therefore, the phase in this range is inconsequential.*

- *For* $0 < \alpha < 1$*, the phase of* $G^{\alpha}(\omega)$ *is* $-\pi$ *radians in the frequency range* $(-\omega_0(\alpha), 0)$*, where* ω_0 *is defined in Eq. (22), and is* 0 *radians in the frequency range* $\big(-\infty, -\omega_0(\alpha)\big) \bigcup (0, \infty)$*.*

3.1.2. Test Results and Validation

The concept of generalized Hilbert transform is tested and validated by symbolic analysis of time series data, generated from the same apparatus of nonlinear electronic systems reported in the earlier publication [13].

The nonlinear active electronic system in the test apparatus emulates the forced Duffing equation:

$$\frac{d^2x}{dt^2} + \beta\frac{dx}{dt} + x(t) + x^3(t) = Acos(\omega t) \tag{23}$$

Having the system parameters set to $\beta = 0.24$, $A = 22.0$, and $\omega = 5.0$, time series data of the variable $x(t)$ were collected from the electronic system apparatus. These data sets do not contain any substantial noise because the laboratory apparatus is carefully designed to shield spurious signals and noise. Therefore, to emulate the effects of noise in the time series data, additive first-order colored Gaussian noise was injected to the collected time series data to investigate the effects of signal-to-noise ratio (SNR). The profile of a typical signal, contaminated with 10 db additive Gaussian noise (i.e., $SNR = 10$), is shown in Fig. 12.

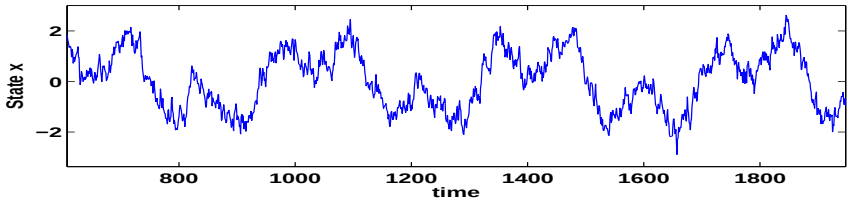


Figure 12. Signal contaminated with 10 db additive colored Gaussian noise

Construction of the Transformed Phase Space: Let the real-valued noisy time-series data $x(t)$ contain *N* data points. Upon generalized Hilbert transformation of this data sequence, a complex-valued generalized analytic signal $\mathcal{X}^\alpha(t)$ is constructed. Similar to the procedure described in [13], $\mathcal{X}^\alpha(t)$ is represented as a one-dimensional trajectory in the two-dimensional pseudo-phase space. Let Ω be a compact region in the pseudo-phase space, which encloses the trajectory of *N* such data points.

Partitioning and Symbol Generation: The next task is to partition Ω into finitely many mutually exclusive and exhaustive segments, where each segment is labelled with a symbol or letter. The partitioning is based on the magnitude and phase of the complex-valued signal $\mathcal{X}^\alpha(t)$ as well as the density of data points in these segments, following the procedure described in [13]. Each point in the partitioned data set is represented by a pair of symbols; one belonging to the alphabet Σ_R based on the magnitude (i.e., in the radial direction) and the other belonging to the alphabet Σ_A based on the phase (i.e., in the angular direction). In this way, the complex-valued signal $\mathcal{X}^\alpha(t)$ is partitioned into a symbol sequence by associating each pair of symbols to a single symbol belonging to an alphabet Σ that is defined as:

$$\Sigma \triangleq \big\{(\sigma_i, \sigma_j) \in \Sigma_R \times \Sigma_A\big\} \text{ and } |\Sigma| = |\Sigma_R||\Sigma_A| \tag{24}$$

The results presented in this chapter are generated with $\Sigma_R = 8$ in the radial direction and $\Sigma_A = 5$ in the angular direction, i.e., $|\Sigma| = 40$.

State Transition Matrices: The symbol sequence is now used to construct *D*-Markov machine models [15]. The assumption of statistical stationarity of the symbol sequence is implicit in the construction of Markov models. In this study, Markov chain models of depth $D = 1$ and $D = 2$ have been constructed.

Modeling of the symbolic process as a ($D = 1$) Markov chain involves evaluation of the Π^1 matrix, where the ij^{th} matrix element π^1_{ij} is defined as the probability that $(n+1)^{th}$ state is i given that the n^{th} state was j, i.e.,

$$\pi^1_{ij} \triangleq P\left(q^1_{n+1} = i \mid q^1_n = j\right) \tag{25}$$

where q_k is the state at discrete time instant k. Evidently, the size of the Π matrix is $|\Sigma|\times|\Sigma|$, where $|\Sigma|$ is the number of symbols in the alphabet Σ.

Modeling the symbolic process as a ($D = 2$) Markov chain involves evaluation of a 3-dimensional matrix, where the ijk^{th} matrix element π^2_{ijk} is defined as:

$$\pi^2_{ijk} \triangleq P\left(q^2_{n+2} = i | q^2_{n+1} = j, q^2_n = k\right) \tag{26}$$

and size of the (sparse) Π^2 matrix is $|\Sigma| \times |\Sigma| \times |\Sigma|$.

Remark 3..3. *Elements of both Π^1 and Π^2 matrices are estimated by conditional frequency count and their convergence requires a symbol sequence of sufficient length. This aspect has been discussed in [17][6] and is referred to as the stopping rule that assigns a bound on the length of the symbol sequence for parameter identification of the stochastic matrices Π^1 and Π^2.*

Computation of Mutual Information: Effectiveness of generalized Hilbert transform for Markov model construction has been examined from an information theoretic perspective [5]. The rationale is that, in a noise-corrupted system, higher values of mutual information imply less uncertainties in the symbol sequence. The mutual information $\mathcal{I}$ is expressed in terms of entropy $\mathcal{S}$ for both ($D = 1$) and ($D = 2$) Markov chains in the following set of equations:

$$\mathcal{I}\left(q_{n+3}; q_{n+2}\right) \triangleq \mathcal{S}\left(q_{n+3}\right) - \mathcal{S}\left(q_{n+3}|q_{n+2}\right) \tag{27}$$

$$\mathcal{S}\left(q_{n+3}\right) \triangleq -\sum_{\ell=1}^{|\Sigma|} P\left(q_{n+3} = \ell\right) \ \log_2 P\left(q_{n+3} = \ell\right) \tag{28}$$

Usage of maximum entropy partitioning [12] for symbol generation yields: $\mathcal{S}\left(q_{n+3}\right) = \log_2\left(|\Sigma|\right)$.

$$\mathcal{S}\left(q_{n+3}|q_{n+2}\right) \triangleq \sum_{\ell=1}^{|\Sigma|} P\left(q_{n+2} = \ell\right) \ \mathcal{S}\left(q_{n+3}|q_{n+2} = \ell\right) \tag{29}$$

where

$$\mathcal{S}\left(q_{n+3}|q_{n+2} = \ell\right) = -\sum_{j=1}^{|\Sigma|} P\left(q_{n+3} = j \mid q_{n+2} = \ell\right) \cdot \log_2 P\left(q_{n+3} = j|q_{n+2} = \ell\right) \tag{30}$$

$$\mathcal{I}\left(q_{n+3}; q_{n+2}, q_{n+1}\right) \triangleq \mathcal{S}\left(q_{n+3}\right) - \mathcal{S}\left(q_{n+3}|q_{n+2}, q_{n+1}\right) \tag{31}$$

$$\begin{aligned}
&\mathcal{S}\left(q_{n+3}|q_{n+2}, q_{n+1}\right) \\
&\quad \triangleq -\sum_{i=1}^{|\Sigma|}\sum_{j=1}^{|\Sigma|} P\left(q_{n+2}=i, q_{n+2}=j\right) \cdot \\
&\quad \mathcal{S}\left(q_{n+3}|q_{n+2}=i, q_{n+1}=j\right)
\end{aligned} \tag{32}$$

where

$$\begin{aligned}
&\mathcal{S}\left(q_{n+3}|q_{n+2}=i, q_{n+1}=j\right) \\
&\quad = -\sum_{\ell=1}^{|\Sigma|} P\left(q_{n+3}=\ell|q_{n+2}=i, q_{n+1}=j\right) \cdot \\
&\quad \log_2 P\left(q_{n+3}=\ell|q_{n+2}=i, q_{n+j}=j\right)
\end{aligned} \tag{33}$$

Based on Eq. (25) and Eqs. (27) to (30), the mutual information $\mathcal{I}\left(q_{n+3}; q_{n+2}\right)$ is calculated from the Π^1 matrix. Similarly, based on Eq. (26) and Eqs. (31) to (33), $\mathcal{I}\left(q_{n+3}; q_{n+2}, q_{n+1}\right)$ is calculated from the Π^2 matrix. Then, information gain (abbreviated as $\mathcal{I}_\mathcal{G}$) with $D = 2$ instead of $D = 1$ in the Markov chain construction is defined as:

$$\mathcal{I}_\mathcal{G} \triangleq \mathcal{I}\left(q_{n+3}; q_{n+2}, q_{n+1}\right) - \mathcal{I}\left(q_{n+3}; q_{n+2}\right) \tag{34}$$

Pertinent Results:

The pertinent results on mutual information and information gain are presented in Fig. 13 and Fig. 14, respectively. Although results are shown only for $SNR = \infty, 10, 4$ and 0, several other experiments with intermediate values of SNR between ∞ and 0 were performed, which shows the same trend.

The information gain is always a positive quantity as seen in Eq. (34). In other words, there is always a gain in information upon increasing the depth of the Markov chain model. Pertinent inferences, drawn from these results, are presented below.

1. Mutual information increases with decrease in α irrespective of D and SNR as seen in Fig. 13.

2. Information gain $\mathcal{I}_\mathcal{G}$ (see Eq. (34) and Fig. 14) is minimal for $SNR \to \infty$ (i.e., for the signal with no noise injection). Therefore, *D*=1 Markov chain should be adequate with $ASSP$ using conventional Hilbert transform (see Eq. 13) for low-noise signals.

3. As SNR is decreased (i.e., percentage of additive noise is increased), information gain $\mathcal{I}_\mathcal{G}$ increases for all values of α in the range of 1.0 down to about 0.2. As α is decreased, information gain decreases as seen in Fig. 14. Therefore, even for a considerable amount of noise, a smaller value of α should be able to achieve noise attenuation and thus allow usage of $D = 1$ in *D*-Markov machines.

4. Results for a pathological case with $SNR \to 0$, (i.e., complete noise capture of the signal) in Fig. 13 and Fig. 14 show similar trends as above. The crossing of the information gain curves in Fig. 14 at low values of α (e.g., $\alpha \leq 0.2$) could possibly be attributed to the effects of coarse graining [7] due to symbol generation.

The rationale for the observed trends in Fig. 13 and Fig. 14 is reiterated as follows. An increase in depth D captures the effects of longer memory in the signal, and similarly reduced α puts more weight on the low-frequency components, which assists noise reduction.

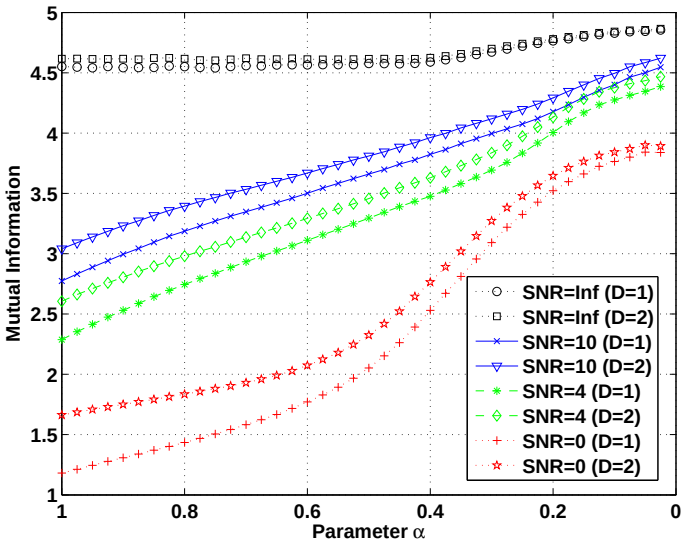


Figure 13. Profiles of mutual information

The following conclusions are drawn from the validation results of the proposed generalization of Hilbert transform on a laboratory apparatus of nonlinear electronic systems as presented above.

- Generalized Hilbert transform with a smaller value of parameter α is capable of extracting more information from a data sequence irrespective of the depth of the D-Markov machine chosen for modeling.
- Information gain for a larger depth *D* reduces with smaller values of the parameter α.
- By selecting small values of the parameter α in the generalized Hilbert transform, it is possible to avoid using a computationally expensive larger depth *D* without loss of significant information.

3.2. Optimal Feature Extraction via Partitioning

Partitioning is the process of coarse-graining of the data-space under some notion of similarity, transforms continuous valued variables to a discrete symbolic space. This is an important pre-processing step essential for several applications such as *symbolic time-series*

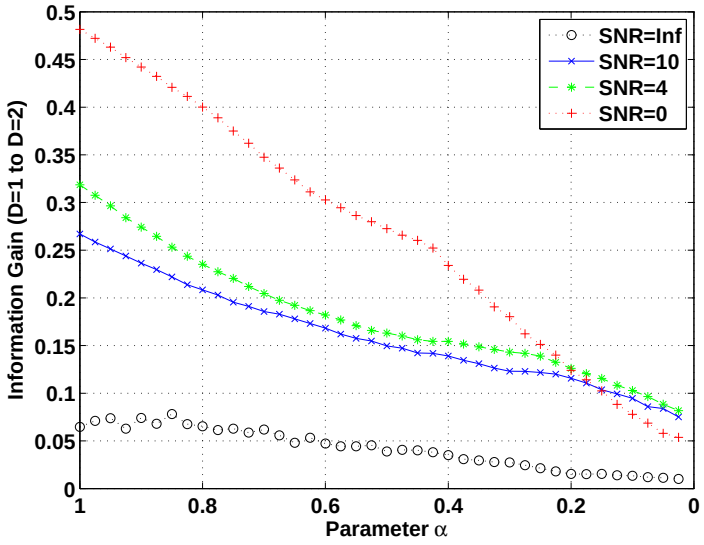


Figure 14. Profiles of information gain

analysis of dynamical systems, data-mining and machine learning. Despite the necessity and importance of discretization, there is no standard way to approach it. This is because several factors such as the nature of the dynamical system or data set in question, choice of the similarity metric, and desired model simplicity affect the nature of a discretization scheme that is appropriate. Moreover, even if one can define the optimality criteria for discretization, the process will be NP-complete [31]. As pointed out by [32], while discretization is desirable pre-processing step, practical discretization schemes are necessarily heuristic in nature and a large number of such schemes have been proposed in the literature. The simplest methods include *equal interval width* (uniform) and *equal interval frequency* (maximum entropy) discretization or partitioning. One key distinction among different methods is whether the discretization process was *supervised* or *unsupervised*. In a supervised process a metric based on a discrete variable (e.g. class label) is used as a feedback to guide the discretization process, while a unsupervised method does not use such information. The literature is vast and there are several reviews available that summarize and compare different techniques [33, 34, 35, 36, 37]. This section reviews two recently developed multi-variate partitioning schemes with particular objective of symbolic dynamic analysis of time-series data from physical systems. While the first scheme (maximally bijective discretization) aims to optimize symbolic modeling of dynamical systems, the second scheme (optimal partitioning for classification) is particularly useful for time-series classification.

3.2.1. Maximally Bijective Partitioning

Consider a dynamical system with input/output signals as:

$$U(t) = 2\cos(0.25t) \quad (35)$$

$$Y(t) = \cos(0.5t) \quad (36)$$

To build a symbolic model of such a system, both input and output space need to be discretized. However, if the discretization of input and output spaces are completely independent, then important functional relationship in the continuous domain may not remain preserved in the discrete domain. Based on this key idea, this section reviews the recently developed methodology and algorithm for a discretization scheme that maximizes the degree of input-output symbol correspondence, hence named Maximally Bijective Discretization (MBD).

Although the required notations follow earlier discussion, it is repeated here for enhanced readability. Let the time series data of a variable generated from a physical complex system or its dynamical model be denoted as **q**. A compact (i.e., closed and bounded) region $\Omega \in \mathbb{R}^n$, where $n \in \mathbb{N}$, within which the time series is circumscribed, is identified. Let the space of time series data sets be represented as $\mathbf{Q} \subseteq \mathbb{R}^{n \times N}$, where the $N \in \mathbb{N}$ is sufficiently large for convergence of statistical properties within a specified threshold. Note, n represents the dimensionality of the time-series and N is the number of data points in the time series. A discretization encodes Ω by introducing a partition $\mathbb{B} \equiv \{B_0, ..., B_{(p-1)}\}$ consisting of p mutually exclusive (i.e., $B_j \cap B_k = \emptyset \; \forall j \neq k$), and exhaustive (i.e., $\cup_{j=0}^{p-1} B_j = \Omega$) cells. For one-dimensional time series data, a discretization consisting of p cells is represented by $p-1$ points that serve as cell boundaries. In the sequel, a p-cell discretization $\mathbb{B}$ is also expressed as $\Gamma_m \triangleq \{\gamma_1, \gamma_2, \cdots, \gamma_{m-1}\}$, where γ_i denotes a bin boundary.

Let the complex system under consideration has m real-valued output variables and n real-valued input variables that are denoted as $w_1, \cdots, w_m$ and $u_1, \cdots, u_n$ respectively. In the continuous domain, an input variable u_i for any i is represented as a function of all output variables for the purpose of discretization as follows:

$$u_i = h_i(w_1, \cdots, w_m) \tag{37}$$

Suppose a m-dimensional discretization is imposed on the space of output variables that divides the data for output variables into K discrete classes. The goal is to discretize each input variable based on the defined classes of output variables. This work develops a *Maximally Bijective* scheme of performing the discretization of each input variable separately. The problem is formulated in the sequel for any input variable u (omitting the subscript).

Let $\mathbb{B} \equiv \{B_0, ..., B_{(l-1)}\}$ be a discretization of one of the input variables, where $B_j, j \in \{0, ..., l-1\}$ is a bin of the discretization. With this setup, $correspondence$ between a class and a bin is defined as follows:

$$\begin{aligned} \text{If } \; i \quad &= \quad \arg\max_k P(C_k | x \in B_j) \\ \text{then, } \; C_i \quad &\Rightarrow \quad B_j \end{aligned} \tag{38}$$

where, $P(\cdot)$ denotes a probability function and $x \in \mathbb{R}$. In words, class C_i corresponds to bin B_j. Note, a class C_i may correspond to more than one disjoint bins. From this perspective, a reward function is defined as follows:

$$R(\mathbb{B}|x) = \{P(C_i | x \in B_j) : C_i \Rightarrow B_j\} \tag{39}$$

Note that this reward function signifies the notion of bijection, i.e., with higher reward, the probability of a class being corresponding to a bin increases. With this setup, the total

expected reward is calculated as

$$TR(\mathbb{B}) = \int_X R(\mathbb{B}|x)P(x)dx \tag{40}$$

The goal here is to maximize this total reward function, hence the discretization scheme is called Maximally Bijective. It is clear from the formulation that maximizing the total reward function is equivalent to maximizing $R(\mathbb{B}|x)$ at each x.

$$\begin{aligned}\mathbb{B}^* &= \arg\max_{\mathbb{B}} \{R(\mathbb{B}|x)\ \forall x\} \\ &= \arg\max_{\mathbb{B}} \{P(C_i|x \in B_j) : C_i \Rightarrow B_j\ \forall x\}\end{aligned} \tag{41}$$

This observation leads to an algorithm that is developed and used to identify MBD in this work. Let $P_m(x)$ denotes $\{P(C_i|x \in B_j) : C_i \Rightarrow B_j\}$ at any x. Note, by this definition

$$P_m(x) = \max_i P(C_i|x)\ \forall x \tag{42}$$

With this setup an overview of the proposed algorithm is provided below:

Overview of the Algorithm

$x = \min(D)$
$k = 1$
while $x < \max(D)$ **do**
  Identify C_i such that $P(C_i|x) = P_m(x)$;
  Identify C_j such that $P(C_j|x + dx) = P_m(x + dx)$;
  if $i \neq j$ **then**
    $\gamma_k = x$ % Note, γ_k denotes the k^{th} bin boundary
    $k \leftarrow k + 1$
  end if
  $x \leftarrow x + dx$
end while

However, the primary issue with the above process is reliable estimation of $P(C_i|x)$ from data [32]. This work adopts a basic frequency counting (over an interval) method to estimate this conditional probability. This means that the optimization process begins with considering a small window in the domain (around $\min(D)$) to identify the most probable class in that interval. Then the window is slided across the data range of the input variable and bin boundaries (γ_ks) are placed where the most probable class changes from one interval to the next. However, this approximation may result in sub-optimality of the solution. The rationale is as follows: Let $[a, b] \subset \mathbb{R}$ be an interval over which the frequencies of different classes are estimated to identify the most probable class. The frequency of class C_i in the interval $[a, b]$ is denoted as $n(C_i)|_{[a,b]}$ and it can be represented as

$$n(C_i)|_{[a,b]} = \int_a^b P(C_i|x)P(x)dx \tag{43}$$

To identify the most probable class in $[a, b]$, one needs to compare $n(C_i)|_{[a,b]}$ with

$n(C_j)|_{[a,b]}$. However, it is known that

$$\begin{aligned} \int_a^b P(C_i|x)P(x)dx &\gtrless \int_a^b P(C_j|x)P(x)dx \\ \nRightarrow P(C_i|x) &\gtrless P(C_j|x)\ \forall x \in [a,b] \end{aligned} \tag{44}$$

Consequently, the solution may become suboptimal due to the consideration of intervals of finite width. Also, in a realistic setting, the intervals need to be wide enough to avoid significant effects of noise in the data.

However, the following lemma can be stated in this scenario:

Lemma 3..3. *The total reward (TR) is a nondecreasing function of adding new bin boundaries.*

This ensures that the total reward at least does not reduce when a new bin boundary is introduced in the sequential algorithm proposed here. A rough proof sketch of this property is provided here:

Proof Sketch: First of all, it should be noted that introduction of a new bin boundary can be considered as splitting one of the current bins. Let B_j be the bin which is splitted into B_j^1 and B_j^2 with the introduction of a new bin boundary. Also, let class C_i be the most probable class of the bin B_j with frequency $n(C_i)|_{B_j}$. Let the total reward function before and after splitting be denoted as $TR(k)$ and $TR(k+1)$ respectively. Now, three cases are possible with this splitting.

- *Case I:* In both new bins B_j^1 and B_j^2, C_i is no longer the most probable class and the most probable classes in B_j^1 and B_j^2 are say C_p and C_q respectively. In this case,

$$\begin{aligned} TR(k+1) = n(C_p)|_{B_j^1} + n(C_q)|_{B_j^2} > \\ n(C_i)|_{B_j^1} + n(C_i)|_{B_j^2} = TR(k) \end{aligned} \tag{45}$$

- *Case II:* In one of the new bins, say in B_j^1, C_i is still the most probable class. However, in B_j^2, C_q is the most probable class. In this case,

$$\begin{aligned} TR(k+1) = n(C_i)|_{B_j^1} + n(C_q)|_{B_j^2} > \\ n(C_i)|_{B_j^1} + n(C_i)|_{B_j^2} = TR(k) \end{aligned} \tag{46}$$

- *Case III:* In both new bins, C_i is still the most probable class. In this case,

$$TR(k+1) = n(C_i)|_{B_j^1} + n(C_i)|_{B_j^2} = TR(k) \tag{47}$$

Therefore, $TR(k+1) \geq TR(k)$, i.e., total reward does not decrease when new bin boundaries are introduced sequentially in the algorithm described above. This is also compatible with the intuitive notion of increase in reward with increase in complexity.

Figure 15 shows the MBD for the illustrative example problem introduced in the beginning. It is clear from the result that in a numerically stable scenario, the present approach can identify the bin boundaries of an admissible discretization. Furthermore, Fig. 16

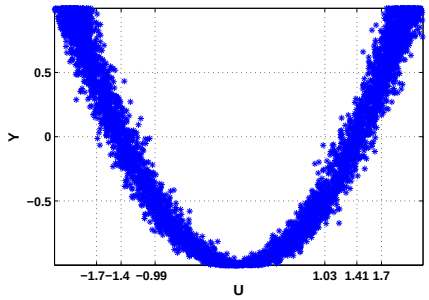


Figure 15. Illustrative example: MBD of noisy input U Given a Uniform discretization of output Y; Bin boundaries are marked on corresponding axes as grid lines

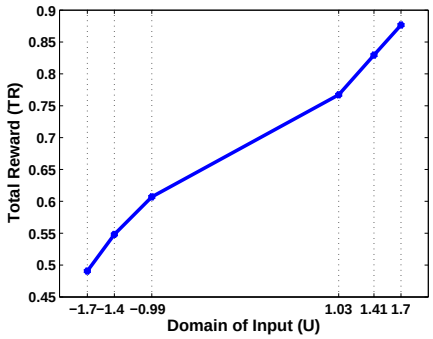


Figure 16. Monotonic increase in Total Reward with addition of new bin boundaries

shows that the total reward monotonically increases with addition of new bin boundaries as stated in Lemma 3..3. The total reward of the MBD in this example is found to be 0.88. For comparison purposes, uniform discretization of same complexity, i.e., with same number of cells/bins (with bin boundaries $[-1.66, -1.00, -0.350.300.961.61]$) has total reward of 0.79 and maximum entropy discretization with same complexity (with bin boundaries $[-1.80 - 1.25 - 0.450.451.231.80]$) has total reward of 0.76. The following remark summarizes some of the key aspects the proposed discretization policy.

Remark 3..4. *The MBD scheme developed here avoids computationally expensive iterative search for bin boundaries in the data space. Instead, this algorithm scans data space once to sequentially place optimal bin boundaries and in turn identifies the optimal number of cells/bins in a stable numerical scenario. Furthermore, it should be noted that discretization of the output variables can be revised once MBD of the input variables are obtained. This process essentially involves removing certain bin boundaries of output variables that do not have effect in the MBD of the input variables. Finally, this scheme can be used for any complex dynamical system. However, it will be particularly useful for highly nonlinear systems where traditional uniform or maximum entropy discretizations may fail to preserve the functional relationships by a great deal.*

It should be noted that there are other popular cost functions (involving e.g., maximizing mutual information and minimizing Bayes risk) available in literature that may have similar effects on discretization as the current reward function proposed in this study. Therefore,

comparison of the strategy presented here with the other similar cost/reward functions is an important topic of future investigation.

3.2.2. Optimal partitioning for Classification

In the SDF framework, a symbol sequence $\{\mathbf{s}\}$ is compressed by a probabilistic finite state automaton ($PFSA$), where $j, k \in \{1, 2, ..., r\}$ are the states of the $PFSA$ with the $(r \times r)$ state transition matrix $\Pi = [\pi_{jk}]$ that is obtained at slow-scale epochs (different classes in this context). (Note: Π is a stochastic matrix, i.e., the transition probability $\pi_{jk} \geq 0$ and $\sum_k \pi_{jk} = 1$). To compress the information further, the state probability vector $\mathbf{p} = [p_1 \ \cdots \ p_r]$ that is the left eigenvector corresponding to the (unique) unity eigenvalue of the irreducible stochastic matrix Π is calculated. The vector $\mathbf{p}$ is the extracted feature vector and is a low-dimensional compression of the long time series data representing the dynamical system at the slow-scale epoch.

For classification using SDF, the reference time series, belonging to a class denoted as Cl_1, is symbolized by one of the standard partitioning schemes (e.g., Uniform Partitioning (UP) or Maximum Entropy partitioning (MEP)) [15, 12, 13]. Then, using the steps described earlier, a low-dimensional feature vector $\mathbf{p}^{Cl_1}$ is constructed for the reference slow-scale epoch. Similarly, from a time series belonging to a different class denoted as Cl_2, a feature vector $\mathbf{p}^{Cl_2}$ is constructed using the same partitioning as in Cl_1. The next step is to classify the data in the constructed low-dimensional feature space. In this respect, there are many options for selecting classifiers that could either be parametric or non-parametric. Among the parametric classifiers, one of the commonly used techniques relies on the second-order statistics in the feature space, where the mean feature is calculated for every class along with the variance of the feature space distribution in each class of the training set. Then, a test feature vector is classified by using the Mahalnobis distance [38] or the Bhattacharya distance [39] of the test vector from the mean feature vector of each class. However, these methods are not efficient for non-Gaussian distributions, where the feature space distributions may not be adequately described by the second order statistics. Consequently, a non-parametric classifier (e.g., k-NN classifier [40]) is potentially a better choice. In this study, the k-NN classifier has been used because the Gaussian probability distribution cannot be assured in the feature space. However, in general, any other suitable classifier, such as the Support Vector Machines (SVM) or the Gaussian Mixture Models (GMM) may also be used [40]. To classify the test data set, the time series sets are converted into feature vectors using the same partitioning that has been used to generate the training features. Then, using the labeled training features, the test features are classified by a k-NN classifier with suitable specifications (neighborhood size and distance metric).

Many optimization criteria have been reported in literature for feature extraction in multi-class classification problems. However, none can be more fundamental than minimization of classification error. Although, there have been attempts to approximate the classification error as the Bayes error using pairwise weighting functions for multi-class problems [41], the Bayes error itself cannot be expressed analytically except a few special cases. On the other hand, even if the Fisher criteria is not directly equivalent to classification error, it is widely used as a feature extraction optimization criteria in literature. This criteria involves maximization of the inter-class variance and minimization of the intra-class

variance for the training data set. For a K-class ($K > 2$) classification problem, a general K-class Fisher criterion may be used; alternatively, a sum of $K(K-1)/2$ number of 2-class Fisher criteria may be used as well. This criterion is very useful for binary classification problems, especially when the samples are distributed in a Gaussian manner in the feature space. For multi-class problems [42], a criterion has been proposed based on the minimum classification error (MCE) that is calculated by the misclassification rate over the training samples. Formally, these two (fundamentally different) optimization criteria are known as the (i) Filter method and (ii) Wrapper method. While a filter method uses information content feedback (e.g., Fisher criterion, and statistical dependence) as optimization criteria for feature extraction, a wrapper method includes the classifier inside the optimization loop and attempts to maximize the predictive accuracy (e.g., classification rate) using statistical resampling or cross-validation [40]. In this methodology, the wrapper method is adopted (i.e., via minimization of the classification error on the training set) for optimization, primarily because of the non-binary nature of the problem at hand and the possible non-Gaussian distribution of training samples in the feature space.

In this context, ideally one should jointly minimize every element of the confusion matrix [43]. However, in that case, the dimension of the objective space may rapidly increase with an increase in the number of classes. To circumvent this situation in the present work, two costs are defined on the confusion matrix by using another weighting matrix, elements of which denote the relative penalty values for different confusions in the classification process. Formally, let there be $Cl_1, \cdots, Cl_n$ classes of labeled time-series data in the training set. A partitioning $\mathbb{B}$ is employed to extract features from each sample and a k-NN classifier $\mathbb{K}$ is used as a classifier. The confusion matrix $\mathbf{C}$ is obtained upon completion of the classification process, where the ij^{th} element c_{ij} of $\mathbf{C}$ denotes the frequency of data from class Cl_i being classified as data from Cl_j. Let $\mathbf{W}$ be the weighting matrix, where the ij^{th} element w_{ij} of $\mathbf{W}$ denotes the penalty incurred for classifying data from Cl_i as data from class Cl_j. (Note: Since there is no penalty for correct classification, the diagonal elements of $\mathbf{W}$ are identically equal to 0, i.e., $w_{ii} = 0 \; \forall \; i$.) With these specifications, two costs, $CostE$ and $CostW$, that are to be minimized are defined as follows.

The cost $CostE$ due to expected classification error is defined as:

$$CostE = \frac{1}{N_s}\left(\sum_i \sum_j w_{ij} c_{ij}\right) \tag{48}$$

where N_s is the total number of training samples including all classes. The above equation represents the total penalty for misclassification across all classes. Thus $CostE$ is related to the expected classification error. The weights w_{ij} are selected based on the prior knowledge about the data and the user's requirements.

It is implicitly assumed in many supervised learning algorithms that the training data set is a statistically similar representation of the whole data set. However, this assumption may not be accurate in practice. A solution to this problem is to choose a feature extractor that minimizes the worst-case classification error [44]. In this setting, that cost $CostW$ due to worst-case classification error is defined as:

$$CostW = \max_i \left(\frac{1}{N_i}\sum_j w_{ij} c_{ij}\right) \tag{49}$$

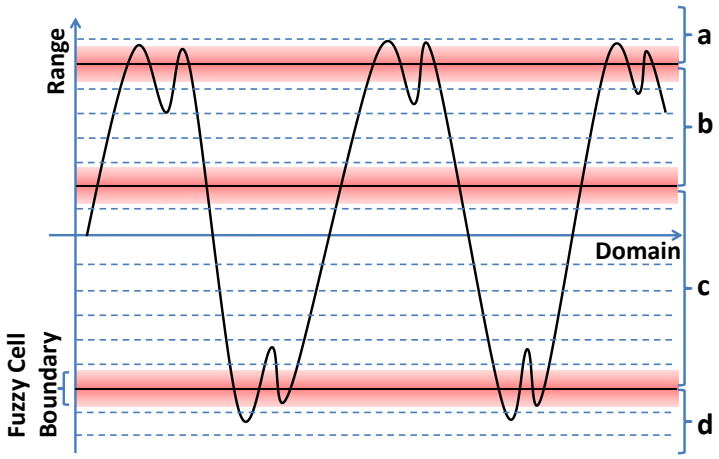


Figure 17. Fuzzy Cell Boundaries to obtain $CostE_{robust}$ and $CostW_{robust}$

where N_i is the number of training samples in the class Cl_i. (Note: In the present formulation, the objective space is two-dimensional for a multi-class classification problem and the dimension is not a function of the number of classes.)

Sensitivity and Robustness

Cost minimization requires a choice of partitioning that could be sensitive to class separation in the data space. However, the enhanced sensitivity of optimal partitioning may cause large classification errors due to noise and spurious disturbances in the data and statistical variations among training and testing data sets. Hence, it is important to invoke robustness properties into the optimal partitioning. In this study, robustness has been incorporated by modifying the costs $CostE$ and $CostW$, introduced above.

For one-dimensional time series data, a partitioning consisting of $|\Sigma|$ cells is represented by $(|\Sigma|-1)$ points that serve as cell boundaries. In the sequel, a Σ-cell partitioning $\mathbb{B}$ is expressed as $\Lambda_{|\Sigma|} \triangleq \{\lambda_1, \lambda_2, \cdots, \lambda_{|\Sigma|-1}\}$, where λ_i denotes a partitioning boundary. Thus, a $Cost$ is dependent on the specific partitioning $\Lambda_{|\Sigma|}$ and is denoted by $Cost(\Lambda_{|\Sigma|})$. The key idea of a robust cost for a partitioning $\Lambda_{|\Sigma|}$ is that it is expected to remain invariant under small perturbations of the partitioning points, $\lambda_1, \lambda_2, \cdots, \lambda_{|\Sigma|-1}$ (i.e., fuzzy cell boundaries). To define a robust cost, a distribution of partitioning $f_{\Lambda_{|\Sigma|}}(\cdot)$ is considered, where a sample of the distribution is denoted as $\tilde{\Lambda}_{|\Sigma|} = \{\tilde{\lambda}_1, \tilde{\lambda}_2, \cdots, \tilde{\lambda}_{|\Sigma|-1}\}$ and $\tilde{\lambda}_i$'s are chosen from independent Gaussian distributions with mean λ_i and a uniform standard deviation σ_λ; the choice of σ_λ is discussed later.

The resulting cost distribution is denoted as $f_{Cost(\Lambda_{|\Sigma|})}(\cdot)$, and $Cost_{robust}$ is defined as the cost value below which 95% of the population of distribution $f_{Cost(\Lambda_{|\Sigma|})}(\cdot)$ remains and is denoted as:

$$Cost_{robust} = P_{95}[f_{Cost(\Lambda_{|\Sigma|})}(\cdot)] \tag{50}$$

For the analysis to be presented here, it is assumed that sufficient samples are generated to obtain a good estimate of $Cost_{robust}$ as explained in Fig. 17, where the firm lines denote

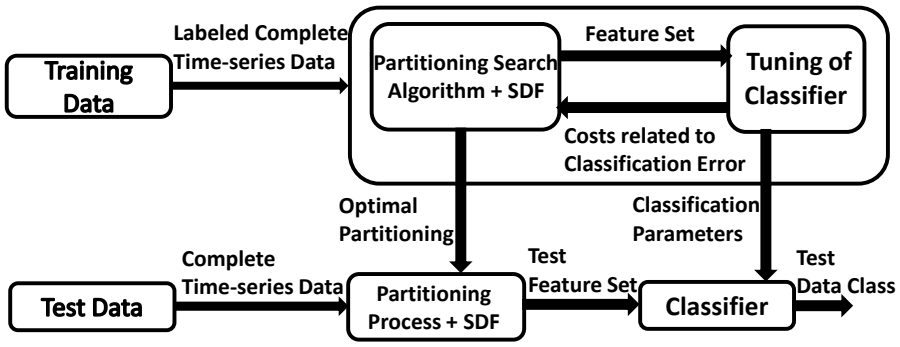


Figure 18. General Framework for Optimization of Feature Extraction

the boundaries that divide the range space into four cells, namely a, b, c and d. However, in general, the cell boundaries are fuzzy in nature, which leads to a probabilistic $Cost_{robust}$ instead of a deterministic cost based on the firm partitioning boundaries. Thus, a robust cost due to expected classification error, $CostE_{robust}$, and a robust cost due to worst-case classification error, $CostW_{robust}$, are defined for a given partitioning $\Lambda_{|\Sigma|}$.

Finally, a multi-objective overall cost $CostO$ is defined as a linear convex combination of $CostE_{robust}$ and $CostW_{robust}$ in terms a scalar parameter $\alpha \in [0, 1]$.

$$CostO \triangleq \alpha CostE_{robust} + (1 - \alpha) CostW_{robust} \tag{51}$$

Ideally, the optimization procedure involves construction of the Pareto front [45] by minimizing $CostO$ for different values of α that can be freely chosen as the operating point. However, for simplicity α is taken to be 0.5 in this work, i.e., equal weight for the costs $CostE_{robust}$ and $CostW_{robust}$. Thus, the optimal Σ-cell partitioning $\mathbb{B}^*$ is the solution to the following optimization problem:

$$\mathbb{B}^* = \arg\min_{\mathbb{B}} CostO(\mathbb{B}) \tag{52}$$

Figure 18 depicts a general outline of the classification process. Labeled time series data from the training set are partitioned. The low-dimensional feature vectors that are generated by symbolization and $PFSA$ construction are fed to the classifier. After classification, the training error costs, defined above, are computed and fed back to the feature extraction block. In the classification aspect, the classifier may be tuned to the obtain better classification rates. For example, for k-NN classifiers [40], the choice of neighborhood size or the distance metric can be tuned. Similarly, for support vector machines [40], an appropriate hyperplane should be selected to achieve good classification. The key idea is to update the partitioning to reduce the cost based on the feedback. The iteration is continued until the set of optimal partitioning in a multi-objective scenario and the correspondingly tuned classifier are obtained. Generally, choice of the optimal partitioning is made based on the choice of operating point α by the user. After the choice is made, the optimal partitioning and the tuned classifier are used to classify the test data set. Although this is the general framework that is being proposed for the optimization methodology, tuning of the classifier is not addressed in this chapter, because the main focus here is to choose the optimal partitioning to minimize the cost related to classification errors.

Optimization Procedure: A sequential search-based technique has been adopted in this study for optimization of the partitioning, which was previously proposed in [46]. As the continuity of the partitioning function with respect to the range space of classification error-related costs may not exist or at least are not adequately analyzed, gradient-based optimization methods are not explored here. To construct the search space, a suitably fine grid size depending on the data characteristics needs to be assumed. Each of the grid boundaries denotes a possible position of a partitioning cell boundary, as illustrated in Fig. 17. Here, the dotted lines denote the possible positions of a partitioning cell boundary and as discussed before, for a chosen partitioning (denoted by firm lines), the partitioning boundaries are perturbed to obtain a $Cost_{robust}$.

Let the data space region Ω be divided into G grid cells for search, i.e., there are $(G-1)$ grid boundaries excluding the boundaries. Thus, there are $|\Sigma|-1$ partitioning boundaries to choose among $(G-1)$ possibilities, i.e., the number of elements (i.e., $(|\Sigma|-1)$-dimensional partitioning vectors) in the space $\mathcal{P}$ of all possible partitioning is: ${}^{(G-1)}C_{(|\Sigma|-1)}$. It is clear from this analysis that the partitioning space $\mathcal{P}$ may become significantly large with an increase in values of G and $|\Sigma|$ (e.g., for $G >> |\Sigma|$, computational complexity increases approximately by a factor of $G/|\Sigma|$ with increase in the value of $|\Sigma|$ by one). Furthermore, for each element of $\mathcal{P}$, a sufficiently large number of perturbed samples need to be collected in order to obtain the $Cost_{robust}$. Therefore, usage of a direct search approach becomes infeasible for evaluation of all possible partitioning. Hence, a sub-optimal solution is developed in this chapter to reduce the computational complexity of the optimization problem.

The objective space consists of the scalar-valued cost $CostO$, while decisions are made in the space $\mathcal{P}$ of all possible partitionings. The overall cost is dependent on a specific partitioning Λ and is denoted by $CostO(\Lambda)$. This sub-optimal partitioning scheme involves sequential estimation of the elements of the partitioning Λ.

The partitioning process is initiated by searching the optimal cell boundary to divide the data set into two cells, i.e., $\Lambda_2 = \{\lambda_1\}$, where λ_1 is evaluated as

$$\lambda_1^* = \arg\min_{\lambda_1} CostO(\Lambda_2) \tag{53}$$

Now, the two-cell optimal partitioning is given by $\Lambda_2^* = \{\lambda_1^*\}$.

Note that to obtain $CostO$ (i.e., both $CostE_{robust}$ and $CostW_{robust}$) for a given partitioning, a suitable σ_λ needs to be chosen. Let the gap between two search grid boundaries (i.e., two consecutive dotted lines in Fig. 17) be l_λ, and σ_λ is chosen as $l_\lambda/3$ in this study. The choice of such a standard deviation of the Gaussian perturbation is made for approximately complete coverage of the search space. Note that there could be an overlap of perturbation regions between two consecutive search grid boundaries, which leads to a smoother (that is essentially robust) cost variation across the domain space. In addition, depending on the gap l_λ and data characteristics, a suitable sample size is chosen to approximate the cost distribution under the fuzzy cell boundary condition.

The next step is to partition the data into three cells as Λ_3) by dividing either of the two existing cells of Λ_2^* with the placement of a new partition boundary at λ_2, where λ_2 is evaluated as

$$\lambda_2^* = \arg\min_{\lambda_2} CostO(\Lambda_3) \tag{54}$$

where $\Lambda_3 = \{\lambda_1^*, \lambda_2\}$. The optimal 3-cell partitioning is obtained as $\Lambda_3^* = \{\lambda_1^*, \lambda_2^*\}$. In this (local) optimization procedure, the cell that provides the largest decrement in $CostO$ upon further segmentation ends up being partitioned. Iteratively, this procedure is extended to obtain the $|\Sigma|$ cell partitioning as follows.

$$\lambda_{|\Sigma|-1}^* = \arg\min_{\lambda_{|\Sigma|-1}} CostO(\Lambda_{|\Sigma|}) \tag{55}$$

where $\Lambda_{|\Sigma|} = \Lambda_{|\Sigma|-1}^* \cup \{\lambda_{|\Sigma|-1}\}$ and the optimal $|\Sigma|$ cell partitioning is given by $\Lambda_{|\Sigma|}^* = \Lambda_{|\Sigma|-1}^* \cup \{\lambda_{|\Sigma|-1}^*\}$

In this optimization procedure, the cost function decreases monotonically with every additional sequential operation, under the assumption of correct estimation of $CostE_{robust}$, $CostW_{robust}$ and hence, $CostO$ under the fuzzy cell boundary condition. Formally, $CostO(\Lambda_{|\Sigma|-1}^*) \geq CostO(\Lambda_{|\Sigma|}^*)$. as explained below.

Let $\Lambda_{|\Sigma|-1}^*$ be the ($|\Sigma|$-1)-cell partitioning that minimizes $CostO$, based on the algorithm, $\Lambda_{|\Sigma|} = \Lambda_{|\Sigma|-1}^* \cup \{\lambda_{|\Sigma|-1}\}$). If $\lambda_{|\Sigma|-1}$ is chosen such that it already belongs to $\Lambda_{|\Sigma|-1}^*$, then there would be no change in the partitioning structure, i.e.,

$$CostO(\Lambda_{|\Sigma|}) = CostO(\Lambda_{|\Sigma|-1}^*) \text{ for } \lambda_{|\Sigma|-1} \in \Lambda_{|\Sigma|-1}^* \tag{56}$$

If $\lambda_{|\Sigma|-1} \in \Lambda_{|\Sigma|-1}^*$, then the partitioning $\Lambda_{|\Sigma|}$ is essentially treated as a $(|\Sigma| - 1)$-cell partitioning for the purpose of cost calculation. By definition,

$$CostO(\Lambda_{|\Sigma|}^*) \leq CostO(\Lambda_{|\Sigma|}) \ \forall \Lambda_{|\Sigma|} \tag{57}$$

Then, it follows that,

$$\min(CostO(\Lambda_{|\Sigma|-1})) \geq \min(CostO(\Lambda_{|\Sigma|})) \tag{58}$$

Or,

$$CostO(\Lambda_{|\Sigma|-1}^*) \geq CostO(\Lambda_{|\Sigma|}^*) \tag{59}$$

The monotonicity in the cost function allows formulation of a rule for termination of the sequential optimization algorithm. The process of creating additional partitioning cells is stopped if the cost decrease falls below a specified positive scalar threshold η_{stop} and the stopping rule is as follows.

$\Lambda_{|\Sigma|-1}^*$ is the optimal partitioning (and $|\Sigma| - 1$ is the optimal Alphabet size) if

$$CostO(\Lambda_{|\Sigma|-1}^*) - CostO(\Lambda_{|\Sigma|}^*) \leq \eta_{stop}. \tag{60}$$

In contrast to the direct search of the entire space of partitioning, the computational complexity of this approach increases linearly with $|\Sigma|$. This approach also allows the user to have finer grid size for the partitioning search.

Validation Example 1: Fatigue Damage Classification- As described earlier, the process of fatigue damage is broadly classified into two phases: crack initiation and crack propagation. For validation purpose, the crack propagation stage is divided into four classes as presented below.

Class	Crack Length
1	0.5 to 1.75 mm
2	1.75 to 3.5 mm
3	3.5 to 5.5 mm
4	more than 5.5 mm

As described earlier, the weighing matrix W is chosen based on the user's requirement. In the present case, W is defined from the perspective of risk involved in miss-classification. For example, penalty is low when data in Class 1 (i.e., small crack length) is classified as in classes for larger crack length; however, penalty is high for the converse. For the results presented in this chapter, W matrix is defined as follows.

$$\mathbf{W} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 3 & 0 & 1 & 2 \\ 6 & 3 & 0 & 1 \\ 9 & 6 & 3 & 0 \end{pmatrix}$$

In the present work, a large volume of ultrasonic data has been collected for different crack lengths on different specimens to demonstrate the efficiency of the present classification scheme to account for the stochastic nature of material properties. The experimental procedure has been discussed earlier. Classification results for the crack propagation phase using the optimal partitioning along with results obtained by using Maximum Entropy and Uniform partitioning [12] are provided here. The classification process is started by wavelet transformation of the time series data with suitable scales and time shifts for a given basis function. Each transformed signal is normalized with the maximum amplitude of transformed signal obtained at the beginning of experiment, when there is no damage in the specimen. The data are normalized to mitigate the effects of variability in the placement of ultrasonic sensors during different experiments.

As described earlier, both training and test data sets are divided into four classes based on crack length as all the ultrasonic signals are labeled with the corresponding crack lengths. The sequential optimization of partitioning has been carried out on the training data set to find optimal partitioning. The specifications of SDF and the classifier remain same as in the first example. The optimal alphabet size is 6 with stopping threshold value $\eta_{stop} = 0.005$. The confusion matrices for Optimal, Uniform and Maximum Entropy Partitioning on the test data set are given by $\mathbf{C}_{test}^{OptP}$ $\mathbf{C}_{test}^{UP}$ and $\mathbf{C}_{test}^{MEP}$ respectively. Table 2 shows the comparison of classification performances using different partitioning processes.

$$\mathbf{C}_{test}^{OptP} = \begin{pmatrix} 93 & 7 & 0 & 0 \\ 5 & 89 & 6 & 0 \\ 0 & 4 & 92 & 4 \\ 0 & 3 & 7 & 90 \end{pmatrix}$$

$$\mathbf{C}_{test}^{UP} = \begin{pmatrix} 94 & 5 & 1 & 0 \\ 15 & 74 & 11 & 0 \\ 0 & 9 & 85 & 6 \\ 1 & 5 & 11 & 83 \end{pmatrix}$$

$$\mathbf{C}_{test}^{MEP} = \begin{pmatrix} 93 & 7 & 0 & 0 \\ 12 & 82 & 6 & 0 \\ 0 & 7 & 89 & 4 \\ 1 & 3 & 7 & 89 \end{pmatrix}$$

Table 2. Performance Comparison of Partitioning Schemes

Partitioning	$Cost_E$	$Cost_W$
OptP	0.255	0.5
UP	0.40333	0.68
MEP	0.31333	0.52

It is observed that both costs $Cost_E$ and $Cost_W$ are reduced for optimal partitioning as compared to maximum entropy and uniform partitioning. It is also evident from the confusion matrix that optimal partitioning has improved the classification results. A close observation of confusion matrix indicates that chances of higher damage level data samples being classified as a lower level damage is reduced.

Validation Example 2: Fault detection in Nuclear power plants - Component-level anomaly detection in nuclear power plants involves identification of the anomaly type and location & quantification of the anomaly level. Although the model configuration in the IRIS test-bed can be easily extended to simultaneous anomalies in multiple components, this work deals with a single component, namely, the reactor coolant pump (RCP), where the task is to detect an anomaly and identify its level for (possibly unknown) sensor noise variance. The RCP overspeed percentage (ψ_{RCP}) is chosen as a health parameter. Table 3 shows the approximate ranges of ψ_{RCP} under different anomaly levels. Here, the low anomaly level indicate very minimal overspeed in RCP and hence also includes the absolute nominal health condition ($\psi_{RCP} = 0$). Similarly, depending on the standard deviation of the noise in the hot-leg coolant temperature sensor T_{HL}, three sensor noise levels are selected. Table 4 shows the approximate ranges of the standard deviation of sensor noise under different levels. The primary coolant RTDs in PWR plants are typically calibrated to an accuracy of $0.5^{\circ}F$ or better before installation [25]. In general, the level 1 noise is defined to be within the accuracy threshold of RTDs, while level 2 and 3 noise are beyond this accuracy threshold and hence are measurable by RTDs.

In the above context, $(3 \times 3) = 9$ classes of data sets are chosen to define each class by an RCP anomaly level and a noise level of the T_{HL} sensor. One hundred simulation runs are performed on the IRIS system for each class to generate time series data, among which 50 samples were chosen for training and the remaining samples for testing. The

Table 3. Anomaly Levels in Reactor Coolant Pump (RCP)

Anomaly Level	**RCP Overspeed ψ_{RCP} (%)**
Low	0.00 to 0.30
Medium	0.30 to 0.60
High	0.60 to 0.90

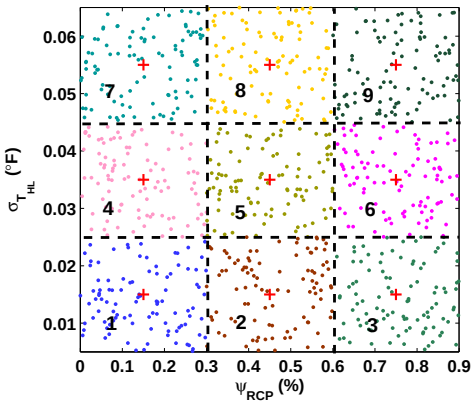


Figure 19. Original class labels for data collection

parameters, RCP overspeed percentage ψ_{RCP} and standard deviation of T_{HL} are chosen randomly from independent uniform distribution such that all of the parameter values are within the prescribed ranges given in Tables 3 and 4. Figure 19 plots the generated samples in the two-dimensional parameter space, where different classes of samples are shown in different colors and are marked within the class number. Note that the standard deviation ($\sigma_{T_{HL}}$) in Fig. 19 shows values of actual standard deviation, not as percents.

Often component degradations in nuclear plants occur on a slow-time scale, i.e., the anomaly gradually evolves over a relatively long time span. Sudden failures in plant components are disastrous and may occur only if a component accumulates sufficient amount of anomaly over a prolonged period and reaches the critical point, which justify early detection of an anomaly for enhancement of plant operational safety. However, component anomaly at the early stages of a malfunction is usually very small and is difficult to detect, especially under steady-state operations. One way to "amplify" the anomaly signature is to perturb the plant by (quasi-)periodically perturbing the turbine load such that the plant is in a transient state for a period that is sufficiently long for anomaly detection and insignificant from the perspectives of plant operation.

For each sample point in the parameter space, a time series is collected for T_{HL} sensor under persistent excitation of turbine load inputs that have truncated triangular profiles with the mean value of 99% of nominal output power, fluctuations within $\pm 1\%$ and frequency of 0.00125Hz (period $T = 800$ sec). In other words, the turbine load is fluctuating between nominal power (i.e., 335 MW) and 98% of nominal power (i.e., 328.3 MW). Figure 21 shows the profile of turbine load and a typical example of turbine output power as a result of fluctuation in turbine load.

For each experiment, the sampling frequency for data collection is 1 Hz (i.e., the inter-

Table 4. Variable Noise Levels in the the Sensor T_{HL}

Noise Level	**Standard Deviation Range** $\sigma_{T_{HL}}$ **(**$^\circ F$**)**
Level 1	0.005 to 0.025
Level 2	0.025 to 0.045
Level 3	0.045 to 0.065

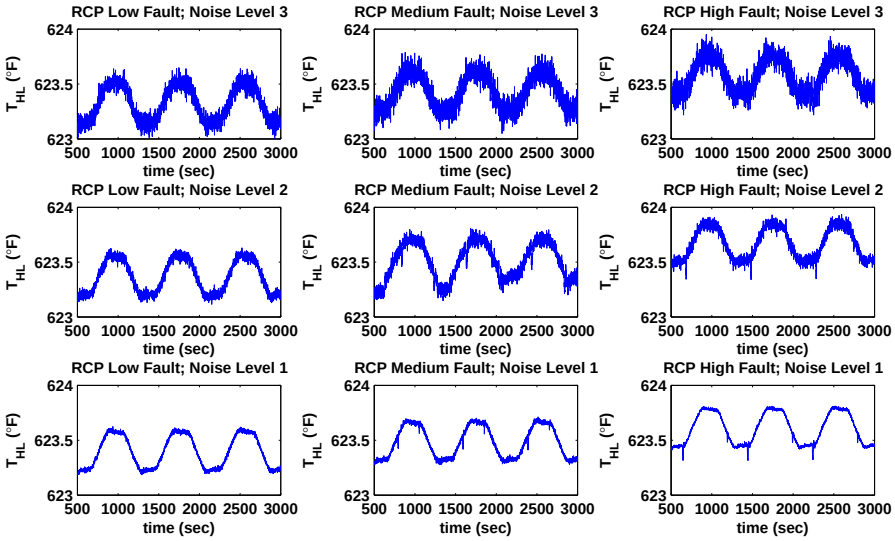


Figure 20. Representative time series data for RCP anomaly and T_{HL} degradation conditions

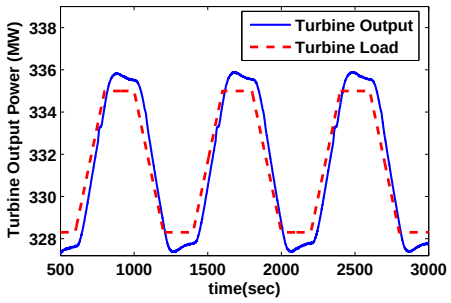


Figure 21. Profile of turbine load and turbine output response

sample time of 1 sec) and the length of the perturbation time window is 2,400 seconds, which generate 2,400 data points. The total perturbation period for each experiment is 3,000 seconds. In this study, to eliminate the possible transients, the data for the first 600 seconds have not been used so that quasi-stationarirty is achieved. Figure 20 shows representative examples of T_{HL} time series data from each of nine classes. Reduction of the perturbation period needs to be investigated for in-plant operations, which is a topic of future research.

With the objective of building a data-driven diagnostic algorithm that is robust to sensor noise (within an allowable range), a data class is defined to be only dependent on the RCP overspeed parameters. Thus, the 9 original classes are reduced to 3 classes as shown in Fig. 22. This is the final class assignment for the data set, where each class has $(50 \times 3) = 150$ training samples and $(50 \times 3) = 150$ test samples. Thus, the problem of component level anomaly detection is formulated in presence of sensor noise as a multi-class classification problem; in the present scenario, number of classes is 3.

Pertinent results are presented here for the case study of anomaly detection in the reactor

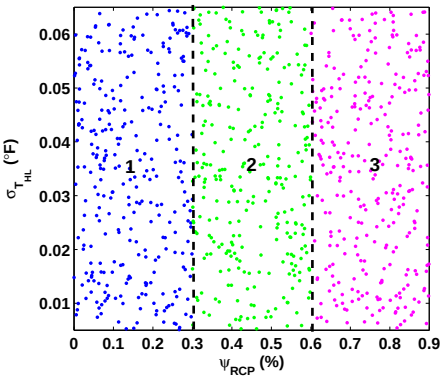


Figure 22. Revised class assignment for anomaly detection

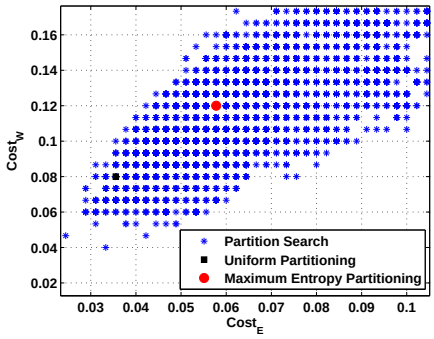


Figure 23. Two dimensional objective space for partitioning optimization

coolant pump (RCP) for comparative evaluation of the optimal partitioning-based SDF tool with those based on classical methods of partitioning as well as PCA.

At the beginning of the optimization procedure, a weighting matrix $\mathbf{W}$ needs to be defined to calculate the cost functionals $Cost_E$ and $Cost_W$ from the confusion matrix for the training data set. In this case study, $\mathbf{W}$ is defined according to the adjacency properties of classes in the parameter space, i.e., $w_{ii} = 0 \; \forall i \in \{1, 2, 3\}$, i.e. there is no penalty for correct classification. The weights are selected as: $w_{ij} = |i - j|, \; \forall i \in \{1, 2, 3\}$, i.e., given that a data sample originally from Cl_i is classified as a member of Cl_j, the penalty incurred by the classification process increases with increase in the separation of Cl_i and Cl_j in the parameter space. Then, it follows that:

$$\mathbf{W} = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}$$

The data space region Ω is divided into 40 grid cells, i.e., 39 grid boundaries excluding the boundaries of Ω. Each partitioning in the space $\mathcal{P}$ is evaluated by calculating $Cost_E$ and $Cost_W$ to identify the optimal partitioning. Figure 23 shows a relevant region of the (two-dimensional) $Cost_E$-$Cost_W$ objective space, where the elements of the space $\mathcal{P}$ are located. The Pareto front is also generated from this evaluation. The threshold ϵ, i.e., the

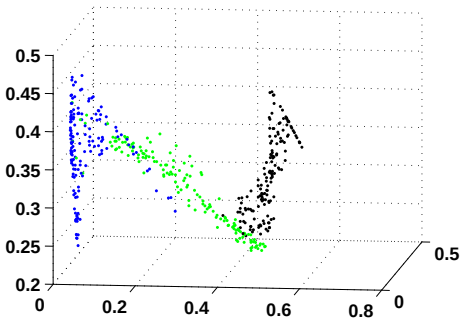


Figure 24. Feature space of the training set using optimal partitioning

maximum allowable $Cost_W$ is taken to be 0.041 in this case and the optimal partitioning (OptP) is chosen by the Neyman-Pearson criterion as discussed earlier. Thus, the user has the option of choosing a classification operating point with selected values of $Cost_E$ and $Cost_W$. The Pareto front [47] is generated after the threshold ϵ is chosen. Locations of the classical partitioning (i.e., uniform (UP) and maximum entropy (MEP)) are also plotted along with the elements of $\mathcal{P}$ in the figures for comparative evaluation.

For SDF analysis, the alphabet size is taken to be $|\Sigma| = 4$ and the depth for constructing PFSA is taken to be $D = 1$. Features are classified by a k-NN classifier using the Euclidean distance metric. Figure 24 shows locations of the training features in the three-dimensional plot using first three linearly independent elements of the feature vectors obtained by using the chosen optimal partitioning, OptP. Note, only $(|\Sigma|-1)$ out of its $|\Sigma|$ elements of a feature vector are linearly independent, because a training feature vector, $\mathbf{p}$ is also a probability vector, i.e., the sum of its elements is constrained to be equal to 1. The class separability is retained by the feature extraction (partitioning) process even after compressing a time series (with 2,400 data points) into 3 numbers.

For comparison purpose, classical partitioning schemes, such as, Uniform Partitioning (UP) and Maximum Entropy Partitioning (MEP) are also used with the same alphabet size, $|\Sigma| = 4$. Figures 25 and 26 show the location of each training time series in the three dimensional (using first three linearly independent elements of the feature vectors) feature space plot using UP and MEP, respectively.

Finally, the confusion matrices for the SDF-based methods (OptP, UP and MEP) with k-NN on the test data set are given by $\mathbf{C}_{test}^{OptP+kNN}$, $\mathbf{C}_{test}^{UP+kNN}$ and $\mathbf{C}_{test}^{MEP+kNN}$, respectively.

$$\mathbf{C}_{test}^{OptP+kNN} = \begin{pmatrix} 142 & 8 & 0 \\ 14 & 136 & 0 \\ 0 & 4 & 146 \end{pmatrix}$$

$$\mathbf{C}_{test}^{UP+kNN} = \begin{pmatrix} 145 & 5 & 0 \\ 17 & 132 & 1 \\ 0 & 6 & 144 \end{pmatrix}$$

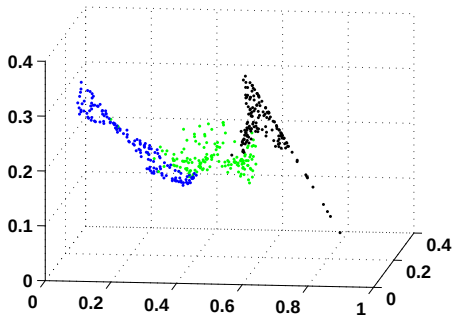


Figure 25. Feature space of training set – uniform partitioning (UP)

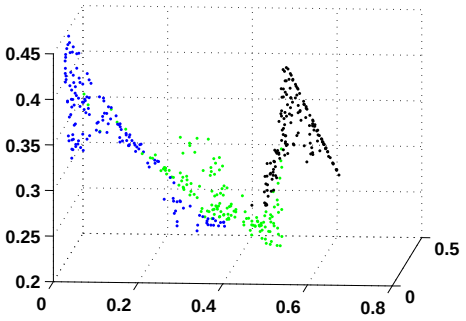


Figure 26. Feature space of training set – maximum entropy partitioning (MEP)

$$\mathbf{C}_{test}^{MEP+kNN} = \begin{pmatrix} 143 & 7 & 0 \\ 18 & 132 & 0 \\ 0 & 5 & 145 \end{pmatrix}$$

For comparative evaluation, the data sets are analyzed using other common pattern recognition tools [4][40]. In this case study, principal component analysis (PCA) is used as the feature extraction tool, while k-NN, linear discriminant analysis (LDA) and least squares algorithm (LS) are used as the pattern classifiers. For PCA anomaly detection, $M = 450$, $N = 2400$, and by choosing $\eta = 0.90$, the corresponding number of largest eigenvalues turns out to be $m = 20$. The confusion matrices for PCA with k-NN, LDA and LS are given by $\mathbf{C}_{test}^{PCA+kNN}$, $\mathbf{C}_{test}^{PCA+LDA}$ and $\mathbf{C}_{test}^{PCA+LS}$, respectively.

$$\mathbf{C}_{test}^{PCA+kNN} = \begin{pmatrix} 149 & 1 & 0 \\ 40 & 110 & 0 \\ 0 & 4 & 146 \end{pmatrix}$$

$$\mathbf{C}_{test}^{PCA+LDA} = \begin{pmatrix} 136 & 14 & 0 \\ 18 & 126 & 6 \\ 0 & 3 & 147 \end{pmatrix}$$

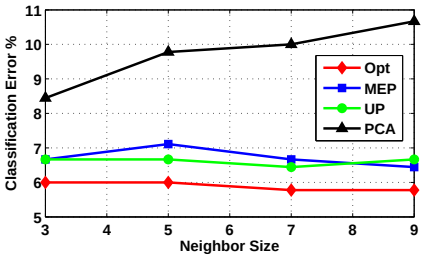


Figure 27. Classification error vs. neighbor size in k-NN classifiers

$$\mathbf{C}_{test}^{PCA+LS} = \begin{pmatrix} 145 & 5 & 0 \\ 27 & 115 & 8 \\ 0 & 0 & 150 \end{pmatrix}$$

Figure 27 shows how the neighbor size in the k-NN classifiers affect the classification performance, and the performances of Opt, MEP, UP and PCA with different neighbor size k are compared. Only odd values of neighbor size are shown because when using an even value for k, it might be necessary to break a tie in the number of nearest neighbors by selecting a random tiebreaker or using the nearest neighbor among the tied groups. It is seen in Fig. 27 the classification performance of SDF-based methods is consistently better than that of PCA-based method and almost independent of the neighbor size, whereas the classification error of PCA increases with the number of neighbor size. This results show that the feature vectors extracted by SDF retain better separability among classes than those extracted by PCA.

Table 5. Comparison of Classification Performances of Different Methods on Test-data set (50×9 samples)

Methods	$CostE$	$CostW$	Classification Error %
OptP+kNN	0.0578	0.0933	5.78
UP+kNN	0.0644	0.1200	6.44
MEP+kNN	0.0667	0.1200	6.67
PCA+kNN	0.1000	0.2667	10.00
PCA+LDA	0.0911	0.1200	9.11
PCA+LS	0.0889	0.1800	8.89

Table 5 presents the comparison of the classification error related costs for SDF-based methods and PCA-based methods on the test data set. The observation made from these results indicate that the classification performance of SDF-based methods are superior to that of the PCA-based methods. The classification performance of MEP and UP are close, and UP performs slightly better in this case study. The costs due to worst-case classification error and expected classification error are reduced compared to that of the uniform partitioning scheme by optimizing the partitioning process over a representative training set.

It is noted that, for some problems, the classical partitioning schemes may perform similar to the optimal one. Therefore, the optimization procedure may also be used to evaluate the capability of any partitioning scheme towards achieving a better classification rate. The evaluation can be performed by using a part of the labeled training data set as the validation set. Although the construction of the cost functions allow solutions of problems with a large number of classes, its upper limit could be constrained by the alphabet size used for data partitioning which determines the dimension of the feature space.

4. Symbolic Dynamic Analysis of Short-length Transient Time-series Motifs

Many classification algorithms are not well-suited for applications to nonlinear dynamical systems [48][49][50]. This phenomenon has led to departure from the conventional continuous-domain modeling towards a formal language-theoretic paradigm in the symbolic domain [51]. The theory of symbolic dynamic filtering (SDF) has been developed as a tool for anomaly detection [15][12] and also for feature extraction [52] in pattern classification problems [1].

A symbol string is obtained from the output of the dynamical system by partitioning (also called quantization) of the time-series data. Thereafter, a probabilistic finite state automaton (PFSA) is constructed from the (finite-length) symbol sequence via one of the construction algorithms (e.g., [15], [53], and [54]). Due to the quasi-stationarity assumption in SDF, it may not be feasible to obtain sufficiently long strings of symbols in both training and testing phases of classification. Therefore, the estimated parameters of the resulting PFSA model may not be precise.

The goal of this section is to identify the hidden structure of the symbol string and to construct a Bayesian classifier for identification of the probability morph matrices of PFSA based on the data length in both training and testing phases of transient time-series motifs. In this context, Dirichlet and multinomial distributions have been used to construct the *a priori* and *a posteriori* models of uncertainties, respectively. The algorithms are formulated by quantitatively incorporating the effects of finite-length symbol strings in both training and testing phases of pattern classification. In this setting, pertinent information (e.g., probability morph matrix and state transition function) derived from a PFSA model serves as a feature vector even though the PFSA representations may have dissimilar structures. The Proposed algorithm is validated on an industrial application which is as follows: Transient Fault Detection in Aircraft Gas Turbine Engine.

4.1. Theory

A symbol string is obtained from the output of system under consideration by partitioning, also called quantization, of the time-series data. Thereafter, a probabilistic finite state automaton (PFSA) model is developed from the (finite-length) symbol sequence [15] as explained in the subsection 1.1.. This section constructs a Bayesian classifier for identification of the probability morph matrices of PFSA based on the transient data in both training and testing phases. The Dirichlet and multinomial distributions have been used to

construct the *a priori* and *a posteriori* models of uncertainties, respectively. This formulation quantitatively incorporates the effects of finite-length symbol strings in both training and testing phases of pattern classification.

4.1.1. Partitioning of time series data

The sensor time series is encoded by data partitioning in the range of the signal [55][12], where the conversion to symbol strings is achieved by substituting each (real-valued) data point in the time series by a symbol corresponding to the region (i.e., interval) within which the data point lies. This step enables transformation of the sensory information from the continuous domain to the symbolic domain; in other words, the sensor data at each sampling instant is replaced by a symbol. Sarkar et al. [55][56] have reported details of data partitioning and its usage for fault detection in gas turbine engines.

4.1.2. Modeling via Probabilistic Finite State Automaton

The symbolic sequence is modeled as a probabilistic finite state automaton (PFSA) that is constructed as a tuple $G \triangleq (Q, \Sigma, \delta, \Pi)$, where the alphabet Σ is a nonempty finite set of symbols and the set of states Q is constrained to be nonempty and finite. To simplify the PFSA model construction, this section considers only a class of PFSA, known as D-Markov machines [15], where the states are strings of the D past symbols; the positive integer D is called the depth of the machine and the number of states $|Q| \leq |\Sigma|^D$. Given the previous state and an observed symbol, the state transition function $\delta : Q \times \Sigma \rightarrow Q$ yields the new state. In addition, the morph function $\pi : Q \times \Sigma \rightarrow [0, 1]$ is an output mapping that satisfies the condition: $\sum_{\sigma \in \Sigma} \pi(q, \sigma) = 1$ for all $q \in Q$. The morph function π has a matrix representation Π, called the (probability) morph matrix of dimension $(|Q| \times |\Sigma|)$. Each row sum of Π is equal to 1 and each matrix element Π_{ij} is strictly positive due to the finite length constraint of time series from which PFSA models are constructed [57].

Remark 4..1. *The PFSA $G \triangleq (Q, \Sigma, \delta, \Pi)$ is not dependent on the initial state $q_0 \in Q$ due to the assumption of quasi-stationarity of the observed sensor data.*

4.1.3. The Online Classification Problem

Let there be K symbolic systems (i.e., classes) of interest, denoted by $C_1, C_2, \ldots, C_K$, over the same alphabet Σ. Each class C_i is modeled by an ergodic (equivalently, irreducible) PFSA $G^i = (Q^i, \Sigma, \delta^i, \Pi^i)$, where $i = 1, 2 \ldots, K$.

During the training phase, a symbol string $S^i \triangleq s^i_1 s^i_2 \ldots s^i_{N_i}$ is generated from each class C_i. The state transition function δ and the set of states Q of the D-Markov machine are fixed by choosing an appropriate depth D. Thus, Π^i's become the only unknowns and could be selected as the feature vectors for the purpose of classification. The distribution of the morph matrix Π^i is computed in the training phase from the finite length symbol sequences for each class.

In the testing phase, let another symbol string $\widetilde{S}$ be obtained from a sensor time series data. Then, the task is to determine which class this observed symbol string $\widetilde{S}$ belongs to. While the previous work [15][12] has aimed at identification of a PFSA from a given

symbol string, the objective of this section is to imbed the uncertainties due to the finite length of the symbol string in the identification algorithm that would influence the final classification decision.

In the training phase, each row of Π^i is treated as a random vector. Let the m^{th} row of Π^i be denoted as Π^i_m and the n^{th} element of the m^{th} row as $\Pi^i_{mn} > 0$ and $\sum_{n=1}^{|\Sigma|} \Pi^i_{mn} = 1$. The *a priori* probability density function $f_{\Pi^i_m|S^i}$ of the random row-vector Π^i_m, conditioned on a symbol string S^i, follows the Dirichlet distribution [58] [59] as described below.

$$f_{\Pi^i_m|S^i}(\boldsymbol{\theta}^i_m|S^i) = \frac{1}{B(\boldsymbol{\alpha}^i_m)} \prod_{n=1}^{|\Sigma|} (\theta^i_{mn})^{\alpha^i_{mn}-1} \tag{61}$$

where $\boldsymbol{\theta}^i_m$ is a realization of the random vector Π^i_m, namely,

$$\boldsymbol{\theta}^i_m = \begin{bmatrix} \theta^i_{m1} & \theta^i_{m2} & \cdots & \theta^i_{m|\Sigma|} \end{bmatrix}$$

and the normalizing constant is

$$B(\boldsymbol{\alpha}^i_m) \triangleq \frac{\prod_{n=1}^{|\Sigma|} \Gamma(\alpha^i_{mn})}{\Gamma(\sum_{n=1}^{|\Sigma|} \alpha^i_{mn})} \tag{62}$$

where $\boldsymbol{\alpha}^i_m \triangleq \begin{bmatrix} \alpha^i_{m1} \alpha^i_{m2} \cdots \alpha^i_{m|\Sigma|} \end{bmatrix}$ with $\alpha^i_{mn} = N^i_{mn} + 1$ and N^i_{mn} is the number of times the symbol σ_n in S^i is emanated from the state q_m, i.e.,

$$N^i_{mn} \triangleq \big|\{(s^i_k, v^i_k) : s^i_k = \sigma_n, v^i_k = q_m\}\big| \tag{63}$$

where s^i_k is the k^{th} symbol in S^i and v^i_k is the k^{th} state as derived from the symbolic sequence S^i. Recall that a state is defined as a string of D past symbols. Then, the number of occurrence of the state q_m in the state sequence is given by $N^i_m \triangleq \sum_{n=1}^{|\Sigma|} N^i_{mn}$. It follows from Eq. (62) that

$$B(\boldsymbol{\alpha}^i_m) = \frac{\prod_{n=1}^{|\Sigma|} \Gamma(N^i_{mn}+1)}{\Gamma(\sum_{n=1}^{|\Sigma|} N^i_{mn} + |\Sigma|)} = \frac{\prod_{n=1}^{|\Sigma|} (N^i_{mn})!}{(N^i_m + |\Sigma| - 1)!} \tag{64}$$

by use of the relation $\Gamma(n) = (n-1)!\ \ \forall n \in \mathbb{N}_1$.

By the Markov property of the PFSA G^i, the $(1 \times |\Sigma|)$ row-vectors, $\{\Pi^i_m\}, m = 1, \ldots |Q|$, are statistically independent of each other. Therefore, it follows from Eqs. (61) and (64) that the *a priori* joint density $f_{\Pi^i|S^i}$ of the probability morph matrix Π^i, conditioned on the symbol string S^i, is given as

$$\begin{aligned} f_{\Pi^i|S^i}(\boldsymbol{\theta}^i|S^i) &= \prod_{m=1}^{|Q|} f_{\Pi^i_m|S^i}\left(\boldsymbol{\theta}^i_m|S^i\right) \\ &= \prod_{m=1}^{|Q|} \big(N^i_m + |\Sigma| - 1\big)! \prod_{n=1}^{|\Sigma|} \frac{(\boldsymbol{\theta}^i_m)^{N^i_{mn}}}{(N^i_{mn})!} \end{aligned} \tag{65}$$

where $\boldsymbol{\theta}^i = \left[(\boldsymbol{\theta}_1^i)^T\ (\boldsymbol{\theta}_2^i)^T\ \cdots\ (\boldsymbol{\theta}_{|q|}^i)^T\right] \in [0,1]^{|Q|\times|\Sigma|}$

In the testing phase, the probability of observing a symbol string $\widetilde{S}$ belonging to a particular class of PFSA $(Q, \Sigma, \delta, \Pi^i)$ is a product of independent multinomial distribution [60] given that the exact morph matrix Π^i is known.

$$\begin{aligned}
&\Pr\left(\widetilde{S}|Q,\delta,\Pi^i\right)\\
&= \prod_{m=1}^{|Q|}(\widetilde{N}_m)! \prod_{n=1}^{|\Sigma|}\frac{\left(\Pi_{mn}^i\right)^{\widetilde{N}_{mn}}}{(\widetilde{N}_{mn})!} &(66)\\
&\triangleq \Pr\left(\widetilde{S}|\Pi^i\right) \text{ as } Q \text{ and } \delta \text{ are kept invariant} &(67)
\end{aligned}$$

Similar to N_{mn}^i defined earlier for S^i, $\widetilde{N}_{mn}$ is the number of times the symbol σ_n is emanated from the state $q_m \in Q$ in the symbol string $\widetilde{S}$ in the testing phase, i.e.,

$$\widetilde{N}_{mn} \triangleq \left|\{(\tilde{s}_k, \tilde{v}_k) : \tilde{s}_k = \sigma_n, \tilde{v}_k = q_m\}\right| \tag{68}$$

where $\tilde{s}_k$ is the k^{th} symbol in the string $\widetilde{S}$ and $\tilde{v}_k$ is the k^{th} state derived from $\widetilde{S}$. It is noted that $\widetilde{N}_m \triangleq \sum_{n=1}^{|\Sigma|}\widetilde{N}_{mn}$.

The results, derived in the training and testing phases, are now combined. Given a symbol string S^i in the training phase, the probability of observing a symbol string $\widetilde{S}$ in the testing phase is obtained as follows.

$$\begin{aligned}
\Pr(\widetilde{S}|S^i) &= \int\cdots\int \Pr\left(\widetilde{S}|\Pi^i = \boldsymbol{\theta}^i\right) f^i_{\Pi^i|S^i}(\boldsymbol{\theta}^i|S^i)d\boldsymbol{\theta}^i\\
&= \int\cdots\int \left[\prod_{m=1}^{|Q|}(\widetilde{N}_m)! \prod_{n=1}^{|\Sigma|}\frac{\left(\theta_{mn}^i\right)^{\widetilde{N}_{mn}}}{(\widetilde{N}_{mn})!}\right]\\
&\times \prod_{m=1}^{|Q|}\left[\left(N_m^i + |\Sigma| - 1\right)! \prod_{n=1}^{|\Sigma|}\frac{(\theta_{mn}^i)^{N_{mn}^i}}{(N_{mn}^i)!}\, d\theta_{mn}^i\right]\\
&= \prod_{m=1}^{|Q|}(\widetilde{N}_m)!\left(N_m^i + |\Sigma| - 1\right)!\\
&\times \frac{\int\cdots\int\prod_{n=1}^{|\Sigma|}(\theta_{mn}^i)^{\widetilde{N}_{mn}+N_{mn}^i}d\theta_{mn}^i}{\prod_{n=1}^{|\Sigma|}(\widetilde{N}_{mn})!(N_{mn}^i)!}
\end{aligned} \tag{69}$$

The integrand in Eq. (69) is the density function for the Dirichlet distribution up to the multiplication of a constant. Hence, it follows from Eq. (64) that

$$\begin{aligned}
&\int\cdots\int\prod_{n=1}^{|\Sigma|}(\theta_{mn}^i)^{\widetilde{N}_{mn}+N_{mn}^i}\, d(\theta_{mn}^i)\\
&= \frac{\prod_{n=1}^{|\Sigma|}(\widetilde{N}_{mn}+N_{mn}^i)!}{\left(\widetilde{N}_m + N_m^i + |\Sigma| - 1\right)!}
\end{aligned}$$

Then, it follows from Eq. (69) that

$$\Pr(\widetilde{S}|S^i) = \prod_{m=1}^{|Q|} \frac{(\widetilde{N}_m)!\,\big(N_m^i + |\Sigma| - 1\big)!}{\Big(\widetilde{N}_m + N_m^i + |\Sigma| - 1\Big)!} \times \prod_{n=1}^{|\Sigma|} \frac{(\widetilde{N}_{mn} + N_{mn}^i)!}{(\widetilde{N}_{mn})!(N_{mn}^i)!} \tag{70}$$

It might be easier to compute the logarithm of $\Pr(\widetilde{S}|S^i)$ by using Stirling's approximation formula $\log(n!) \approx n\log(n) - n$ [2] because both N^i and $\widetilde{N}$ would be large numbers. The posterior probability of a symbol string S belonging to the class C_i is denoted as $\Pr(C_i|\widetilde{S})$ and is given as

$$\Pr(C_i|\widetilde{S}) = \frac{\Pr(\widetilde{S}|S^i)\Pr(C_i)}{\sum_{j=1}^{K}\Pr(\widetilde{S}|S^j)\Pr(C_j)}, \;\; i = 1, 2, \cdots, K \tag{71}$$

where $\Pr(C_i)$ is the known prior distribution of the class C_i. Then, the classification decision is made as follows.

$$\begin{aligned} D_{class} &= \arg\max_i \Pr(C_i|\widetilde{S}) \\ &= \arg\max_i \Big(\Pr(\widetilde{S}|S^i)\Pr(C_i)\Big) \end{aligned} \tag{72}$$

Algorithms 1 and 2 respectively summarize the training and testing phases for classification of time-series data.

Algorithm 1 Training

Input: Time-series data, one for each of the K classes
User defined Input: Symbol alphabet Σ, depth D of PFSA
the symbols are defined $\sigma_1, \sigma_2, ..., \sigma_{|\Sigma|}$ and states are defined as $q_1, q_2, ..., q_{|\Sigma|^D}$
Output: N_{mn}^i, where $i = 1, ..., K$, $n = 1, ..., |\Sigma|$ and $m = 1, ..., |\Sigma|^D$
Initialize: All $N_{mn}^i = 0$.
for all $i \in \{1, 2, ..., K\}$ **do**
Symbolize time-series data for class i to obtain a symbol sequence S^i
Obtain N_{mn}^i, the number of times the symbol σ_n in S^i is observed immediately after the state q_m
end for
return N_{mn}^i, where $i = 1, ..., K$, $n = 1, ..., |\Sigma|$ and $m = 1, ..., |\Sigma|^D$

The concept of the online classifier can be clarified using a simple example [61] as follows. Figure 28 shows two single-state PFSA with the alphabet $\Sigma = \{1, 2, 3\}$, which

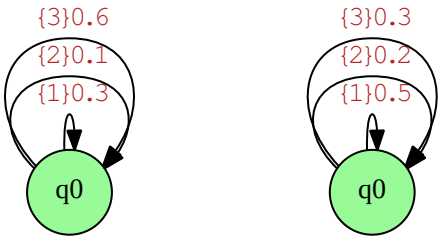
Algorithm 2 Testing

Input: Test time-series data, and output of training N^i_{mn}
User defined Input: Prior probability of each class $Pr(C_i)$,
symbol alphabet Σ, and depth D of PFSA,
the symbols are defined as $\sigma_1, \sigma_2, ..., \sigma_{|\Sigma|}$, and states are defined as $q_1, q_2, ..., q_{|\Sigma|^D}$
Output: Posterior probability of each class.
Initialize: All $\tilde{N}^i_{mn} = 0$.
Symbolize test time-series data to obtain a symbol sequence $\tilde{S}^i$
Obtain $\tilde{N}^i_{mn}$, the number of times the symbol σ_n in $\tilde{S}$ is observed immediately after the state q_m is reached
for all $i \in \{1, 2, ..., K\}$ **do**
Evaluate Pr($\tilde{S}|S^i$) using Eq. (70)
end for
for all $i \in 1, 2, ..., K$ **do**
Evaluate Pr($C_i|\tilde{S}$) using Eq. (71)
end for
return Pr($C_i|\tilde{S}$), $i \in \{1, 2, ..., K\}$

have identical algebraic structures, but they differ in their morph probabilities; these PFSA belong to the respective classes, C_1 and C_2. Since both PSFA have only one state, each symbol in a string is independent and identically distributed. Given an observed symbol string $\widetilde{S}$, the task is to identify one of the two classes to which $\widetilde{S}$ belongs, i.e., to select one of the two PFSA that would more likely generate the symbol string $\widetilde{S}$. In this example, two training symbol strings S^1 and S^2 are chosen, one from each class. In the testing phase, the quantities $\widetilde{N}^1_{01}$, $\widetilde{N}^1_{02}$, $\widetilde{N}^1_{03}$ are obtained following Eq. (68), which are essentially the frequency counts of the symbols 1, 2, and 3, respectively. Let η_1, η_2, and η_3 be the normalized frequency counts obtained by $\eta_k \triangleq \frac{\widetilde{N}^1_{0k}}{\widetilde{N}^1_{01}+\widetilde{N}^1_{02}+\widetilde{N}^1_{03}}$, $k = 1, 2, 3$. Two classes are generated on the simplex plane in Fig. 29 that shows how the classification boundary changes as the length of a symbol string in the testing phase is increased.

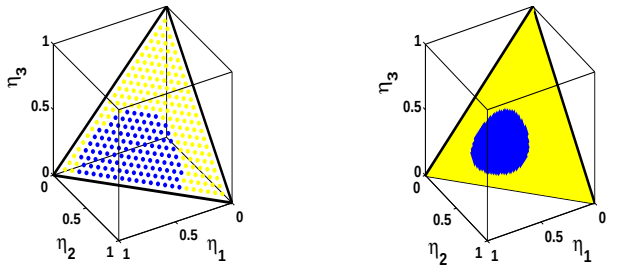
4.2. Transient Fault Detection in Aircraft Gas Turbine Engine

Performance monitoring of aircraft gas turbine engines is typically conducted on quasi-stationary steady-state data collected during cruise conditions. This issue has been addressed by several researchers by using a variety of analytical tools. For example, Lipowsky et al. [62] made use of Bayesian forecasting for change detection and performance analysis of gas turbine engines. Guruprakash and Ganguli [63] reported optimally weighted recursive median filtering for gas turbine diagnostics from noisy signals. However, transient operational data (e.g., from takeoff, climb, or landing) can be gainfully utilized for early detection of incipient faults. Usually engines operate under much higher stress and temperature conditions under transient operations compared to those under steady-state conditions. Signatures of certain incipient engine faults (e.g., bearing faults, controller miss-scheduling,



(a) Class C_1 (b) Class C_2

Figure 28. Two one-state PFSA with different morph probabilities



(a) Testing data length = 25 (b) Testing data length = 80

Figure 29. Both subfigures (a) and (b) show the classification boundaries for the two classes C_1 (in light shade) and C_2 (in dark shade) for different lengths of testing string $\widetilde{S}$. The training strings S^1 and S^2 for classes C_1 and C_2 are obtained from the two machines and have lengths 30 and 80, respectively.

and starter system faults) tend to magnify during the transient conditions [64], and appropriate usage of transient data could enhance the probability of fault detection [65]. Along this line, several model-based and data-driven fault diagnosis methods have been proposed by making use of transient data. For example, Wang et al. [66] developed a neural network-based fault diagnosis method for automotive transient operations, and Surendar and Ganguli [67] used an adaptive Myriad filter to improve the quality of transient operations for gas turbine engines. However, model-based diagnostics may suffer from inaccuracy and loss of robustness due to low reliability of the transient models. This problem is partially circumvented through usage of data-driven diagnostics; for example, Menon et al. [68] used hidden Markov models (HMMs) for transient analysis of gas turbine engines.

A recently developed data-driven technique, called the symbolic dynamic filtering (SDF) [15], has been shown to yield superior performance in terms of early detection of anomalies and robustness to measurement noise in comparison to other techniques such as Principal Component Analysis (PCA), Neural Networks (NN) and Bayesian techniques [6]. Recently, in a two-part paper [69][70], an SDF-based algorithm for detection and isolation of engine subsystem faults (specifically, faults that cause efficiency degradation in engine components) has been reported and an extension of that work to estimate simultaneously

occurring multiple component-level faults has been presented in [56]. Furthermore, an optimized feature extraction technique has been developed under the same semantic framework in [55]. However, all of the above studies were conducted on steady-state cruise flight data that conform with the assumption of quasi-stationarity made in SDF. Due to this assumption, SDF may not be able to adequately handle transient data that are usually of limited length. To handle this limitation the above mentioned algorithm is applied using the *Transient Test-case Generator* [71] of the Commercial Modular Aero Propulsion System Simulation (C-MAPSS) test bed, developed by NASA to detect transient fault.

4.2.1. Fault Injection in C-MAPSS Test Bed

A stochastic damage model has been developed and incorporated in the C-MAPSS *Transient Test-case Generator* [71] (developed by NASA), based on the experimental data for trending the natural deterioration of the engine components as explained in section 2.. Although injection of faults is described in the transient test-case generator code [71], it is explained in this section for completeness. For all five rotating components, faults exhibit random magnitudes (F_m), and a random health parameter ratio (HPR). While F_m and HPR directly determines the change in efficiency health parameter $\psi(C)$ of a component C, a change in the flow health parameter ζ_C is determined by HPR for a given perturbation in ψ_C. Formally, the following two relations are used.

$$\delta_{\psi_C} = -\frac{F_m}{\sqrt{1+HPR^2}} \text{ and } \delta_{\zeta_C} = \delta_{\psi_C} \cdot HPR \tag{73}$$

where δ_{ψ_C} and δ_{ζ_C} denote the changes in ψ_C and ζ_C respectively.

In the case study, fault magnitude (F_m) follows a random uniform distribution ranging from 1 to 7. Health parameter ratios (HPR) for Fan, LPC, and HPC are uniformly distributed between 1.0 and 2.0, whereas HPR s for HPT and LPT are uniformly distributed between 0.5 to 1.0. The changes in health parameters occur from certain base values of ψ_C and ζ_C.

4.3. Performance of Transient Fault Classifier

This section presents the simulation results of symbolic analysis of transient time series and incipient fault detection on the C-MAPSS test bed. The goal here is to demonstrate how the binary faults (i.e., $K = 2$ in Algorithms 1 and 2) are detected and identified at an early stage in time-critical operations like engine health management. To establish the relationship between detection time and detection accuracy, numerous simulation experiments have been performed during the "takeoff" operation, where the Mach number is varied from 0 to 0.24 in 60 seconds keeping the altitude at zero (i.e., sea level) and TRA at 80%. Faulty operations in three different components, Fan, LPC and HPT, are simulated along with the ideal engine condition. The HPT exit temperature (T_{48}) sensor (that captures the above failure signatures in the gas path) is chosen to provide the transient response. The rationale behind choosing the T_{48} sensor is attributed to the following physical facts: (i) T_{48} is placed between HPT and LPT, and (ii) LPT is mechanically connected to the Fan and LPC via the outer shaft. The sampling frequency of the T_{48} sensor data is 66.7 Hz. For the purpose of

training, the duration for each simulation run is chosen to be 60 sec (i.e., a time series of length ~ 4000) and the transient data set for each fault class is constructed by concatenating 50 such blocks of time series data that were individually generated on the simulation test bed under similar transient operating conditions. It is noted that these blocks of data are statistically similar but they are not identical.

The next step is to partition the data sets to generate respective symbol strings. The range of the time series is partitioned into 5 intervals (i.e., the alphabet size $|\Sigma| = 5$), where each segment of the partitioning corresponds to a distinct symbol in the alphabet Σ. The training phase commences after the symbol strings are obtained for each of the four classes, i.e., one nominal and three faulty conditions. In this application, it suffices to choose the depth $D = 1$ [15], which implies that the probability of generation of a future symbol depends only on the last symbol, and hence the set of states is isomorphic to the symbol alphabet (i.e., $Q \equiv \Sigma$). For each class C_i, the parameters N^i_{mn} are obtained by counting the number of times the symbol σ_n is emitted from state q_m.

The testing phase starts with a new time series from one of the classes, where the symbol sequence is generated by using the same alphabet and partitioning as in the training phase. Following Eq. (71), the posterior probability of each class is calculated as a function of the length of the testing data set. Figure 30 shows the posterior probability of each class as a function of the length of the observed test data. It is seen that the observed sequence is correctly identified to belong to the class of LPC faults as the posterior probability of the corresponding class approaches one, while it approaches zero for each of the remaining classes (i.e., the other two component faults and nominal condition). By repeating the same classification technique on 50 new test runs of LPC fault, it is observed that a data length of 500 is sufficient to detect the fault with a reasonable confidence. For other two component faults, the posterior probability for the correct fault class approaches unity within the data length of 50, as seen in Fig. 31. The rationale is that the fault signatures for fan and HPT in T_{48} response are dominant, even if they are incipient.

In a hierarchical fault detection and isolation strategy, at a low resolution level, the goal is to identify the faulty component (e.g., Fan, LPC or HPT). Once the fault is located, the next step is to quantify the severity of the located fault. The symbolic transient fault detection technique can be extended to classify different levels of fault in single components of a gas turbine engine. For verification, samples of nominal data are injected with a low-level fan fault, where the fault magnitude (F_m) follows a random uniform distribution ranging from 1 to 3. The remaining samples of nominal data are injected with a high-level fan fault of magnitude (F_m) within the range of 5 to 7. In this case, the alphabet size is chosen to be $|\Sigma| = 6$ to obtain better class separability. Figure 31 shows that the posterior probability for a high fan fault saturates to the value of 1 within a data length of 70, which is equivalent to ~ 1 sec. (Note: The oscillations in posterior probabilities in Fig. 31 are due to randomness of the estimated parameters for data lengths smaller than 50. But when the test case is a low-fan fault (see Fig. 32), the posterior probability of a high-level fan fault remains dominant until the data length reaches ~ 400. This would result in a false classification as the posterior probability for the true class reaches 1 monotonically at around the data length of 1000 at, about 15 seconds, as seen in Fig. 32. This result agrees well with the intuition that the detection time increases with smaller the fault levels.

To examine the performance of symbolic transient fault detection, a family of receiver

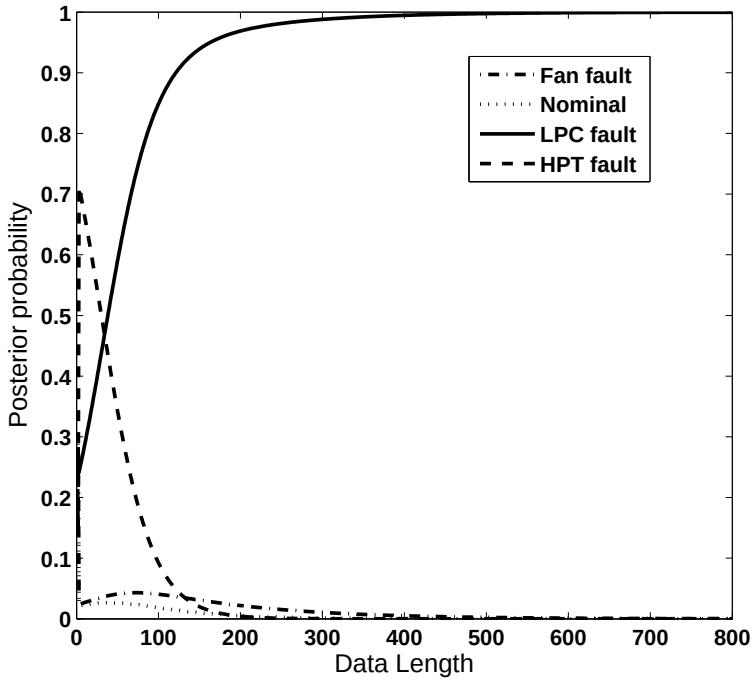


Figure 30. Detection in a multi-fault framework (Ground truth: a faulty LPC)

operating characteristics (ROC) is constructed for different lengths of test data. A binary classification scenario is constructed, which consists of two classes: the nominal engine condition belonging to the class C_1, and a faulty fan condition belonging to the class C_2. The training data length is the same as described in the previous simulation runs. The general classification rule [72] in a symbol string $\widetilde{S}$ is given by

$$\frac{\Pr(\widetilde{S}|C_1)}{\Pr(\widetilde{S}|C_2)} \underset{C_2}{\overset{C_1}{\gtrless}} \lambda \tag{74}$$

where the threshold parameter λ is varied to generate the ROC curves. For the binary classification problem at hand, an ROC curve provides the trade-off between the probability of detection $P_D = \Pr\{\text{decide } C_2|C_2 \text{ is true}\}$ and the probability of false alarms $P_F = \Pr\{\text{decide } C_2|C_1 \text{ is true}\}$. Figure 33 exhibits a family of ROC curves for the proposed fault detection technique with varying lengths of test data. For each ROC curve, λ is varied between 0.01 and 2 with steps of 0.01. It is observed that the ROC curves yield improved performance (i.e., movement toward the top left corner) progressively as the test data length is increased from $N_{test} = 20$ to $N_{test} = 200$. Thus, based on a family of such ROC curves, a good combination of P_D and N_{test} is obtained for a given P_F.

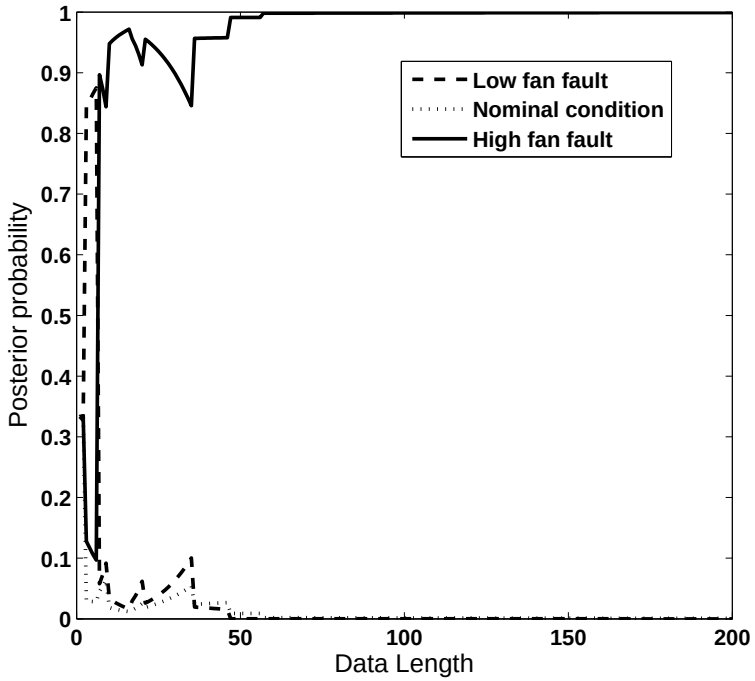


Figure 31. High-level fan fault detection (Ground truth: A high-level fan fault)

5. Semantic Framework for Multi-sensor Data Interpretation and Fusion

Sensor fusion has been one of the focused topics in the data analysis of distributed physical processes, where the individual sensory information is often used to reveal the underlying process dynamics and to identify potential changes therein. Distributed physical processes are usually equipped with multiple sensors having (possibly) different modalities over a sensor network to accommodate both model-based and data-driven diagnostics & control. The ensemble of distributed and heterogeneous information needs to be fused to generate accurate inferences about the states of critical systems in real time. Various sensor fusion methods have been reported in the literature to address the fault detection & classification problems; examples are linear and nonlinear filtering, adaptive model reference methodologies and neural network-based estimation schemes.

Researchers have used multi-layer perception [73] and radial basis function [74] configurations of neural networks for detection & classification of plant component, sensor and actuator faults [75]. Similarly, principal component analysis [76] and kernel regression [77] techniques have been proposed for data-driven pattern classification. These approaches address nonlinear dynamics as well as scaling and data alignment issues. However, the effectiveness of data-driven techniques may often degrade rapidly for extrapolation of non-stationary data in the presence of multiplicative noise. Some of the above difficulties can be alleviated to a certain extent by simplifying approximations along with a combination of

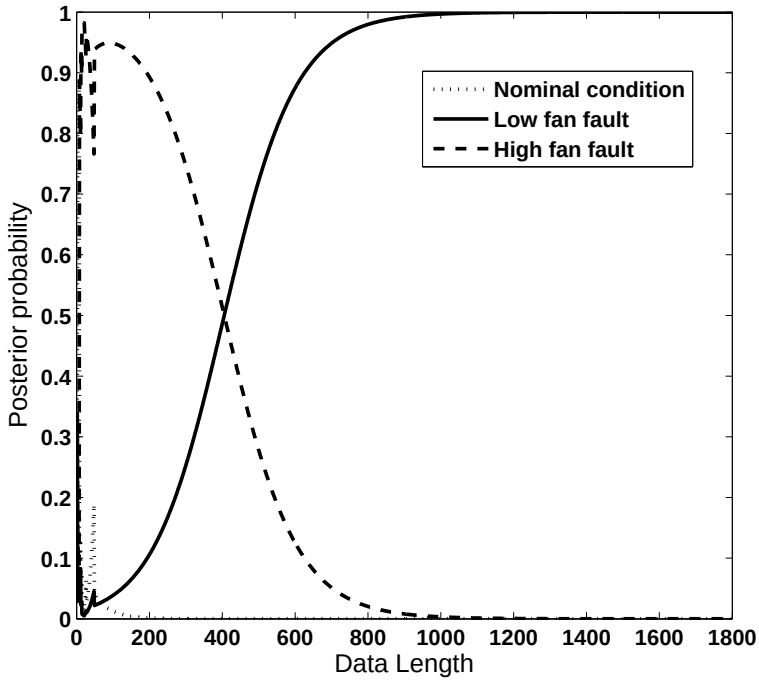


Figure 32. Low-level fan fault detection (Ground truth: A low-level fan fault)

model-based and data-driven analysis as discussed below.

Robust filtering techniques have been developed to generate reliable estimations from sensor signals, because sensor time series data are always noise-contaminated to some extent [78][79]. Recent literature has also reported Monte Carlo Markov chain (MCMC) techniques (e.g., particle filtering [80] and sigma point techniques [81]) that yield numerical solutions to Bayesian state estimation problems and have been applied to diverse nonlinear dynamical systems [82]. The performance and quality of estimation largely depend on the modeling accuracy which is the central problem in the filtering approach; either the dynamics must be linear or linearized, or the data must be strictly periodic or stationary for the linear models to be good estimators. It is noted that the estimation error could be considerably decreased with the availability of high-fidelity models and usage of nonlinear filters, which require numerical solutions; such numerical methods are usually computationally expensive and hence may not be suitable for real-time estimation. Many techniques of reliable state estimation have been reported in literature; examples are multiple model schemes [83], techniques based on analytical redundancy and residuals [84], and nonlinear observer theory [85]. In essence, the information from multiple sources must be synergistically aggregated for diagnosis and control of distributed physical processes (e.g., shipboard auxiliary systems). In this context, information fusion requires processing of non-stationary sensor information to detect parametric or nonparametric changes in the underlying system.

This section presents the development of a sensor data fusion method for fault detection & classification in distributed physical processes with an application to shipboard auxiliary

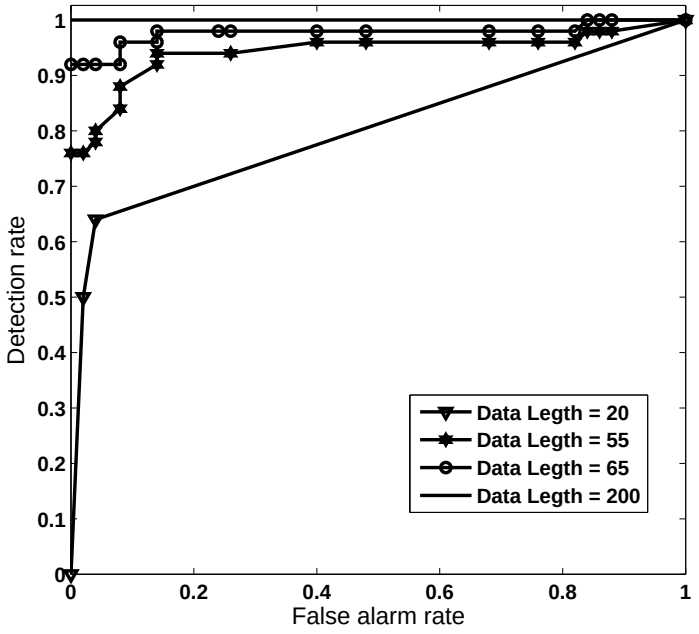


Figure 33. ROC curves for fan fault identification with different test data lengths (λ varying between 0.01 and 2 with steps of 0.01)

systems, where the process dynamics are interactive. For example, the electrical system is coupled with hydraulic system with time-dependent thermal load. The challenge here is to mitigate several inherent difficulties that include: (i) non-stationary behavior of signals, (ii) diverse nonlinearities of the process dynamics, (iii) uncertain input-output & feedback interactions and scaling, and (iv) alignment of multi-modal information and multiplicative process noise.

The sensor fusion concept, proposed in this section, is built upon the algorithmic structure of symbolic dynamic filtering (SDF) [15]. A spatiotemporal pattern network is constructed from disparate sensors and the fully connected network is then pruned by applying an information-theoretic (e.g., mutual information-based) approach to reduce computational complexity. The developed algorithms are demonstrated on a test bed that is constructed based on a notional MATLAB/Simulink model of Shipboard Auxiliary Systems, where a notional electrical system is coupled with a notional hydraulic system under a thermal load. A benchmark problem is created and the results under different performance metrics are presented.

The section is organized in three sections including the present one. Section 5.1. presents the semantic framework for multi-sensor data modeling and explains how the proposed technique is used to prune the heterogenous sensor network for information fusion. Section 5.2. presents a fault injection scheme in ship-board auxiliary unit (section 2.4.) for conducting simulation exercises and validates the fault detection accuracy for different scenarios in the proposed method of information fusion.

5.1. Multi-sensor Data Modeling and Fusion

This section presents a semantic information fusion framework that aims to capture temporal characteristics of individual sensor observations along with co-dependence among spatially distributed sensors. The concept of spatiotemporal pattern networks (STPNs) represent temporal dynamics of each sensor and their relational dependencies as probabilistic finite state automata (PFSA). Patterns emerging from individual sensors and their relational dependencies are called atomic patterns (AP) and relational patterns (RP), respectively. Sensors, APs, and RPs are represented as nodes, self-loop links, and links between pairs of nodes, respectively, in the STPN framework.

5.1.1. Modeling of Temporal Dynamics of Individual Sensor Data

This subsection briefly describes the concept of symbolic dynamic filtering (SDF) [15] for extracting atomic patterns from single-sensor data. The key concepts of SDF are succinctly presented below for completeness of the section. For more detail subsection 1.1. should be consulted.

Symbolic feature extraction from time series data is posed as a two-time-scale problem as explained before. Symbolic dynamic filtering (SDF) encodes the behavior of (possibly nonlinear) dynamical systems from the time series by symbolization and subsequent state machine construction. This is followed by computation of the state probability vectors (or symbol generation matrices) that are representatives of the evolving statistical characteristics of the dynamical system. To this end, the time series data are partitioned into a mutually exclusive and exhaustive set of finitely many cells; maximum-entropy partitioning (MEP) [12] has been adopted to construct the symbol alphabet Σ for generating symbol sequences, where the information-rich regions of the data set are partitioned finer and those with sparse information are partitioned coarser in order that the Shannon entropy [5] of the generated symbol sequence is maximized.

Consequently a D-Markov machine is constructed from the symbol sequence based on: (i) state splitting that generates symbol blocks of different lengths according to their relative importance; and (ii) state merging that assimilates histories from symbol blocks leading to the same symbolic behavior [86]. Words of length D on a symbol sequence are treated as the states of the D-Markov machine before any state-merging operation is executed. Thus, on an alphabet Σ, the total number of possible states becomes less than or equal to $|\Sigma|^D$; and operations of state merging may significantly reduce the number of states.

Let the state of a sensor A at the k^{th} instant be denoted as q_k^A. With this notation, the ij^{th} matrix element π_{ij}^A of the (stationary) state transition matrix Π^A (atomic pattern for sensor A) is the probability that q_{k+1}^A state is i given that the q_k^A state was j, i.e.,

$$\pi_{ij}^A \triangleq P\left(q_{k+1}^A = i \mid q_k^A = j\right) \text{ for an arbitrary instant } k$$

5.1.2. Pattern Analysis of Multi-sensor Information

Relational patterns (that are necessary for construction of STPN) are essentially extracted from the relational probabilistic finite state automata (PFSA). These PFSA are obtained

as xD-Markov machines to determine cross-dependence as defined below; the underlying algorithm is described in Subsection 5.1.2..

Construction of Relational PFSA: xD-Markov machine

This section describes the construction of xD-Markov machines from two symbol sequences $\{\mathbf{s}_1\}$ and $\{\mathbf{s}_2\}$ obtained from two different sensors (possibly of different modalities) to capture the symbol level cross-dependence. A formal definition is as follows:

Definition 5..1 (xD-Markov)**.** *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be the PFSAs corresponding to symbol streams $\{\boldsymbol{s}_1\}$ and $\{\boldsymbol{s}_2\}$ respectively. Then a xD-Markov machine is defined as a 5-tuple $\mathcal{M}_{1\to 2} \triangleq (\mathcal{Q}_1, \Sigma_1, \Sigma_2, \delta_1, \tilde{\Pi}_{12})$ such that:*

- $\Sigma_1 = \{\sigma_0, ..., \sigma_{|\Sigma_1|-1}\}$ *is the alphabet set of symbol sequence* $\{\boldsymbol{s}_1\}$
- $\mathcal{Q}_1 = \{q_1, q_2, \ldots, q_{|\mathcal{Q}_1|}\}$ *is the state set corresponding to symbol sequence* $\{\boldsymbol{s}_1\}$
- $\Sigma_2 = \{\sigma_0, ..., \sigma_{|\Sigma_2|-1}\}$ *is the alphabet set of symbol sequence* $\{\boldsymbol{s}_2\}$
- $\delta_1 : \mathcal{Q}_1 \times \Sigma_1 \to \mathcal{Q}_1$ *is the state transition mapping that maps the transition in symbol sequence* $\{\boldsymbol{s}_1\}$ *from one state to another upon arrival of a symbol in* $\{\boldsymbol{s}_1\}$
- $\tilde{\Pi}_{12}$ *is the symbol generation matrix of size* $|\mathcal{Q}_1| \times |\Sigma_2|$*; the* ij^{th} *element of* $\tilde{\Pi}_{12}$ *denotes the probability of finding the symbol* σ_j *in the symbol string* $\{\boldsymbol{s}_2\}$ *while making a transition from the state* q_i *in the symbol sequence* $\{\boldsymbol{s}_1\}$

In practice, $\tilde{\Pi}_{12}$ is reshaped into a vector of length $|\mathcal{Q}_1| \times |\Sigma_2|$ and is treated as the extracted feature vector that is a low-dimensional representation of the relational dependence between $\{\mathbf{s}_1\}$ and $\{\mathbf{s}_2\}$. This feature vector is called a relational pattern (RP). When both symbol sequences are the same, RPs are essentially the atomic pattern (AP) corresponding to the symbol sequence; in that case, the xD-Markov machine reduces to a simple D-Markov machine. It is noted that an RP between two symbol sequences is not necessarily symmetric; therefore, RPs need to be identified for both directions. It is also useful to quantify cross state transition matrices Π^{AB} and Π^{BA} to quantify the state level cross-dependence between sensors A and B. As illustrated in Fig. 34, elements of the state transition matrices Π^{AB} and Π^{BA} corresponding to the cross machines are expressed as:

$$\pi_{k\ell}^{AB} \triangleq P\left(q_{n+1}^{B} = \ell \mid q_n^{A} = k\right) \ \forall n$$

$$\pi_{ij}^{BA} \triangleq P\left(q_{n+1}^{A} = j \mid q_n^{B} = i\right) \ \forall n$$

where $j, k \in Q^A$ and $i, \ell \in Q^B$. For a xD-Markov machine, the cross state transition matrix is constructed from symbol sequences generated from two sensors by identifying the probability of occurrence of a state in one sensor from another state in the second sensor. For depth $D = 1$ in a a xD-Markov machine, a cross state transition matrix and the corresponding cross symbol generation matrix are identical.

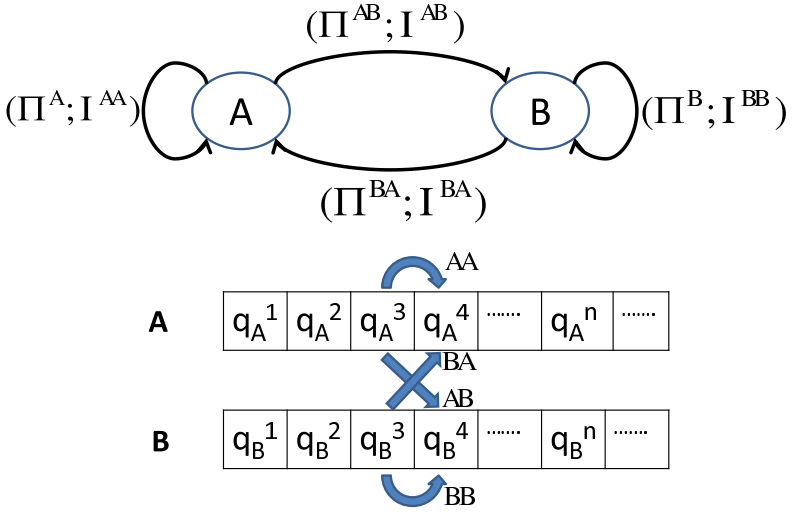


Figure 34. Illustration of Spatiotemporal Pattern Network (STPN)

Pruning of STPN

From the system perspectives, all APs and RPs need to be considered in order to model the nominal behaviors and to detect anomalies. However, it is obvious that there is a scalability issue if there is a significant number of sensors because the number of relational patterns increases quadratically with the number of sensors; for example, the number of RPs could be $S(S-1)$ where S is the total number of sensors and total number of patterns become S^2. The explosion of the pattern space dimension may prohibit the use of a *complete* STPN approach for monitoring of large systems under computational and memory constraints. However, for many real systems, a large fraction of relational patterns may have a very low information content due to the lack of their physical (e.g., electro-mechanical or via feedback control loop) dependencies. Therefore, a pruning process needs to be established to identify a *sufficient* STPN for a system. This section adopts an information-theoretic measure based on *Mutual Information* to identify the importance of an AP or an RP. Mutual information-based criteria have been very popular and useful in general graph pruning strategies [87, 88] including structure learning of Bayesian Networks [89]. In the present context, mutual information quantified on the corresponding state transition matrix essentially provides the information contents of APs and RPs. The concept of network pruning strategy is briefly described below.

Mutual information for the atomic pattern of sensor A is expressed as:

$$I^{AA} = I(q^A_{n+1}; q^A_n) = H(q^A_{n+1}) - H(q^A_{n+1}|q^A_n)$$

where

$$H(q^A_{n+1}) = -\sum_{i=1}^{Q_A} P(q^A_{n+1} = i) \log_2 P(q^A_{n+1} = i)$$

$$H(q^A_{n+1}|q^A_n) = -\sum_{i=1}^{Q_A} P(q^A_n = i) H(q^A_{n+1}|q^A_n = i)$$

$$H(q^A_{n+1}|q^A_n = i) = -\sum_{l=1}^{Q_A} P(q^A_{n+1} = l|q^A_n = i) \cdot \log_2 P(q^A_{n+1} = l|q^A_n = i)$$

The quantity I^{AA} essentially captures the temporal self-prediction capability (self-loop) of the sensor A. However, as an extreme example, the AP for a random sensor data may not be very informative and its self mutual information becomes zero under ideal estimation.

Similarly, mutual information for the relational pattern R^{AB} is expressed as:

$$I^{AB} = I(q^B_{n+1}; q^A_n) = H(q^B_{n+1}) - H(q^B_{n+1}|q^A_n)$$

where

$$H(q^B_{n+1}|q^A_n) = -\sum_{i=1}^{Q_A} P(q^A_n = i) H(q^B_{n+1}|q^A_n = i)$$

$$H(q^B_{n+1}|q^A_n = i) = -\sum_{l=1}^{Q_B} P(q^B_{n+1} = l|q^A_n = i) \cdot \log_2 P(q^B_{n+1} = l|q^A_n = i)$$

The quantity I^{AB} essentially captures sensor A's capability of predicting sensor B's outputs and vice versa for I^{BA}. Similar to atomic patterns, an extreme example would be the scenario where sensors A and B are not co-dependent (i.e., sensor A completely fails to predict temporal evolution of sensor B). In this case, R^{AB} is not very informative and I^{AB} will also be zero under ideal estimation.

Therefore, mutual information is able to assign weights on the patterns based on their relative importance (i.e., information content). The next step is to select certain patterns from the entire library of patterns based on a threshold on the metric. In this section, patterns are selected based on a measure of information gain due to atomic and relational patterns. Formally, let the total information gain I^{tot}_G is defined as the sum of mutual information for all patterns, i.e.,

$$I^{tot}_G = \sum_{(A,B)\in\mathcal{S}\times\mathcal{S}} I^{AB} \tag{75}$$

where $\mathcal{S}$ is set of all sensors. Now, the goal is to eliminate insignificant patterns from the set $\mathcal{S}\times\mathcal{S}$ of all patterns. Let the set of rejected patterns be denoted as $\mathcal{P}^{rej} \subset \mathcal{S}\times\mathcal{S}$ and the corresponding information gain be denoted as I^{rej}_G. The set of rejected patterns is chosen such that, for a specified $\eta \in (0, 1)$,

$$\frac{I^{rej}_G}{I^{tot}_G} < \eta \tag{76}$$

Table 6. Sensors of the system

System	Sensor	Physical Quantity
Electrical	T_e	Torque output of PMSM
	w_e	Rotor speed of PMSM
Hydraulic	w_{hp}	Angular Velocity of Hydraulic Pump
	P_{hm}	Pressure across Hydraulic Motor (HM)
	T_{hm}	Torque output of HM
	w_{hm}	Angular Velocity of output shaft of HM
Thermal	T_f	Temperature

Table 7. Fault parameters of the system

System	Fault parameter	Symbol	Range
Electrical	Flux linkage of PMSM	W_b	Nominal:0.05 ± 0.005
			Fault:0.03 ± 0.005
Hydraulic	Volumetric Efficiency of HM	ν_{vm}	Nominal:0.9 ± 0.02
			Fault:0.8 ± 0.02
	Total Efficiency of HM	ν_{tm}	Nominal:0.8 ± 0.02
			Fault:0.65 ± 0.02
Thermal	Thermal efficiency	ν_{th}	Nominal:0.9 ± 0.02
			Fault:0.8 ± 0.02

where mutual information for any pattern in the reject set should be smaller than the mutual information for any pattern in the accepted set $\mathcal{P}^{acc} \triangleq (\mathcal{S} \times \mathcal{S}) \setminus \mathcal{P}^{rej}$, which is expressed as $I^{AB}|_{(A,B)\in\mathcal{P}^{rej}} \ll I^{CD}|_{(C,D)\in\mathcal{P}^{acc}}$ for a sufficiently small η. In this section, η is chosen as 0.1 for the validation experiments. In the pruning strategy, it is possible that all patterns related to a certain sensor might be rejected, implying that the sensor is not useful for the purpose at hand. However, the user may choose an additional constraint to keep at least one (atomic or relational) pattern for each sensor in the accepted set of patterns.

Remark 5..1. *In order to use the STPN for fault detection, a network of PFSA can be identified following the above process under the nominal condition. Faulty conditions can then be detected by identifying the changes in parameters related to the accepted patterns. However, under some severe faults, the causal dependency characteristics (i.e., the structure itself) among the sensor nodes may not remain invariant. In such cases, new structures of the STPN can signify severely faulty conditions.*

5.2. Fault Detection in Ship-board Auxiliary Unit via Sensor Fusion

This section presents and discusses the results of validation of the the sensor fusion algorithm on the simulation test bed of shipboard auxiliary systems described in subsection 2.4..

5.2.1. Fault injection, sensors, data partitioning

Each of the electrical, hydraulic and thermal subsystems of the shipboard auxiliary system is provided with a set of sensors as listed in Table 6. In the simulation test bed, selected parameters in each subsystem can be perturbed to induce faults as listed in Table 7.

The pertinent assumptions in the execution of fault detection algorithms are delineated below.

- At any instant of time, the system is subjected to at most one of the faults mentioned in Table 7, because the occurrence of two simultaneous faults is rather unlikely in real scenarios.
- The mechanical efficiency of a hydraulic motor or a pump is assumed to stay constant over the period of observation as the degradation of machines due to wear and tear occurs at a much slower rate with respect to the drop in efficiency.
- The dynamical models in the simulation test bed are equipped with standard commercially available sensors. Exploration of other feasible sensors (e.g., ultra-high temperature sensors) to improve fault detection capabilities is not the focus of this study.

Figure 35 depicts a typical electromagnetic torque output for nominal and PMSM fault cases. As it is seen that the data itself is very noisy and, due to feedback control actions, there is no significant observable difference in the two cases in Fig. 35. Information integration from disparate sensors has been performed to enhance the detection & classification accuracy in such critical fault scenarios.

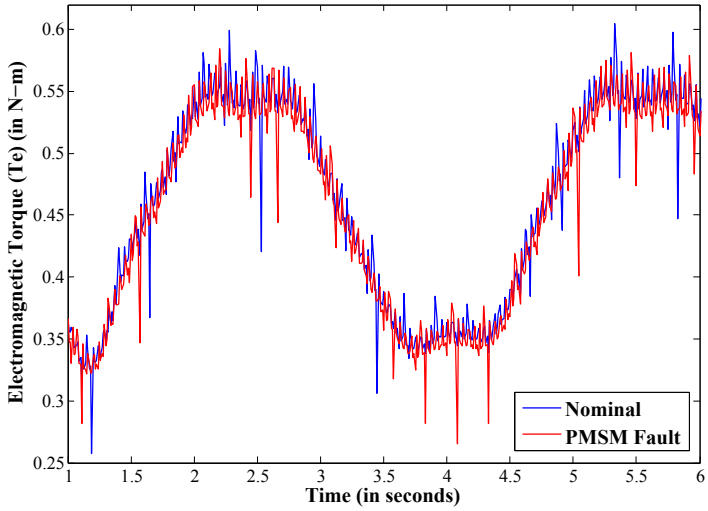


Figure 35. Electromagnetic torque under nominal and PMSM fault conditions

For all the fault scenarios 100 samples from each sensor are equally divided into two parts for training and testing purposes. For symbolization, maximum entropy partitioning is used with alphabet size, $|\Sigma| = 6$ for all sensors although $|\Sigma|$ does not need to be same

for individual sensors. The depth for constructing $PFSA$ states is taken to be $D = 1$ for construction of both atomic pattern and relational pattern. A reduced set of these patterns are aggregated to form the composite pattern, which serves as the feature classified by a k-NN classifier (with $k = 5$) using the Euclidean distance metric for fault detection [40].

5.2.2. Fusion with complete STPN

One sensor from each of the subsystems (i.e., T_f from thermal subsystem, T_e from electrical subsystem and T_{hm} from hydraulic subsystem) is selected for sensor fusion to identify component faults in the system. The composite patterns (CPs) are formed by concatenating atomic and relational patterns. Therefore, while patterns with high information content (based on the formulation above) help distinguishing between classes, patterns with low information content dilutes the ability of separating classes. Therefore, removing non-informative patterns may lead to reduction of both false alarm, missed detection rates and computational complexity. In this study, CPs consist of all possible APs and RPs of T_f, T_e and T_{hm}. It is seen in Table 8 that CPs perform better for detection of the nominal condition than individual sensors, but the false alarm rate is still high.

Table 8. Fault classification accuracy by exhaustive fusion

class	T_f	T_e	T_{hm}	CP
Nominal	32%	42%	32%	68%
PMSM fault	30%	100%	40%	84%
HM fault	40%	100%	100%	100%
Thermal fault	100%	58%	44%	100%

5.2.3. Pruning of STPN

Pruning of large sensor networks of the given system is attempted here to reduce the complexity of fusion and improve the detection accuracy by capturing the essential spatiotemporal dynamics of the system. Left half of the Fig. 36 shows a fully connected graph of seven sensors of the system where each node is a sensor; bi-directional arcs among them depict the RPs in both directions and self-loops are the APs corresponding to sensors.

The right half of Fig. 36 demonstrates the pruned STPN, where the thickness of arcs represents the intensity of mutual information of the RPs among sensors. Both directions of arrows are preserved as the mutual information of the two oppositely directed RPs for a pair of sensors are comparable. In this example, all the self loops are kept intact and arcs with negligible mutual information are omitted from the graph. In this simulation study, the structure of the reduced STPN is observed to remain stable for all the fault classes. The reduction in complexity of network graph is more significant in larger STPNs. The following two scenarios are chosen to justify the credibility of the pruned STPN in the light of fault detection accuracy.

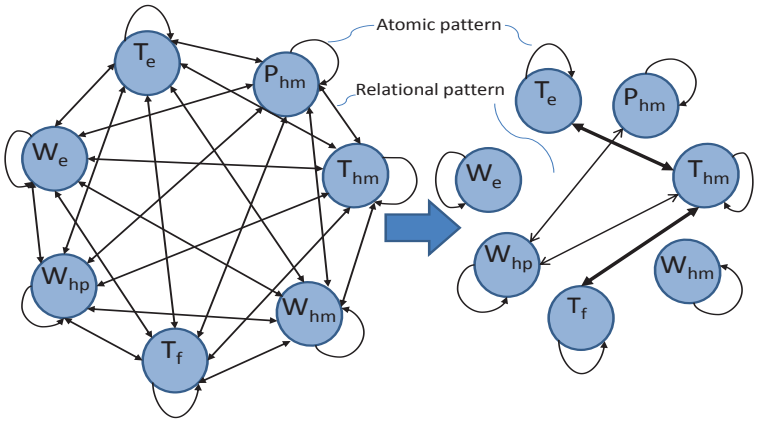


Figure 36. Pruning of STPN

Reduction of false alarm rates

The same set of sensors, namely, T_f, T_e and T_{hm}, are selected as the STPN and it is subjected to the proposed pruning technique, which results in a composite pattern of AP of T_e (Π^{T_e}) and two RPs ($\Pi^{T_{hm}T_e}$, $\Pi^{T_{hm}T_f}$ as shown by two thick arcs in Fig. 36). This action significantly reduces the false alarm rate as seen in Table 9, where APs of T_f and T_{hm} are dropped from the CP because these patterns do not facilitate better detection. Also PMSM fault detection accuracy does not degrade from 100% unlike fusion with complete STPN. Hence, this pruning technique reduces a CP containing 9 patterns (i.e., 3 APs, 6 RPs) to a CP of three APs and two RPs along with providing better class separability.

Table 9. Comparison of false alarm rate generated by exhaustive sensor fusion and pruned STPN

Fusion type	False alarm rate
complete STPN	32%
Pruned STPN	8%

Adaptability to malfunctioning sensors

In a distributed physical process, such as the shipboard auxiliary system under consideration, malfunctioning of primary sensors in a subsystem is a plausible event. One of the current challenges in the fault detection area is to identify a fault in the subsystem with malfunctioning sensors from the sensor responses of the subsystems that are electromechanically connected. To simulate that situation, three prime heterogenous sensors from the hydraulic subsystem, namely, w_{hp}, P_{hm} and T_{hm}, are selected and a fault is injected to the thermal subsystem by degrading the thermal efficiency (see Table 7). The T_f sensor of thermal subsystem is chosen to be the malfunctioning sensor and hence it is not incorporated in the detection process of thermal fault.

As the individual sensors of the hydraulic subsystem performs rather poorly in detecting thermal faults as seen in Table. 10, the information from these three sensors are

Table 10. Thermal fault detection by sensors of hydraulic subsystem

	w_{hp}	P_{hm}	T_{hm}	CP
Detection accuracy	58%	18%	18%	70%

fused by applying the proposed pruning technique on the hydraulic subsystem. The pruned STPN yields a CP consisting of an AP of w_{hp} ($\Pi^{w_{hp}}$) and two RPs, namely, $\Pi^{P_{hm}w_{hp}}$ and $\Pi^{T_{hm}w_{hp}}$, depicted by two thin arcs in Fig. 36; it results in a decent detection accuracy of 70% as seen in Table 10.

6. Summary, Conclusions and Future Research

The chapter presents some of the key recent developments of the stochastic time-series analysis tool called the symbolic dynamic filtering (SDF) with a special emphasis on its application in the field of fault detection and diagnostics of complex industrial systems. Three major aspects of the tool are highlighted: (i) data-space abstraction via partitioning, (ii) encoding and pattern classification of time-series motifs and (iii) spatio-temporal information fusion. Validation results are presented on high fidelity industrial testbeds of various domains, such as aircraft gas turbine engines, fatigue damage problems, nuclear power plants and ship-board auxiliary systems. The major advantages of SDF for detecting both small and large changes in a dynamical system are listed below:

- Robustness to measurement noise and spurious operational disturbances [12]
- Adaptability to low-resolution sensing due to the coarse graining in space partitions [15]
- Capability for early detection of anomalies because of sensitivity to signal distortion [90]
- real-time execution on commercially available inexpensive platforms [6][90].

While the above mentioned advantages have been demonstrated in multiple industrial system testbeds and small-scale laboratory setups, a key next step is to validate the concepts on real-life systems in order to reduce possible risks and increase robustness. Apart from that, some of the broader research directions that are currently being pursued are:

- *From early detection to prognostics*: SDF has been particularly useful for early detection of anomalies in complex systems. An important current work is concentrating on how to translate this capability to a prognostics capability which is absolutely crucial especially for safety-critical systems.
- *Contextual adaptation*: To become a sustainable health monitoring tool over the lifecycles of industrial systems, SDF will be required to adapt based on contexts regarding internal operations or external environment. Future research will be dedicated to build in and automate different contextual adaptation features in SDF.

- *Soft data integration and HMI*: It is increasingly becoming obvious in the machine learning community that exploiting human knowledge and intent can be significantly more rewarding compared to designing a completely automated systems. To this end, a unified mathematical framework is being investigated that can accommodate both human and machine information.

- *Mathematics of PFSA*: A recent development on Hilbert space formulation of Probabilistic Finite State Automata (PFSA) [91] demonstrated a path of formalizing the SDF developments that can be particularly useful for information fusion purposes. Further studies are being pursued to investigate this area in a greater detail.

Acknowledgements

The authors acknowledge the assistance and technical contributions of their colleagues Dr. Kushal Mukherjee, Dr. Yicheng Wen, Dr. Abhishek Srivastav and Dr. Shashi Phoha in the reported work.

References

[1] S. Gupta and A. Ray, "Statitical mechanics of complex systems for pattern identification," *Journal of Statistical Physics*, vol. 134, no. 2, pp. 337–364, 2009.

[2] R. Pathria, *Statistical Mechanics*. Oxford, UK: Butterworth-Heinemann, 2nd ed., 1996.

[3] D. Lind and M. Marcus, *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, Cambridge, U.K., 1995.

[4] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. New York, NY: Wiley Interscience, 2nd edition ed., 2001.

[5] T. Cover and J. Thomas, *Elements of Information Theory, 2nd ed.* New York, NY, USA: Wiley, 2006.

[6] C. Rao, A. Ray, S. Sarkar, and M. Yasar, "Review and comparative evaluation of symbolic dynamic filtering for detection of anomaly patterns," *Signal, Image and Video Processing*, vol. 3, no. 2, pp. 101–114, 2009.

[7] C. Beck and F. Schl*ö*gl, *Thermodynamics of chaotic systems: an introduction*. Cambridge University Press, United Kingdom, 1993.

[8] R. Badii and A. Politi, *Complexity hierarchical structures and scaling in physics*. Cambridge University Press, United Kingdom, 1997.

[9] C. S. Daw, C. E. A. Finney, and E. R. Tracy, "A review of symbolic analysis of experimental data," *Review of Scientific Instruments*, vol. 74, no. 2, pp. 915–930, 2003.

[10] H. Kantz and T. Schreiber, *Nonlinear Time Series Analysis, 2nd ed.* Cambridge, U.K.: Cambridge University Press, 2004.

[11] M. B. Kennel and M. Buhl, "Estimating good discrete partitions form observed data: Symbolic false nearest neighbors," *Physical Review E*, vol. 91, no. 8, pp. 084–102, 2003.

[12] V. Rajagopalan and A. Ray, "Symbolic time series analysis via wavelet-based partitioning," *Signal Processing*, vol. 86, no. 11, pp. 3309–3320, 2006.

[13] A. Subbu and A. Ray, "Space partitioning via hilbert transform for symbolic time series analysis," *Applied Physics Letters*, vol. 92, no. 8, pp. 084107–1 to 084107–3, February 2008.

[14] S. Sarkar, K. Mukherjee, and A. Ray, "Generalization of hilbert transform for symbolic analysis of noisy signals," *Signal Processing*, vol. 89, no. 6, pp. 1245–1251, 2009.

[15] A. Ray, "Symbolic dynamic analysis of complex systems for anomaly detection," *Sig. Process.*, vol. 84, no. 7, pp. 1115–1130, 2004.

[16] C. Rao, K. Mukherjee, S. Sarkar, and A. Ray, "Statistical estimation of multiple parameters via symbolic dynamic filtering," *Signal Processing*, vol. 89, no. 6, pp. 981–988, June 2009.

[17] A. Ray, "Signed real measure of regular languages for discrete-event supervisory control," *Int. Journal of Control*, vol. 78, no. 12, pp. 949–967, 2005.

[18] R. Bapat and T. Raghavan, *Nonnegative Matrices and Applications*. Cambridge, U.K: Cambridge University Press, 1997.

[19] Y. Wen and A. Ray, "A stopping rule for symbolic dynamic filtering,," *Signal, Image, and Video Processing*, vol. 7, no. 1, pp. 189–195, 2013.

[20] Y. Wen, A. Ray, and Q. Du, "A variance-estimation-based stopping rule for symbolic dynamic filtering," *Signal, Image, and Video Processing*, vol. 7, no. 1, pp. 189–195, 2013.

[21] D. K. Frederick, J. A. DeCastro, and J. S. Litt, "Users guide for the commercial modular aero-propulsion system simulation (C-MAPSS)," October 2007. NASA/TM2007-215026.

[22] M. D. Carelli, "The design and safety features of the IRIS reactor," *Nuclear Engineering and Design*, vol. 230, pp. 151–167, 2004.

[23] S. C. Stultz and J. B. Kitto, *Steam: its generation and use*. Babcock & Wilcox Company, 40 ed., 1992.

[24] F. Koushanfar, M. Potkonjak, and A. Sangiovanni-Vincentelli, "On-line fault detection of sensor measurements," *IEEE Sensors*, pp. 974–980, 2003.

[25] H. M. Hashemian, *Maintenance of Process Instrumentation in Nuclear Power Plants*. Springer, Berlin, Germany, 2006.

[26] D. Gabor, "Theory of communications," *J. Inst. Electrical Engineering*, vol. 93, p. 429457, 1946.

[27] L. Cohen, *Time-Frequency Analysis*. Prentice Hall PTR, 1995.

[28] A. Lohmann, D. Mendlovic, and Z. Zalevsky, "Fractional hilbert transform," *Opt. Lett.*, vol. 21, no. 4, p. 281283, 1996.

[29] S.-C. Pei and M.-H. Yeh, "Discrete fractional hilbert transform," *IEEE Transactions on Circuits and SystemsII: Analog and Digital Signal Processing*, vol. 47, no. 11, 2000.

[30] Y. Luo, S. Al-dossary, M. Mahroon, and M. Alfaraj, "Generalized hilbert transform and its applications in geophysics," *The Leading Edge*, March 2003.

[31] B. S. Chlebus and S. H. Nguyen, "On finding optimal discretizations for two attributes," in *Proceedings of the First International Conference on Rough Sets and Current Trends in Computing*, RSCTC '98, (London, UK, UK), pp. 537–544, Springer-Verlag, 1998.

[32] Y. Yang and G. I. Webb, "Discretization for naive-bayes learning: managing discretization bias and variance," *Machine Learning*, vol. 74, pp. 39–74, 2009.

[33] S. Garcia, J. Luengo, J. A. Saez, V. Lopez, and F. Herrera, "A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 99, no. PrePrints, 2012.

[34] H. Liu, F. Hussain, C. Tan, and M. Dash, "Discretization: An enabling technique," *Data Mining and Knowledge Discovery*, vol. 6, pp. 393–423, 2002.

[35] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and Unsupervised Discretization of Continuous Features," in *Proceedings of the Twelfth International Conference on Machine Learning* (A. Prieditis and S. Russell, eds.), pp. 194–202, 1995.

[36] Y. Yang, G. I. Webb, and X. Wu, "Discretization methods," in *Data Mining and Knowledge Discovery Handbook, 2nd ed.*, pp. 101–116, Springer, New York, NY, USA, 2010.

[37] A. Bakar, Z. Othman, and N. Shuib, "Building a new taxonomy for data discretization techniques," in *Data Mining and Optimization, 2009. DMO '09. 2nd Conference on*, pp. 132 –140, Oct. 2009.

[38] G. J. Mclachlan, *Discriminant Analysis and Statistical Pattern Recognition*. (Wiley Series in Probability and Statistics) Wiley-Interscience, 2004.

[39] E. Choi and C. Lee, "Feature extraction based on the bhattacharyya distance," *Pattern Recognition*, vol. 36, pp. 1703 – 1709, August 2003.

[40] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[41] M. Loog, R. Duin, and R. Haeb-Umbach, "Multiclass linear dimension reduction by weighted pairwise fisher criteria," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 7, p. 762766, 2001.

[42] A. Biem, S. Katagiri, and B. Juang, "Pattern recognition using discriminative feature extraction," *IEEE Trans. Signal Process.*, vol. 45, no. 2, p. 500504, 1997.

[43] R. Kohavi and F. Provost, "Glossary of terms," *Machine Learning*, vol. 30, no. 2/3, pp. 271–274, 1998.

[44] R. Alaiz-Rodrguez, A. Guerrero-Curieses, and J. Cid-Sueiro, "Minimax classifiers based on neural networks," *Pattern Recognition*, vol. 38, no. 1, pp. 29 – 39, 2005.

[45] K. Miettinen, *Nonlinear Multiobjective Optimization*. Kluwer Academic, Boston, MA, USA, 1998.

[46] X. Jin, S. Sarkar, K. Mukherjee, and A. Ray, "Suboptimal partitioning of time-series data for anomaly detection," in *Proceedings of 48th IEEE Conference on Decision and Control*, (Shanghai, China), pp. 1020–1025, December 2009.

[47] R. Steuer, *Multiple Criteria Optimization: Theory, Computations, and Application*. John Wiley & Sons, Inc., New York, USA, 1986.

[48] L. Ljung, *System identification: Theory for the User, 2nd ed.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999.

[49] O. Nelles, *Nonlinear system identification from classical to neural networks and fuzzy models*. Springer, Berlin, Germany, 2001.

[50] H. Garnier and L. Wang, *Identification of continuous-time models from sampled data from classical to neural networks and fuzzy models*. Springer, London, U.K., 2008.

[51] I. Chattopadhyay and A. Ray, "Structural transformation of probabilistic finite state machines," *Int. J. Control*, vol. 81, no. 5, pp. 820–835, 2008.

[52] X. Jin, S. Gupta, K. Mukherjee, and A. Ray, "Wavelet-based feature extraction using probabilistic finite state automata for pattern classification," *Pattern Recognition*, vol. 44, no. 7, pp. 1343–1356, 2011.

[53] C. R. Shalizi and K. L. Shalizi, "Blind construction of optimal nonlinear recursive predictors for discrete sequences," in *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, (Arlington, Virginia, United States), pp. 504–511, AUAI Press, 2004.

[54] I. Chattopadhyay, Y. Wen, A. Ray, and S. Phoha, "Unsupervised inductive learning in symbolic sequences via recursive identification of self-similar semantics," in *Preprints Proceedings of American Control Conference, San Francisco, CA, USA*, June-July 2011.

[55] S. Sarkar, X. Jin, and A. Ray, "Data-driven fault detection in aircraft engines with noisy sensor measurements," *Journal of Engineering for Gas Turbines and Power-Transactions of the ASME*, vol. 133, no. 8, p. 081602 (10 pages), 2011.

[56] S. Sarkar, C. Rao, and A. Ray, "Statistical estimation of multiple faults in aircraft gas turbine engines," *Proceedings of the I Mech E Part G: Journal of Aerospace Engineering*, vol. 223, no. 4, pp. 415–424, 2009.

[57] P. Adenis, Y. Wen, and A. Ray, "An inner product space on irreducible and synchronizable probabilistic finite state automata," *Math. Control Signals Syst.*, vol. 23, no. 4, pp. 281–310, 2012.

[58] T. Ferguson, "Ea bayesian analysis of some nonparametric problems," *The Annals of Statistics*, vol. 1, no. 2, 1973.

[59] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *ICRA'94*, pp. 3310–3317, 1994.

[60] S. Wilks, *Mathematical Statistics*. John Wiley, New York, NY, USA, 1963.

[61] Y. Wen, K. Mukherjee, and A. Ray, "Adaptation in symbolic dynamic systems for pattern classification," in *2012 American Control Conference*, (Monreal, Canada), 2012.

[62] H. Lipowsky, S. Staudacher, M. Bauer, and K.-J. Schmidt, "Application of Bayesian forecasting to change detection and prognosis of gas turbine performance," *Journal of Propulsion and Power*, vol. 132, no. 3, p. 031602, 2010.

[63] V. Guruprakash and R. Ganguli, "Three-and seven-point optimally weighted recursive median filters for gas turbine diagnostics," *Proceedings of the I Mech E Part G: Journal of Aerospace Engineering*, vol. 222, no. 3, pp. 307–318, May 2008.

[64] Y. G. Li, "A gas turbine diagnostic approach with transient measurements," *Proceedings of IMechE, Part A: Journal of Power and Energy*, vol. 217, no. 2, pp. 169–177, 2003.

[65] G. Merrington, O. Kwon, G. Goodwin, and B. Carlsson, "Fault detection and diagnosis in gas turbines," *J. Eng. Gas Turbines Power*, vol. 113, pp. 276–282, 1991.

[66] X. Wang, N. McDowell, U. Kruger, G. McCullough, and G. W. Irwin, "Semi-physical neural network model in detecting engine transient faults using the local approach," in *Proceedings of the 17th World Congress The International Federation of Automatic Control*, July 6-11, 2008.

[67] V. P. Surender and R. Ganguli, "Adaptive myriad filter for improved gas turbine condition monitoring using transient data," *J. Eng. Gas Turbines Power*, vol. 127, pp. 329–339, 2005.

[68] S. Menon, O. Uluyol, K. Kim, and E. O. Nwadiogbu, "Incipient fault detection and diagnosis in turbine engines using hidden markov models," *ASME Conference Proceedings*, vol. 2003, no. 36843, pp. 493–500, 2003.

[69] S. Gupta, A. Ray, S. Sarkar, and M. Yasar, "Fault detection and isolation in aircraft gas turbine engines: Part i - underlying concept," *Proceedings of the I Mech E Part G: Journal of Aerospace Engineering*, vol. 222, no. 3, pp. 307–318, May 2008.

[70] S. Sarkar, M. Yasar, S. Gupta, A. Ray, and K. Mukherjee, "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*, vol. 222, no. 3, pp. 319–330, May 2008.

[71] J. Armstrong, "Users guide for the transient test case generator," September 2009. NASA GRC Internal Report.

[72] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[73] J. Liu and J. M. A. Scherpen, "Fault detection method for nonlinear systems based on probabilistic neural network filtering," *International Journal of Systems Science*, vol. 33, pp. 1039–1050, 2002.

[74] S. Haykin, *Neural Networks : A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice Hall, 1999.

[75] M. R. Napolitano, Y. An, and B. A. Seanor, "A fault tolerant flight control system for sensor and actuator failures using neural networks," *Aircraft Design*, vol. 3, pp. 103–128, 2000.

[76] K. Fukunaga, *Statistical Pattern Recognition*. Boston, MA, USA: Academic Press, 2 ed., 1990.

[77] J. Shawe-Taylor, *Kernel Methods for Pattern Analysis*. Cambridge, U.K.: Cambridge University Press, 2004.

[78] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice Using MATLAB*. John Wiley & Sons, 2 ed., 2001.

[79] A. Gelb, ed., *Applied Optimal Estimation*. The MIT Press, 1974.

[80] C. Andrieu, A. Doucet, S. Singh, and V. B. Tadic, "Particle methods for change detection, system identification, and control," *Proceedings of IEEE*, vol. 92, pp. 423–438, 2004.

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

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

[83] M. Gopinathan, J. D. Boskovic, R. K. Mehra, and C. Rago, "A multiple model predictive scheme for fault-tolerant flight control design," in *Conference on Decision and Control*, pp. 1376–1381, 2001.

[84] J. J. Gertler, "Survey of model-based failure detection and isolation in complex plants," *IEEE Control Systems Magazine*, vol. 8, pp. 3–11, 1988.

[85] E. A. Garcia and P. M. Frank, "Deterministic nonlinear observer-based approaches to fault diagnosis: A survey," *Control Engineering Practice*, vol. 5, pp. 663–670, 1997.

[86] P. Adenis, K. Mukherjee, and A. Ray, "State splitting and state merging in probabilistic finite state automata," in *American Control Conference, San Francisco, CA, USA*, pp. 5145–5150, 2011.

[87] H. Kretzschmar, C. Stachniss, and G. Grisetti, "Efficient information-theoretic graph pruning for graph-based slam with laser range finders," in *International Conference on Intelligent Robots and Systems (IROS)*, pp. 865 – 871, 2011.

[88] A. J. Butte and I. S. Kohane, "Mutual information relevance networks:functional genomic clustering using pairwise entropy measurements," in *Pacific Symposium on Biocomputing*, pp. 418–429, 2000.

[89] L. M. de Campos, "A scoring function for learning bayesian networks based on mutual information and conditional independence tests," *Journal of Machine Learning Research*, vol. 7, pp. 2149–2187, 2006.

[90] S. Gupta, A. Ray, and E. Keller, "Symbolic time series analysis of ultrasonic data for early detection of fatigue damage," *Mechanical Systems and Signal Processing*, vol. 21, no. 2, pp. 866–884, 2007.

[91] Y. Wen, A. Ray, and S. Phoha, "Hilbert space formulation of symbolic systems for signal representation and analysis," *Signal Processing*, vol. 93, no. 9, pp. 2594–26, September 2013.