Data-Driven Fault Detection in Aircraft Engines With Noisy Sensor Measurements

Soumik Sarkar
e-mail: szs200@psu.edu

Xin Jin
e-mail: xuj103@psu.edu

Asok Ray
e-mail: axr2@psu.edu

Department of Mechanical Engineering,
Pennsylvania State University,
University Park, PA 16802

An inherent difficulty in sensor-data-driven fault detection is that the detection performance could be drastically reduced under sensor degradation (e.g., drift and noise). Complementary to traditional model-based techniques for fault detection, this paper proposes symbolic dynamic filtering by optimally partitioning the time series data of sensor observation. The objective here is to mask the effects of sensor noise level variation and magnify the system fault signatures. In this regard, the concepts of feature extraction and pattern classification are used for fault detection in aircraft gas turbine engines. The proposed methodology of data-driven fault detection is tested and validated on the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) test-bed developed by NASA for noisy (i.e., increased variance) sensor signals.
[DOI: 10.1115/1.4002877]

Keywords: aircraft gas turbine engines, data-driven fault detection, optimal feature extraction, multiclass classification

1 Introduction

Health monitoring of human-engineered complex systems (e.g., aircraft gas turbine engines), is generally divided into three steps, namely, detection, diagnosis, and prognosis. While the detection step identifies the presence of a fault, its level, type, and possible location are determined in the diagnosis step. Given possible statistics of future operating conditions and an expected component deterioration profile, the role of prognosis is to estimate the remaining useful life of the system from the information generated in the diagnosis step. Along this line, health monitoring algorithms are primarily divided into two different categories, namely, model-based and data-driven.

Both model-based and data-driven techniques have been reported in literature for health monitoring of gas turbine engines. An example of model-based health monitoring is usage of state-space models of gas turbine engines coupled with Kalman filtering [1–3]. Among data-driven approaches, neural network (NN)-based approaches [4] are the most popular. A regression-based approach was used for detecting anomalies in aircraft performance during cruise flight [5]. While detail models could be computationally too intensive, lumped-parameter models of significantly less computational complexity have been adopted for in-situ fault detection [6]. Although such model-based techniques have their advantages in terms of on-board real-time application, their reliability for health monitoring often decreases as the system complexity increases. On the other hand, data-driven techniques may remain reliable and computationally efficient in spite of system complexity if they are required to monitor the input-output information from a limited number of (appropriately calibrated) sensors while considering the entire system as a black-box. However, unless the ensemble of acquired information is systematically handled, data-driven techniques may become computationally intensive and the performance of health monitoring may deteriorate due to sensor degradation. Furthermore, data-driven techniques usually require high volume training data (engine failure data in the present context). In real life, acquiring this type of data becomes problematic for reliable systems as aircraft engines. Thus, developing reliable damage models is an important aspect of this problem, which was addressed in Ref. [7]. Data-driven techniques for health monitoring of gas turbine engines either use snapshot data at a time instant from various sensors [7] or a window of time series data from selected sensor observations. The data volume is less when snapshot type data are handled and as a consequence, the computational expense is low as well. However, just by using snapshot data, statistical changes in the acquired information may not be adequately captured, eventually resulting in missed detection of faults; this problem can be alleviated by using a window of time series data. The problem with handling time series data is its volume and the associated computational complexity; therefore, the available information must be appropriately compressed via transformation of high-dimensional data into a low-dimensional feature space with minimal loss of class separability. In their previous work [8], the authors proposed a nonlinear feature extraction method, namely, symbolic dynamic filtering (SDF) for detection of anomalies (i.e., deviations from the nominal condition) in complex systems. This method is shown to be particularly useful for extraction of features from time series data and has been experimentally validated for real-time execution in different applications (e.g., electronic circuits [9] and fatigue damage monitoring in polycrystalline alloys [10]). Algorithms, constructed in the SDF setting, are shown to yield 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), NNs, and Bayesian techniques [11]. Recently, in a two-part paper [12,13], a 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 Ref. [14].

A major challenge in any sensor-data-driven detection tool is to identify the actual failure in the system in the presence of sensor degradation (e.g., drift and noise) without succumbing to a large number of false alarms or missed detections. The situation becomes even more critical if the control system uses observations from the degraded sensors as feed-back signals and generates the control inputs accordingly. Traditionally, redundant sensors along with methods based on analytic redundancy are used for sensor

Contributed by International Gas Turbine Institute (IGTI) of ASME for publication in the Journal of Engineering for Gas Turbines and Power. Manuscript received March 12, 2010; final manuscript received September 2, 2010; published online April 6, 2011. Assoc. Editor: Allan Volponi.

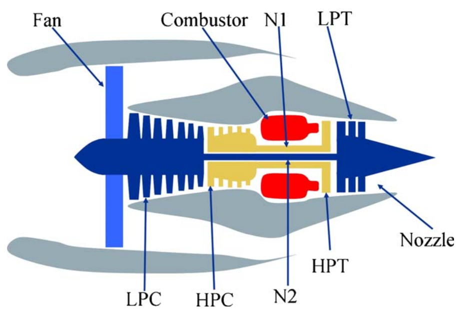

Fig. 1 Gas turbine engine schematic [17]

Table 1 Sensor suite for the engine system

Sensors	Description
P_2	Fan inlet pressure
T_2	Fan inlet temperature
P_{24}	LPC exit/HPC inlet pressure
T_{24}	LPC exit/HPC inlet temperature
Ps_{30}	HPC exit static pressure
T_{30}	HPC exit/burner inlet temperature
T_{48}	HPT exit temperature
N_f	Fan spool speed
N_c	Core spool speed

fault identification [15]. This paper presents a different approach to this problem, where different class labels are assigned to data sets that are generated from different engine health conditions. The same class labels are assigned to data from engines with similar health conditions and the associated sensor data are subjected to different noise variance at respective sensor degradation levels. To this end, the data-driven fault detection problem is posed as a multiclass pattern classification problem, where the tasks of class assignment and the SDF-based feature extraction are optimized in a supervised manner to enhance the classification performance. A major step in SDF-based feature extraction is partitioning of time series data to generate symbol blocks that are subsequently converted to feature vectors by the use of the probabilistic finite state automata (PFSA) concept [8]. The major contributions of this article beyond the authors' recent work [12–14] in the field of engine health monitoring are as follows:

- formulation of the data-driven fault detection problem at the engine component level in the presence of varying sensor noise condition, as a multiclass classification problem
- optimization of partitioning in the SDF-based feature extraction setting

This paper is organized into seven sections including the present one. Section 2 describes the NASA C-MAPSS simulation test-bed [16,17] of a commercial aircraft gas turbine engine model on which the problem of engine health monitoring has been formulated and validated. Section 3 poses data-driven fault detection under varying sensor noise condition, as a multiclass pattern classification problem. Section 4 presents a brief background and formulation of SDF-based feature extraction for the current problem and Sec. 5 describes the partitioning optimization methodology. Section 6 presents the results of two case studies to validate the proposed approach on the C-MAPSS test-bed. Section 7 summarizes the paper and makes major conclusions along with recommendations for future research.

2 Description of the C-MAPSS Test-Bed

This section briefly describes the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) simulation test-bed [17] that has been developed at NASA on a commercial-scale two-spool turbofan engine and its control system. Figure 1 shows a schematic diagram of the commercial aircraft gas turbine engine used in the C-MAPSS simulation test-bed.

The engine under consideration produces a thrust of approximately 400,000 N and is designed for operation at altitude (A) from sea level (i.e., 0 m) up to 12,200 m, Mach number (M) from 0 to 0.90, and sea-level temperatures from approximately -50°C to 50°C. The throttle resolving angle (TRA) can be set to any value in the range between 0 deg at the minimum power level and 100 deg 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 Fig. 1. Given the inputs of TRA, A and M, the interactively controlled component models at 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. To achieve fast execution of simulation runs, the sensors and actuators are approximated to have instantaneous response, no computational time delays, and no drift and or bias. The entire test-bed code is written on MATLAB and SIMULINK platform.

Out of the different types of sensors (e.g., pressure, temperature, and shaft speed) used in the C-MAPSS simulation model, Table 1 lists those sensors that are commonly adopted in the instrumentation and control system of commercial aircraft engines as seen in Fig. 2, where the engine control system makes use of the data from sensors P_2, Ps_{30}, T_{48}, N_f, and N_c as feedback signals. 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), and pressure ratio modifiers, that simulate the effects of faults and/or degradation in the engine components. Ten of these 13 health parameters are selected to modify efficiency (η) and flow (ϕ) that are defined [18] as follows:

- $\eta \triangleq$ 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), $(\psi_{\text{LPC}}, \zeta_{\text{LPC}})$, $(\psi_{\text{HPC}}, \zeta_{\text{HPC}})$, $((\psi_{\text{HPT}}, \zeta_{\text{HPT}})$, and $(\psi_{\text{LPT}}, \zeta_{\text{LPT}})$. 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, $\psi_{\text{HPC}}=0.98$ signifies a 2% relative loss in efficiency of HPC.

In the C-MAPSS test-bed, sensor degradations are realized as injected faults. Depending on the location and modality of the 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 sen-

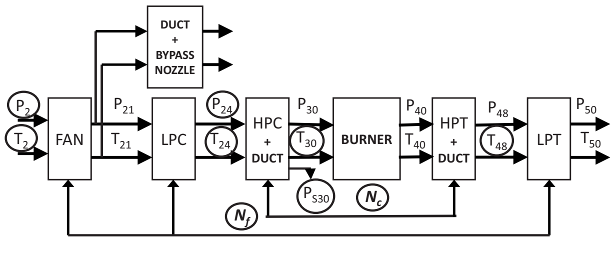

Fig. 2 Schematic diagram of the C-MAPSS engine model

Table 2 Fault levels in HPC

Fault level	Efficiency range (ψ_{HPC})
Low fault	1.0000–0.9834
Medium fault	0.9834–0.9668
High fault	0.9668–0.9500

Table 3 Variable noise levels in Ps_{30}

Noise level	Standard deviation range (%)
Level 1	0.35–0.45
Level 2	0.45–0.55
Level 3	0.55–0.65

sors of the same modality could have different degradation characteristics. In general, sensor degradation is categorized as the following [19]:

- bias fault (i.e., a constant bias in the sensor observation)
- drifting fault that is slowly varying
- change in sensor noise variance

Amongst these sensor degradation types, only sensor degradation due to changes in the sensor noise variance is chosen for this study. Two case studies have been presented in this paper to validate the proposed methodology. In case study 1, the HPC is chosen to be the location of the component-level fault and Ps_{30}, i.e., the static pressure sensor at the HPC outlet, is chosen for fault detection in HPC. In case study 2, the HPT is chosen to be the location of the component-level fault and T_{48}, i.e., the temperature sensor at the HPT outlet, is chosen for fault detection in HPT. It is noted that the integrated engine controller takes feedback signals from both Ps_{30} and T_{48} sensors. Therefore, it is likely that degradation in those sensors would pervade through the engine system, potentially affecting the outputs of the remaining components of the engine system due to the inherent electromechanical feedback. While the entire details of case study 2 is presented in Sec. 6.2, the problem description of case study 1 is described in Sec. 3 to pose the component-level fault detection under varying sensor noise condition as a multiclass classification problem.

3 Data-Driven Fault Detection Posed as a Multiclass Pattern Classification Problem

Component-level fault diagnosis in an aircraft gas turbine engine involves identification of the fault type, and location and quantification of the fault level. In the C-MAPSS test-bed setting, the physical fault scenarios (e.g., fouling, increased tip clearance, and seal wear) are assumed to manifest themselves in reducing the efficiency of the associated engine component(s). Although the present configuration in the C-MAPSS test-bed can be easily extended to simultaneous faults in multiple components (see the authors' previous publication [14] for details), the present paper deals with a single component, e.g., HPC, and the task is to detect a fault and identify its level under varying sensor noise condition.

3.1 Problem Description of Case Study 1: HPC Fault Detection. The two health parameters that define the health status of HPC are the efficiency and flow health parameters (ψ_{HPC}, ζ_{HPC}). However, it is observed that Ps_{30} sensor observation does not capture any signature of change in the flow health parameter ζ_{HPC} and since, only sensor Ps_{30} is used for the fault detection, three fault levels are considered only based on the efficiency health parameter ψ_{HPC}. Table 2 shows the approximate ranges of efficiency health parameters under different fault levels. Here, the low fault level indicates very minimal loss in efficiency and hence also includes the absolute nominal health condition ($\psi_{HPC}=1$).

Similarly, depending on the noise standard deviation in the Ps_{30} sensor, three sensor noise levels are considered for the study. Table 3 shows the approximate ranges of sensor noise standard deviation as percent of operating point trim values considered under different levels [20,21]. Note that the noise is multiplicative in the sense that its standard deviation is proportional to the operating point trim value of data.

Thus, in the above context, there are $(3\times3)=9$ classes of data sets that need to be obtained to define a class by a HPC fault level and a noise level of the Ps_{30} sensor. Seventy simulation runs of the engine system were performed for each class to generate data sets for analysis, among which 50 samples are chosen as the training set and the rest of the samples are kept as the testing set. HPC efficiency and Ps_{30} standard deviation parameters are chosen randomly from independent Gaussian distributions for both parameters, such that approximately 95% of the parameter values are within the prescribed ranges given in Tables 2 and 3. In other words, the mean of the Gaussian distribution used for a particular parameter level is taken as the central value of the parameter range in that level and the standard deviation is taken such that the boundary values of the parameter range are 2σ distance away from the central value. For example, for the class with low fault in HPC and level 1 noise in Ps_{30}, the HPC efficiency values were chosen from an independent Gaussian distribution with mean as 0.9917 and standard deviation as 4.15×10^{-3}, whereas the standard deviation percentage values were chosen from another independent Gaussian distribution with mean as 0.40 and standard deviation as 0.025. Figure 3 plots the samples generated using the above logic in the two-dimensional parametric space. Different classes of samples are shown in different colors in the figure. The axis for sensor noise standard deviation ($\sigma_{Ps_{30}}$) represents the actual standard deviation values (not as percent of the operating point trim values of Ps_{30} reading).

For each data sample, a time series was collected for Ps_{30} sensor under persistent excitation of TRA inputs that have truncated triangular profiles with the mean value of 40 deg, fluctuations within ±8 deg and frequency of 0.056 Hz as shown in Fig. 4. The ambient conditions are chosen to be at the sea level when the engine is on the ground (i.e., altitude $A=0.0$, Mach number $M=0.0$) for fault monitoring and maintenance by the engineering personnel. For each experiment, the engine simulation is conducted at a frequency of 66.67 Hz (i.e., intersample time of 15 ms)

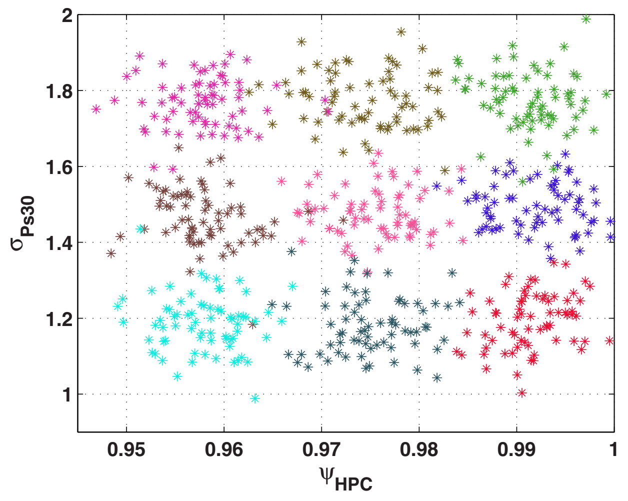

Fig. 3 Original class labels for data collection

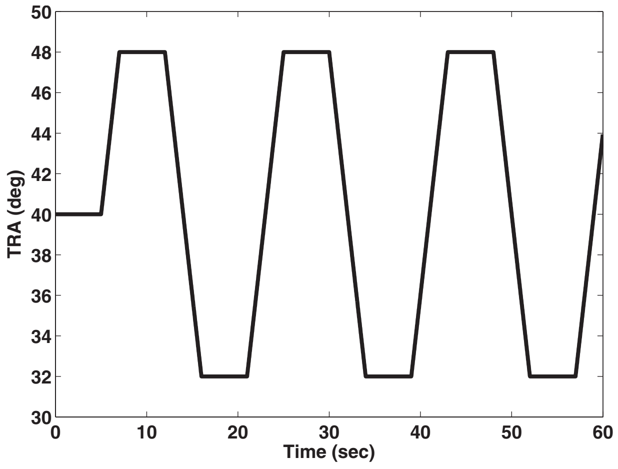

Fig. 4 Profile of throttle resolving angle (TRA)

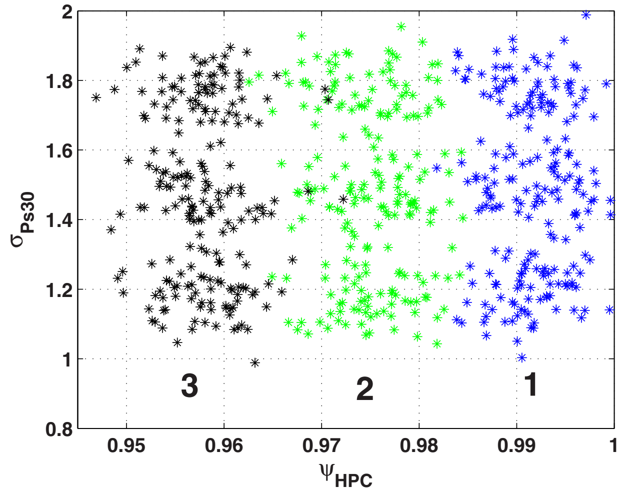

Fig. 6 Revised class assignment for fault detection

and the length of the simulation time window is 150 s, which generate 10,000 data points. Figure 5 shows representative examples of Ps_{30} time series data from each of the nine classes.

As stated earlier, the objective of the paper is to build a data-driven diagnostic algorithm that is robust to varying sensor noise level, provided that the sensor noise is within an allowable range. From this perspective, the definition of a data class is changed and made only dependent on the HPC efficiency parameters. Thus, the nine original classes are reduced to three classes as shown in Fig. 6. This is the final class assignment for the data set, where each class has $(50\times 3)=150$ training samples and $(20\times 3)=60$ testing samples.

Thus, in the above context, the problem of component-level fault detection in presence of varying sensor noise condition in aircraft gas turbine engines is formulated as a multiclass classification problem (in the present scenario, number of classes is three). Section 4 presents a brief review of SDF as a nonlinear feature extraction technique before proposing the partitioning optimization methodology.

4 SDF-Based Feature Extraction

The authors have explored the concepts of symbolic dynamics and time series data partitioning to develop a computationally efficient tool, called the SDF, for anomaly detection in complex dynamical systems [8,9]. The methodology of this symbolic feature extraction tool is reported in recent literature [12]; a brief outline of the procedure is succinctly presented here for completeness of the paper.

Symbolic feature extraction from time series data is posed as a two-time-scale problem. The *fast scale* is related to the response time of the process dynamics. Over the span of data acquisition, dynamic behavior of the system is assumed to remain invariant, i.e., the process is quasi-stationary at the fast scale. On the other hand, the *slow scale* is related to the time span over which nonstationary evolution of the system dynamics may occur. It is expected that the features extracted from the fast-scale data will depict statistical changes between two different slow-scale epochs if the underlying system has undergone a change. The method of

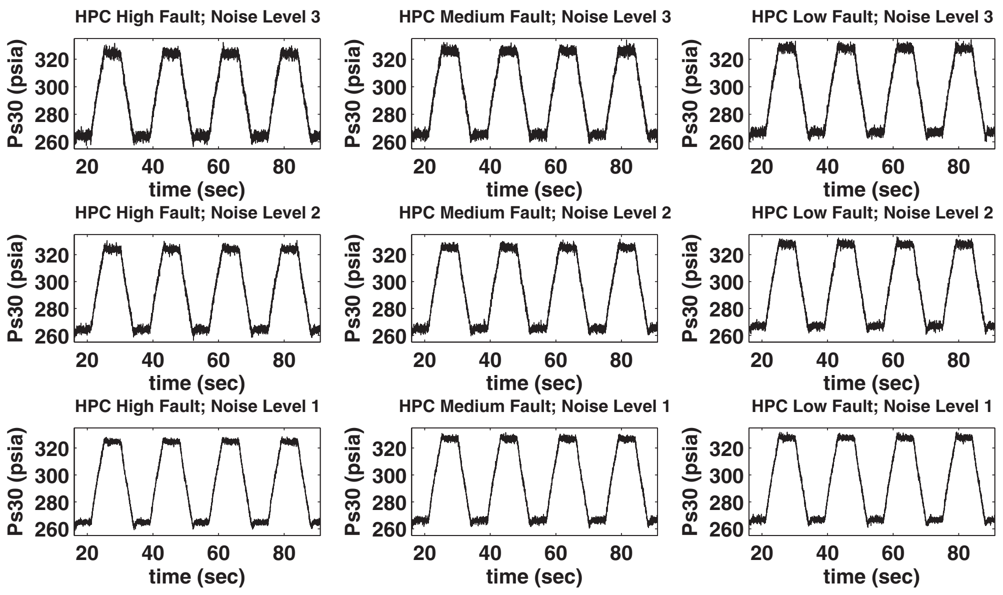

Fig. 5 Representative time series data for HPC fault and Ps_{30} degradation conditions

extracting features from stationary time series data is comprised of the following steps:

- Sensor time series data, generated from a physical system or its dynamical model, are collected at a slow-scale epoch and let it be denoted as $\mathbf{q}$. A compact (i.e., closed and bounded) region $\Omega \in \mathbb{R}^n$, where $n \in \mathbb{N}$, within which the stationary time series is circumscribed, is identified. Let the space of time series data sets be represented as $\mathcal{Q} \subseteq \mathbb{R}^{n\times N}$, where $N \in \mathbb{N}$ is sufficiently large for convergence of statistical properties within a specified threshold. While n represents the dimensionality of the time series, N is the number of data points in the time series. Then, $\{\mathbf{q}\} \in \mathcal{Q}$ denotes a time series at the slow-scale epoch of data collection.
- Encoding of Ω is accomplished by introducing a partition $\mathbb{B} \triangleq \{B_0, \ldots, B_{(|\Sigma|-1)}\}$ consisting of $|\Sigma|$ mutually exclusive (i.e., $B_j \cap B_k = \varnothing \; \forall j \neq k$) and exhaustive (i.e., $\cup_{j=0}^{|\Sigma|-1} B_j = \Omega$) cells, where each cell is labeled by symbols $\sigma_j \in \Sigma$ and $\Sigma = \{\sigma_0, \ldots, \sigma_{|\Sigma|-1}\}$ is called the alphabet. This process of coarse graining can be executed by uniform, maximum entropy, or any other scheme of partitioning. Then, the time series data points that visit the cell B_j are denoted as $\sigma_j \, \forall j = 0, 1, \ldots, |\Sigma|-1$. This step enables transformation of the time series data $\{\mathbf{q}\}$ to a symbol sequence $\{\mathbf{s}\}$, consisting of the symbols σ_j in the alphabet Σ.
- A PFSA, consisting of r states, is then constructed and the symbol sequence $\{\mathbf{s}\}$ is run through the PFSA. Thus, an $(r \times r)$ state transition matrix $\Pi \equiv [\pi_{jk}]$ is obtained at the slow-scale epoch, where $\pi_{jk} \geq 0$ is the transition probability from state j to state k of the PFSA. If the dynamical system is a Markov process, then Π is an irreducible stochastic matrix, i.e., $\Sigma_k \pi_{jk} = 1$, and the state probability vector $\mathbf{p} = [p_1 \cdots p_r]$ is the left eigenvector corresponding to the unique unity eigenvalue of Π. The $(1 \times r)$ vector $\mathbf{p}$ could be treated the extracted feature vector that is a low-dimensional representation of the dynamical system at the slow-scale epoch.

For anomaly detection using SDF, the nominal time series is partitioned by one of the classical schemes (e.g., uniform partitioning (UP) or maximum entropy partitioning (MEP)) [8,9,22]. Then using the steps described before, a low-dimensional feature vector $\mathbf{p}^{\text{nom}}$ is constructed for the nominal slow-scale epoch. Similarly, from a time series at a possibly anomalous epoch, feature vector $\mathbf{p}^{\text{off-nom}}$ is constructed using the same partitioning. Finally, a classifier is used to distinguish the nominal feature vector $\mathbf{p}^{\text{nom}}$ from its off nominal counterpart. There are plenty of choices available for design of both parametric and nonparametric classifiers in literature [23,24]. Among the parametric type of classifiers, one of the most common techniques is to consider up to two orders of statistics in the feature space. In other words, the mean feature is calculated for every class along with the variance of the feature space distribution in the training set. Then, a test feature vector is classified by using the Mahalanobis distance [25] or the Bhattacharya distance [26] of the test vector from the mean feature vector of each class. However, these methods lack in computational efficiency if the feature space distribution cannot be described by second order statistics (i.e., non-Gaussian in nature). In the present context, Gaussian feature space distribution cannot be ensured due to the nonlinear nature of the partitioning feature extraction technique. Therefore, a nonparametric classifier, such as the k-NN classifier may a better candidate for this study [23,24]; however, in general, any other suitable classifier, such as the support vector machines (SVMs) or the Gaussian mixture models (GMMs) may also be used.

5 Optimization of Partitioning

Properties and variations of transformation from the symbol space to the feature space have been extensively studied in mathematics, computer science, and especially, data mining literature. Apparently, similar efforts have not been expended to investigate partitioning of time series data to optimally generate symbol blocks for pattern classification and anomaly detection. Steuer et al. [27] reported comparison of maximum entropy partitioning and uniform partitioning; it was concluded that maximum entropy partitioning is a better tool for change detection in time series than uniform partitioning. Symbolic false nearest neighbor partitioning (SFNNP) optimizes a generating partition by avoiding topological degeneracy. However, a major shortcoming of SFNNP is that it may become extremely computationally intensive if the dimension of the phase space of the underlying dynamical system is large. Furthermore, if the time series data become noise-corrupted, the states of SFNNP rapidly grow in number and thus the partitioning may erroneously require a large number of symbols to capture pertinent information on the system dynamics [22]. This shortcoming could be largely alleviated by wavelet space partitioning (WSP) that is particularly effective for noisy data for large-dimensional dynamical systems [8]; MEP was used by Rajagopalan and Ray [9] to generate symbol blocks from time series data by WSP. Although WSP is significantly computationally faster than SFNNP and is suitable for real-time applications, WSP too has several shortcomings such as requirements of good understanding of signal characteristics for selection of the wavelet basis, identification of appropriate scales, and lossy and nonunique conversion of the two-dimensional scale-shift domain into a single dimension. Subbu and Ray [22] introduced Hilbert-transform-based analytic signal space partitioning (ASSP) as an alternative to WSP, and Sarkar et al. [28] generalized ASSP for symbolic analysis of noisy signals. Nevertheless, these partitioning techniques primarily provide a symbolic representation of the underlying dynamical system under a given quasi-stationary condition, rather than capturing the data-evolution characteristics due to a fault in the system. The partitioning optimization methodology elaborated in this section endeavors to overcome this shortcoming to make SDF, a robust data-driven feature extraction tool for pattern classification and fault detection.

Many optimization criteria have been reported for feature extraction in multi class classification problems, which are broadly classified as follows:

(1) filter method that makes use of the information content feedback (e.g., Fisher criteria, statistical dependence, and information-theoretic measures) as optimization criteria for feature extraction.
(2) wrapper method that includes the classifier inside the optimization loop to maximize the predictive accuracy (e.g., classification rate using statistical resampling or cross-validation [23]).

In this paper, the wrapper method is adopted (i.e., the classification error is minimized on the training set for optimization) primarily because of the nonbinary nature of the problem at hand and the possible non-Gaussian distribution of training samples in the feature space.

In a multiclass problem, ideally one should jointly minimize all the off-diagonal elements of the confusion matrix, while maximizing the diagonal elements. However, in that case, the dimension of the objective space blows up with increase in the number of classes, which is obviously impractical. Therefore, two costs may be defined on the confusion matrix by using another penalty weighting matrix, elements of which denote the relative penalty values for different confusions in the classification process. Formally, let there be $Cl_1, \ldots, Cl_n$ classes of labeled time series data given as the training set. A partitioning $\mathbb{B}$ is employed to extract features from each sample and a k-NN classifier $\mathbb{K}$ is used to classify them. After the classification process, the confusion matrix $\mathbf{C}$ is obtained, where the value of its element c_{ij} 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 value of its element w_{ij}

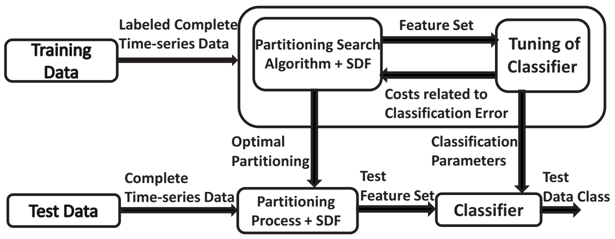
Training Data
Labeled Complete Time-series Data
Partitioning Search Algorithm + SDF
Feature Set
Tuning of Classifier
Costs related to Classification Error
Optimal Partitioning
Classification Parameters
Test Data
Complete Time-series Data
Partitioning Process + SDF
Test Feature Set
Classifier
Test Data Class

Fig. 7 General framework for optimization of feature extraction

denotes the penalty incurred by the classification process for classifying a data set from Cl_i as a data set from class Cl_j. With these definitions, the cost due to expected classification error, Cost_E is defined as

$$\text{Cost}_E = \frac{1}{N_s}\left(\sum_i \sum_j w_{ij} c_{ij}\right) \tag{1}$$

where N_s is the total number training samples including all classes. The outer sum in the above equation sums the total penalty values for misclassifying each class Cl_i. Thus, Cost_E is related to the expected classification error. Although in the current formulation, the total penalty values are equally weighted for all classes that can be changed based on prior knowledge about the data and the user 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 very accurate in practice. A natural solution to this problem is to choose a feature extractor that minimizes the worst-case classification error [29] as well. In the present setting, that cost due to worst-case classification error, Cost_W can be defined as

$$\text{Cost}_W = \max_i\left(\frac{1}{N_i}\sum_j w_{ij} c_{ij}\right) \tag{2}$$

where N_i is the number of training samples in class Cl_i. With such construction, the dimension of the objective space is not a function of the number of classes, which makes it convenient for classification with large number classes. Finally, in accordance with multi-objective optimization literature, an overall cost Cost_O is defined as a linear combination of Cost_E and Cost_W with parameter $\alpha \in [0,1]$.

$$\text{Cost}_O = \alpha \text{Cost}_E + (1-\alpha)\text{Cost}_W \tag{3}$$

Figure 7 depicts the general outline of the classification process. Labeled time series data from the training set are partitioned and the generated low-dimensional feature vectors (via symbolization and PFSA construction) are fed to the classifier. After classification, the training error cost, defined in Eq. (3), is computed and fed back to the feature extraction block. During classification, the classifier may be tuned to obtain better classification rates. For example, for k-NN classifiers [23], choice of neighborhood size or the distance metric can be tuned. The partitioning is updated 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. 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 has not been performed in this paper as the main focus here is to choose the optimal partitioning to minimize the classification error related cost.

Ideally, the optimization procedure involves construction of the Pareto front by minimizing Cost_O for different values of α and the user should choose a particular value of α as the operating point. However, for simplicity α is taken to be 1 in this paper, i.e.,

$$\text{Cost}_O = \text{Cost}_E \tag{4}$$

Thus, the optimal partitioning $\mathbb{B}^*$ is the solution to the following optimization problem:

$$\mathbb{B}^* = \arg\min_{\mathbb{B}} \text{Cost}_O(\mathbb{B}) \tag{5}$$

5.1 Optimization Procedure. This paper adopts a sequential search based optimization technique of partitioning, previously proposed in Ref. [30]. 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 in this paper. Suppose, the number of cells of the partitioning $\mathbb{B}$ is $|\Sigma|$ and the region $\Omega \in \mathbb{R}^1$ (see Sec. 4) that circumscribes the one-dimensional time series data space is identified. 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. Let the data space region Ω be divided into G grid cells, i.e., there are $G-1$ grid boundaries excluding the boundaries of Ω. Thus, there are $|\Sigma|-1$ partitioning boundaries to choose among $G-1$ possibilities, i.e., the number of elements (i.e., $|\Sigma|$-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 get significantly large with increase in values of G and $|\Sigma|$ (e.g., for $G \gg |\Sigma|$, computational complexity increases approximately by a factor of $G/|\Sigma|$ with increase in value of $|\Sigma|$ by 1). In that case, usage of a direct search approach becomes infeasible for evaluation of all possible partitioning. Therefore, this paper develops a suboptimal solution for the multi-objective optimization problem described above.

The objective space consists of the scalar valued cost Cost_O, while decisions are made in the space $\mathcal{P}$ of all possible partitionings. In the case of one-dimensional time series data, a partitioning consisting of $|\Sigma|$ cells may be succinctly represented by the $|\Sigma|-1$ points that separate the cells. In the sequel, a Σ-cell partitioning $\mathbb{B}$ is expressed as $\Lambda_{|\Sigma|} \triangleq \{\lambda_1, \lambda_2, \ldots, \lambda_{|\Sigma|-1}\}$, where $\lambda_i \ \forall i \in \{1,2,\ldots,|\Sigma|-1\}$ denotes a partitioning boundary. The overall cost is dependent on a specific partitioning Λ and is denoted by $\text{Cost}_O(\Lambda)$. This suboptimal 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

$$\lambda_1^* = \arg\min_{\lambda_1} \text{Cost}_O(\Lambda_2) \tag{6}$$

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

$$\lambda_2^* = \arg\min_{\lambda_2} \text{Cost}_O(\Lambda_3) \tag{7}$$

where $\Lambda_3 = \{\lambda_1^*, \lambda_2\}$. The optimal three-cell partitioning is obtained as $\Lambda_3^* = \{\lambda_1^*, \lambda_2^*\}$. In this (local) optimization procedure, the cell that provides the largest decrement in Cost_O on further segmentation ends up being partitioned. Iteratively, this procedure can be extended to obtain the m cell partitioning as follows:

$$\lambda_{|\Sigma|-1}^* = \arg\min_{\lambda_{|\Sigma|-1}} \text{Cost}_O(\Lambda_{|\Sigma|}) \tag{8}$$

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

This optimization procedure is monotonically decreasing in the cost function with every additional sequential operation, i.e., $\text{Cost}_O(\Lambda^*_{|\Sigma|-1})\geq\text{Cost}_O(\Lambda^*_{|\Sigma|})$. This is evident from the following argument.

Let $\Lambda^*_{|\Sigma|-1}$ be the $(|\Sigma|-1)$-cell partitioning that minimizes Cost_O. 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 and $\text{Cost}_O(\Lambda_{|\Sigma|})=\text{Cost}_O(\Lambda^*_{|\Sigma|-1})$. Since $\text{Cost}_O(\Lambda^*_{|\Sigma|})\leq\text{Cost}_O(\Lambda_{|\Sigma|})\ \forall\Lambda_{|\Sigma|}$, it follows that $\min(\text{Cost}_O(\Lambda_{|\Sigma|-1}))\geq\min(\text{Cost}_O(\Lambda_{|\Sigma|}))$. 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: $\Lambda^*_{|\Sigma|-1}$ is the optimal partitioning if

$$\text{Cost}_O(\Lambda^*_{|\Sigma|-1})-\text{Cost}_O(\Lambda^*_{|\Sigma|})\leq\eta_{\text{stop}}\tag{9}$$

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

6 Results and Discussion

This section presents pertinent results for the two case studies (HPC fault detection in Sec. 6.1 and HPT fault detection in Sec. 6.2) reported in this paper. While the problem specification for case study 1 has been provided earlier in Sec. 3.1, the same for case study 2 is provided in the sequel (see Sec. 6.2). Classification results by using the classical partitioning schemes are also provided for comparison.

6.1 Case Study 1: Fault Detection in HPC. At the beginning of the optimization procedure, a weighting matrix $\mathbf{W}$ needs to be defined to calculate the costs 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 100 grid cells, i.e., 99 grid boundaries excluding the boundaries of Ω. The sequential partitioning optimization procedure described in Sec. 5 is then employed to identify the optimal partitioning. The threshold value η_{stop} for stopping the algorithm is chosen to be 0.001 and the optimal alphabet size is found to be $|\Sigma|=4$. For SDF analysis, the depth for constructing PFSA sates is taken to be $D=1$ and features are classified by a k-NN classifier (with $k=5$) using the Euclidean distance metric. Figure 8 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 10,000 data points) into three numbers.

For comparison purpose, classical partitioning schemes, such as UP and MEP, are also used with the same alphabet size $\Sigma=4$.

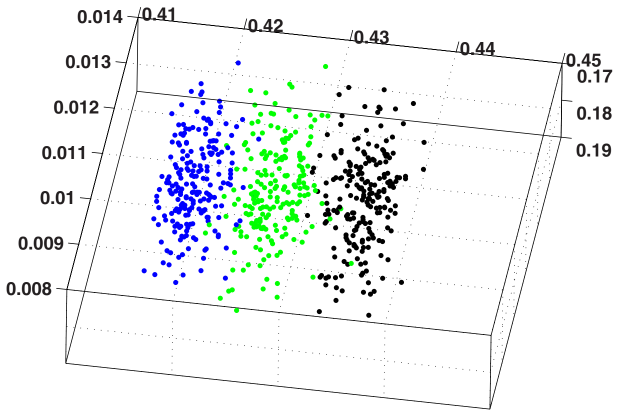

Fig. 8 Feature space of the training set using optimal partitioning

Figures 9 and 10 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 uniform partitioning and maximum entropy partitioning, respectively.

Finally, the confusion matrices for the optimal, uniform, and maximum entropy partitioning on the test data set are given by $\mathbf{C}^{\text{OptP}}_{\text{test}}$, $\mathbf{C}^{\text{UP}}_{\text{test}}$, and $\mathbf{C}^{\text{MEP}}_{\text{test}}$, respectively.

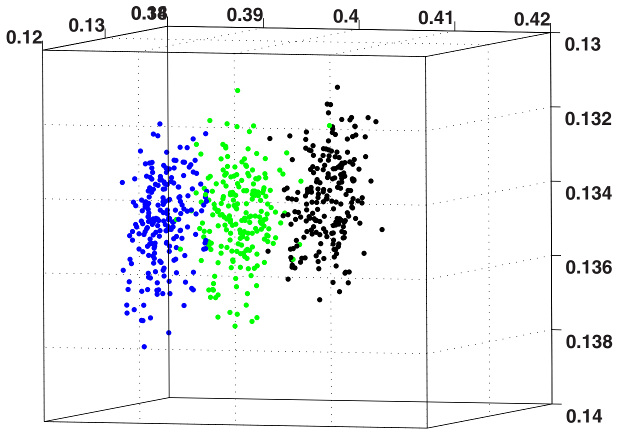

Fig. 9 Feature space of training set: uniform partitioning (UP)

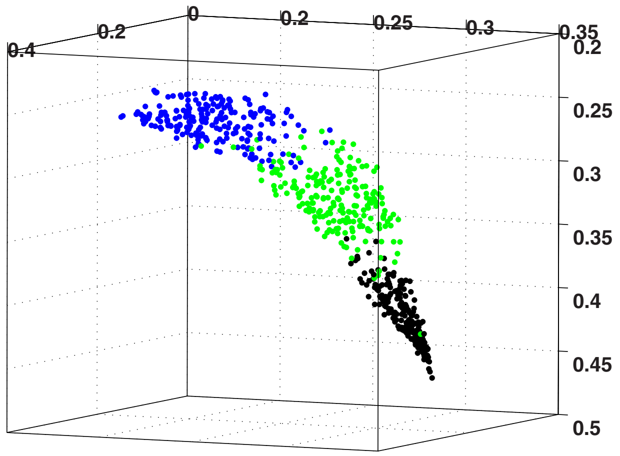

Fig. 10 Feature space of training set: maximum entropy partitioning (MEP)

Table 4 Comparison of classification performances of different partitioning schemes on test data set (60×3 samples)

Partitioning	Cost_E	Cost_W
OptP	0.0389	0.0500
UP	0.0500	0.1000
MEP	0.0833	0.1500

$$\mathbf{C}_{\text{test}}^{\text{OptP}} = \begin{pmatrix} 58 & 2 & 0 \\ 1 & 57 & 2 \\ 0 & 2 & 58 \end{pmatrix}$$

$$\mathbf{C}_{\text{test}}^{\text{UP}} = \begin{pmatrix} 58 & 2 & 0 \\ 4 & 54 & 2 \\ 0 & 1 & 59 \end{pmatrix}$$

$$\mathbf{C}_{\text{test}}^{\text{MEP}} = \begin{pmatrix} 57 & 3 & 0 \\ 7 & 51 & 2 \\ 0 & 3 & 57 \end{pmatrix}$$

Table 4 compares the values of Cost_E and Cost_W for OptP, UP, and MEP on the test set. It is interesting to notice that although the optimization is performed based on only minimization of Cost_E, both Cost_E and Cost_W have reduced compared to the classical partitioning schemes. This fact can be attributed to the nature of the current data set. Thus, HPC fault levels can be efficiently determined under varying noise levels using the SDF methodology and optimization of partitioning may potentially improve its performance over other classical partitioning techniques. However, before going into more discussion on results, the case study for fault detection in HPT is presented.

6.2 Case Study 2: Fault Detection in HPT. This subsection presents a case study of fault detection in the HPT component only by monitoring temperature sensor T_{48} that is located at HPT exit. A representative example of T_{48} time series data is shown in Fig. 11. It has been observed in the previous case study for HPC that sensor Ps_{30} observation only captures signature of change in efficiency health parameter (ψ_{HPC}) and does not capture signature of change in flow health parameters (ζ_{HPC}). In contrast, T_{48} observation captures signature of change in both efficiency health parameters (ψ_{HPT}) and in flow health parameter (ζ_{HPT}). However, ζ_{HPT} signature is much weaker compared to that of ψ_{HPT}. Therefore, this case study will investigate classification of both the fault modes.

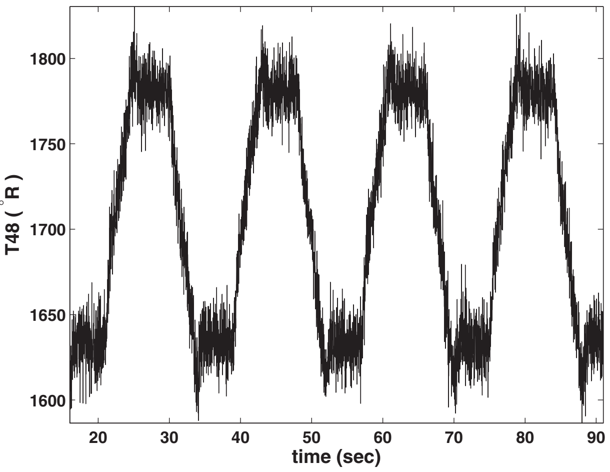

Fig. 11 Representative sensor T_{48} observation

Table 5 Fault classes in HPT

Efficiency class label	Class label	Efficiency range (ψ_{HPT})	Flow range (ζ_{HPT})
Efficiency class 1	Class 1	1.0000–0.9834	1.0000–0.9834
	Class 2	1.0000–0.9834	0.9834–0.9668
	Class 3	1.0000–0.9834	0.9668–0.9500
Efficiency class 2	Class 4	0.9834–0.9668	1.0000–0.9834
	Class 5	0.9834–0.9668	0.9834–0.9668
	Class 6	0.9834–0.9668	0.9668–0.9500
Efficiency class 3	Class 7	0.9668–0.9500	1.0000–0.9834
	Class 8	0.9668–0.9500	0.9834–0.9668
	Class 9	0.9668–0.9500	0.9668–0.9500

Table 5 shows the approximate ranges of efficiency and flow health parameters of different classes considered for the study. Similarly, different (additive) noise levels (i.e., approximate ranges of sensor noise standard deviation as percent of operating point trim values [20,21]) in T_{48} observation are shown in Table 6. Thus, in this context, there are $(9\times3)=27$ classes of data sets that need to be obtained to define a class by a HPT fault level and a noise level of the T_{48} sensor. Seventy simulation runs of the engine system were performed for each class to generate data sets for analysis, among which 50 samples are chosen as the training set and the rest of the samples are kept as the testing set. The samples are generated in the same way as in the case study 1. As different sensor noise level classes are merged into one, there are only nine classes for classification purpose with 150 training samples and 60 testing samples in each class.

The first step for classification is to define the penalty weighting matrix **W** that involves prior knowledge about the data. As shown in Table 5, for each efficiency class, there are three flow classes. However, it is observed that both efficiency and flow degradation signatures are of similar nature in time domain with efficiency degradation signatures being quantitatively stronger than the flow degradation signatures. Therefore, classification among flow classes under one efficiency level is potentially a very hard problem under the current high noise environment. Thus, the primary goal of the classification process is to classify different flow classes under different efficiency classes. The penalty weighting matrix is defined accordingly. The three diagonal blocks represent penalty values among different flow classes in one efficiency class, which are defined similarly as in the previous case study. The off-diagonal blocks represent the penalty values for different flow classes under different efficiency classes. The symmetric weighting matrix for classification process is given below.

Table 6 Variable noise levels in T_{48}

Noise level	Standard deviation range (%)
Level 1	0.45–0.55
Level 2	0.55–0.65
Level 3	0.65–0.75

Table 7 Comparison of classification performances of different partitioning schemes on test data set (60×9 samples)

Partitioning	Cost_E	Cost_W
OptP	0.5315	0.6833
UP	0.6444	0.9500
MEP	0.6370	0.8000

$$\mathbf{W} = \begin{pmatrix} 0 & 1 & 2 & 2 & 4 & 6 & 3 & 6 & 9 \\ 1 & 0 & 1 & 4 & 2 & 4 & 6 & 3 & 6 \\ 2 & 1 & 0 & 6 & 4 & 2 & 9 & 6 & 3 \\ 2 & 4 & 6 & 0 & 1 & 2 & 2 & 4 & 6 \\ 4 & 2 & 4 & 1 & 0 & 1 & 4 & 2 & 4 \\ 6 & 4 & 2 & 2 & 1 & 0 & 6 & 4 & 2 \\ 3 & 6 & 9 & 2 & 4 & 6 & 0 & 1 & 2 \\ 6 & 3 & 6 & 4 & 2 & 4 & 1 & 0 & 1 \\ 9 & 6 & 3 & 6 & 4 & 2 & 2 & 1 & 0 \end{pmatrix}$$

The sequential optimization process has been carried out for this problem and it has been found that the optimal alphabet size is 7, with the algorithm stopping threshold value, $\eta_{\mathrm{stop}}=0.005$. The confusion matrices for optimal, uniform, and maximum entropy partitioning on the test data set are given by $\mathbf{C}_{\mathrm{test}}^{\mathrm{OptP}}$, $\mathbf{C}_{\mathrm{test}}^{\mathrm{UP}}$, and $\mathbf{C}_{\mathrm{test}}^{\mathrm{MEP}}$, respectively. Table 7 shows the comparison of classification performances using different partitioning processes. However, the cost values in this table should not be compared with those in Table 4 as the range of cost values is a function of the weighting matrix used for the optimization process.

$$\mathbf{C}_{\mathrm{test}}^{\mathrm{OPT}} = \begin{pmatrix} 35 & 21 & 3 & 0 & 1 & 0 & 0 & 0 & 0 \\ 11 & 24 & 24 & 0 & 0 & 1 & 0 & 0 & 0 \\ 3 & 22 & 35 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 41 & 17 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 19 & 22 & 19 & 0 & 0 & 0 \\ 1 & 0 & 0 & 7 & 21 & 31 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 39 & 17 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 16 & 25 & 19 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 21 & 35 \end{pmatrix}$$

$$\mathbf{C}_{\mathrm{test}}^{\mathrm{UP}} = \begin{pmatrix} 32 & 22 & 5 & 0 & 0 & 1 & 0 & 0 & 0 \\ 15 & 22 & 22 & 0 & 0 & 1 & 0 & 0 & 0 \\ 3 & 17 & 40 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 37 & 16 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 21 & 15 & 24 & 0 & 0 & 0 \\ 1 & 0 & 0 & 4 & 18 & 37 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 30 & 17 & 13 \\ 0 & 0 & 0 & 0 & 0 & 0 & 21 & 21 & 18 \\ 0 & 0 & 0 & 1 & 0 & 0 & 12 & 27 & 20 \end{pmatrix}$$

$$\mathbf{C}_{\mathrm{test}}^{\mathrm{MEP}} = \begin{pmatrix} 37 & 18 & 4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 16 & 24 & 19 & 0 & 0 & 1 & 0 & 0 & 0 \\ 5 & 21 & 34 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 30 & 23 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 17 & 23 & 20 & 0 & 0 & 0 \\ 1 & 0 & 0 & 7 & 19 & 33 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 30 & 18 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 23 & 21 & 16 \\ 0 & 0 & 0 & 1 & 0 & 0 & 12 & 18 & 29 \end{pmatrix}$$

It is observed from these results that samples of different flow classes under different efficiency classes are efficiently classified. But as expected, the classification performance degrades for different flow classes under one efficiency level. However, the optimization process improves the performance of SDF compared to the classical partitioning methods. With a close inspection of the confusion matrix corresponding to the optimal partitioning, it can be observed that the confusion between low-level faults and high-level fault classes for HPT flow under same efficiency level are significantly reduced by the partitioning optimization.

For some problems, the classical partitioning schemes may perform as well as 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 function allows problems with a large number of classes, its upper limit could be constrained by the alphabet size used for partitioning. The complexity of a PSFA model, as obtained from time series data, is related to the number of states (or the number of partitions) in the PSFA. Thus, a small partitioning alphabet size alleviates the issue of over-training that could arise as a result of optimization performed on the training set.

7 Summary, Conclusions, and Future work

This article presents a SDF-based methodology for data-driven detection of component-level faults in aircraft gas turbine engines under varying sensor noise condition. In this context, time series data partitioning in the SDF methodology is viewed as nonlinear feature extraction technique that is to be optimized. The task of detection is viewed as a multiclass classification problem and feature extraction has been optimized to enhance the classification rate. The proposed methodology intelligently compresses a high-volume database to execute fault detection for a large number of classes. It has been shown that, using partitioning optimization, the classification rate can be improved beyond the performance of classical partitioning techniques.

Scalability is a critical issue for data-driven supervised methods of fault detection. A natural way to circumvent this problem is to perform fault isolation before estimating a particular fault level [12,13]. Furthermore, although this paper has shown application of the developed feature extraction method to component fault detection problems, the technique is general enough to be used for detecting sensor and actuator faults under different architectures.

The fault detection method described in this paper identifies the health status of an engine component at a particular slow-scale epoch. Now, in a real life scenario, faults need to be distinguished from the usual gradual degradation of a component. However, to achieve this goal, the health status must be monitored over several slow-time epochs. Given a temporal profile of the health status identified by the current method, faults can be distinguished from the usual degradation. Typically, the engine health deteriorates at a slow rate for usual degradation while a fault is the cause of an abrupt change in health status or degradation at a comparatively faster rate.

Although the method presented in the paper can perform in real time, various operating conditions need to be investigated for its on-board (in-flight) application. The authors are currently pursuing research for developing a semantic system identification technique, which extends the present method for on-board application (i.e., robust to variable operating conditions). Apart from this important aspect, currently being pursued as well before testing the methodology on a real life engine testbed are the following research areas:

- investigation of effectiveness of cost related to worst-case classification error, Cost_W by using various values of parameter α other than 1 (see Sec. 5)
- use of other classifiers (e.g., support vector machines) and

comparison of performances among different classifiers

- inclusion of the step of tuning the classifier inside the optimization loop as described in the general framework shown in Fig. 7
- extension of the methodology to accommodate other types of sensor degradation, such as bias and drifting
- investigation of situations with simultaneous faults in multiple engine components, actuators, and sensors
- development of a comprehensive information fusion framework to generate composite patterns from atomic patterns in individual sensors [31,32]
- comparative evaluation of SDF with other pattern recognition techniques under sensor degradation

Acknowledgment

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

References

[1] Kobayashi, T., and Simon, D. L., 2003, "Application of a Bank of Kalman Filters for Aircraft Engine Fault Diagnostics," Turbo Expo, Atlanta, GA, Jun. 16–19.

[2] Kobayashi, T., and Simon, D. L., 2007, "Hybrid Kalman Filter Approach for Aircraft Engine In-Flight Diagnostics: Sensor Fault Detection Case," ASME J. Eng. Gas Turbines Power, **129**, pp. 746–754.

[3] Simon, D., and Simon, D. L., 2005, "Aircraft Turbofan Engine Health Estimation Using Constrained Kalman Filtering," ASME J. Eng. Gas Turbines Power, **127**, pp. 323–328.

[4] Volponi, A. J., Brotherton, T., Luppold, R., and Simon, D. L., 2003, "Development of an Information Fusion System for Engine Diagnostics and Health Management," 39th Combustion/27th Airbreathing Propulsion/21st Propulsion Systems Hazards/Third Modeling and Simulation Joint Subcommittee Meeting, Colorado Springs, CO, Dec. 1–5, Paper No. NASA/TM2004-212924.

[5] Chu, E., Gorinevsky, D., and Boyd, S., 2010, "Detecting Aircraft Performance Anomalies From Cruise Flight Data," AIAA Infotech Aerospace Conference, Atlanta, GA.

[6] Park, S., and Himmelblau, D. M., 1983, "Fault Detection and Diagnosis via Parameter Estimation in Lumped Dynamic Systems," Ind. Eng. Chem. Process Des. Dev., **22**(3), pp. 482–487.

[7] Saxena, A., Goebel, K., Simon, D., and Eklund, N., 2008, "Damage Propagation Modeling for Aircraft Engine Run-To-Failure Simulation," *Proceedings of the International Conference on Prognostics and Health Management (PHM08)*, Denver, CO.

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

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

[10] Gupta, S., Ray, A., and Keller, E., 2007, "Symbolic Time Series Analysis of Ultrasonic Data for Early Detection of Fatigue Damage," Mech. Syst. Signal Process., **21**(2), pp. 866–884.

[11] Rao, C., Ray, A., Sarkar, S., and Yasar, M., 2009, "Review and Comparative Evaluation of Symbolic Dynamic Filtering for Detection of Anomaly Patterns," Signal, Image and Video Processing, **3**(2), pp. 101–114.

[12] Gupta, S., Ray, A., Sarkar, S., and Yasar, M., 2008, "Fault Detection and Isolation in Aircraft Gas Turbine Engines: Part I—Underlying Concept," Proc. Inst. Mech. Eng., Part G: Journal of Aerospace Engineering, **222**(3), pp. 307–318.

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

[14] Sarkar, S., Rao, C., and Ray, A., 2009, "Statistical Estimation of Multiple Faults in Aircraft Gas Turbine Engines," Proc. Inst. Mech. Eng., Part G: Journal of Aerospace Engineering, **223**(4), pp. 415–424.

[15] Larson, E. C., Parker, B. E., Jr., and Clark, B. R., 2002, "Model-Based Sensor and Actuator Fault Detection and Isolation," *Proceedings of the American Control Conference*, Vol. 5, pp. 4215–4219.

[16] 2007, C-MAPSS: Commercial Modular Aero-Propulsion System Simulation, NASA GRC Software Repository.

[17] Frederick, D. K., DeCastro, J. A., and Litt, J. S., 2007, User's Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS), NASA/TM2007-215026.

[18] Kobayashi, T., and Simon, D. L., 2001, "A Hybrid Neural Network-Genetic Algorithm Technique for Aircraft Engine Performance Diagnostics," 37th Joint Propulsion Conference and Exhibit cosponsored by the AIAA, ASME, SAE, and ASEE, Salt Lake City, UT.

[19] Koushanfar, F., Potkonjak, M., and Sangiovanni-Vincentelli, A., 2003, "On-Line Fault Detection of Sensor Measurements," *Sensors, Proceedings of IEEE*, Vol. 2, pp. 974–980.

[20] Simon, D. L., and Garg, S., 2010, "Optimal Tuner Selection for Kalman Filter-Based Aircraft Engine Performance Estimation," ASME J. Eng. Gas Turbines Power, **132**(3), pp. 031601.

[21] Lu, P., Zhang, M., Hsu, T., and Zhang, J., 2001, "An Evaluation of Engine Faults Diagnostics Using Artificial Neural Networks," ASME J. Eng. Gas Turbines Power, **123**(2), pp. 340–346.

[22] Subbu, A., and Ray, A., 2008, "Space Partitioning via Hilbert Transform for Symbolic Time Series Analysis," Appl. Phys. Lett., **92**(8), p. 084107.

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

[24] Duda, R., Hart, P., and Stork, D., 2001, *Pattern Classification*, Wiley, New York.

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

[26] Choi, E., and Lee, C., 2003, "Feature Extraction Based on the Bhattacharyya Distance," Pattern Recognition, **36**, pp. 1703–1709.

[27] Steuer, R., Molgedey, L., Ebeling, W., and Jimenez-Montano, M., 2001, "Entropy and Optimal Partition for Data Analysis," Eur. Phys. J. B, **19**, pp. 265–269.

[28] Sarkar, S., Mukherjee, K., and Ray, A., 2009, "Generalization of Hilbert Transform for Symbolic Analysis of Noisy Signals," Signal Process., **89**(6), pp. 1245–1251.

[29] Alaiz-Rodriguez, R., Guerrero-Curieses, A., and Cid-Sueiro, J., 2005, "Minimax Classifiers Based on Neural Networks," Pattern Recognition, **38**(1), pp. 29–39.

[30] Jin, X., Sarkar, S., Mukherjee, K., and Ray, A., 2009, "Suboptimal Partitioning of Time-Series Data for Anomaly Detection," Conference on Decision and Control, Shanghai, China.

[31] Romessis, C., and Mathioudakis, K., 2006, "Bayesian Network Approach for Gas Path Fault Diagnosis," ASME J. Eng. Gas Turbines Power, **128**(1), pp. 64–72.

[32] Basir, O., and Yuan, X., 2007, "Engine Fault Diagnosis Based on Multi-Sensor Information Fusion Using Dempster–Shafer Evidence Theory," Inf. Fusion, **8**(4), pp. 379–386.