Symbolic Identification for Anomaly Detection in Aircraft Gas Turbine Engines

Subhadeep Chakraborty Soumik Sarkar Asok Ray Shashi Phoha
szc138@psu.edu szs200@psu.edu axr2@psu.edu sxp26@psu.edu

The Pennsylvania State University
University Park, PA 16802

***Abstract*— This paper presents a robust and computationally inexpensive technique of fault detection in aircraft gas-turbine engines, based on a recently developed statistical pattern recognition tool. The method involves abstraction of a qualitative description from a general dynamical system structure, using state space embedding of the output data-stream and discretization of the resultant pseudo state and input spaces. The system identification is achieved through grammatical inference techniques, and the deviation of the plant output from the nominal estimated language gives a metric for fault detection. The algorithm is validated on a numerical simulation test-bed that is built upon the NASA C-MAPSS model of a generic commercial aircraft engine.**

Index Terms: *Anomaly Detection, Symbolic Dynamics, System Identification, Fixed Structure Automata, CMAPSS*

I. Introduction

The modern aircraft propulsion system showcases a coalition of a vast number of complex sub-components. Over the decades, health monitoring of aircraft propulsion systems has evolved to be an issue of paramount importance. However, the inherent complexity and uncertainty in this system pose a challenging problem to health monitoring, since first principle models, if available, are commonly oversimplified lump-parameter models, or in worst cases models may not be available at all.

For the purpose of efficient health monitoring of complex interconnected systems, an on-board real-time robust fault detection tool is necessary. Recent research has extensively explored the problem of anomaly detection using symbolic dynamic filtering (*SDF*) [1]–[4]. But, apparently the system identification aspect of the health monitoring problem has not received much attention for systems that are composed of many smaller components. Since human-engineered multi-component systems are usually interconnected physically as well as through the use of feedback control loops, the effect of any one component degradation may affect the input streams to the remaining components. Also, in most practical situations, the system might need to operate in different operating regimes and thus under diverse input conditions.

It is evident, that in the absence of a high-fidelity component-level model of the system, the major challenges here are detection and isolation of faults by developing a description of the dynamical system purely from the input/output characteristics, in such a way, that it should not only be sensitive to changes in the parameters of the actual dynamical system but also invariant with changes in the operating/input conditions.

The purpose of the work reported in this paper is to address this issue, and develop a robust and computationally inexpensive system identification technique based on formal language formulation, which achieves the above-mentioned objectives.

A central step in this kind of identification methodology is discretization of the raw time-series measurements into a corresponding sequence of symbols. An important practical advantage of working with symbols is increased computational efficiency [1], [2]. The proposed method is designed to be robust with respect to sensor noise, and also simple enough to be embedded in the sensors themselves. Thus, it facilitates construction of a reliable sensor network to serve as a backbone to higher levels in the decision-making hierarchy of large-scale complex systems.

The method of real-time diagnostic and fault estimation technique proposed in this paper has been validated on the C-MAPSS (Commercial Modular Aero Propulsion System Simulation), which is a tool for simulating a realistic large commercial turbofan engine. This set-up is particularly relevant for testing condition-monitoring algorithms, since it allows users to choose and design operational profiles, controllers, environmental conditions, thrust levels, etc. to simulate a scenario of interest. Most importantly, it allows the user to tune efficiency and flow parameters to simulate specific fault modes.

II. Problem Description

To apply grammatical inference procedures to identification of non-autonomous dynamical systems, a dynamical system is considered as an entity (i.e., linguistic source) capable of generating a specific language. Almost all physical processes can be represented by dynamical systems operating in continuous time and in continuous state space. It is evident that estimation of deviation of such a system from its nominal operating condition with grammatical inference techniques need to be decomposed into three main tasks:

A. Abstraction

The first task is abstracting a discrete qualitative counterpart of the general dynamical system representing the physical process.

B. Identification

The learning task is to identify a ‘correct’ grammar for the unknown target language, given a finite number of examples of the language.

C. Comparison

On completion of "identification", the output of the abstracted system can be compared to the actual output in the sense of some suitably defined metric for detection of any kind of off-nominality in the system under investigation.

The next section develops these ideas in a systematic manner, with reference to a CMAPSS model undergoing a slow degradation in its fan efficiency.

III. Theoretical Background

Definition The underlying structure of a dynamical system, can be represented by a General Dynamical System (*GDS*), defined as an 8-tuple (see [5] for details)

$$\boldsymbol{D} = (T, U, W, Q, P, f, g, \leq) \tag{1}$$

where

- T is a time set,
- U and W are input and output sets respectively,
- Q are inner states,
- P denotes the input process $P : T \to X$,
- f denotes the global state transition

$$f : T \times T \times Q \times P \to Q \quad \text{time-varying system} \tag{2}$$
$$f : T \times Q \times P \to Q \quad \text{time-invariant} \tag{3}$$

- g denotes the output function

$$g : T \times Q \to W \quad \text{time-varying system} \tag{4}$$
$$g : Q \to W \quad \text{time-invariant} \tag{5}$$

Let $\boldsymbol{D}_i$ be a dynamical system indexed by i representing different parametric conditions, $\boldsymbol{D}_0$ being the nominal or healthy condition of the system, and $i = 1, 2, ...$ signifying deteriorating health conditions of the plant due to a progressing anomaly. Let $\boldsymbol{U}_k, \quad k = 1, 2, ..., \mathcal{K}$ be $\mathcal{K}$ different input conditions, $\boldsymbol{y}_k^i$ be the output from the i^{th} system $\boldsymbol{D}_i$ receiving the k^{th} input $\boldsymbol{U}_k$.

Let $\mathcal{G}$ be the grammatical representation (also called a Qualitative Dynamical System (*QDS*)) of the nominal plant $\boldsymbol{D}_0$.

Definition The quantized abstraction of the *GDS*, called a Qualitative Dynamical System (*QDS*) can be represented as a 5-tuple

$$\mathcal{G} = \{\mathcal{Q}, \Lambda, \Sigma, \delta, \gamma\} \tag{6}$$

where,

- $\mathcal{Q}$ is the finite set of qualitative states of the automaton, i.e. $\mathcal{Q} = \{q_1, q_2, ..., q_f\}$.
- $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_m\}$, is the set of qualitative input events.
- $\Sigma = \{\sigma_1, \sigma_2, ..., \sigma_n\}$ is the set of output alphabets, where the output symbols are one-to-one with the quantized values of output from the dynamical system.
- $\delta : \mathcal{Q} \times \Lambda \to \mathcal{Q}$ is the state transition function which maps the current state into the next state on receiving the input λ. The transition function can also be stochastic in which case, $\delta : \mathcal{Q} \times \Lambda \to Pr\{\mathcal{Q}\}$
- $\gamma : \mathcal{Q} \to \Sigma$ is the output generation function which determines the output symbol from the current state. In its full generality, γ can be stochastic as well, i.e. $\gamma : \mathcal{Q} \to Pr\{\Sigma\}$

Let $\boldsymbol{\chi}$ denote a set of qualitative abstraction functions

$$\boldsymbol{\chi} : \boldsymbol{D}_0 \to \mathcal{G} \tag{7}$$

It may be noted that $\boldsymbol{\chi}$ is a 3-tuple, consisting of three individual abstraction functions.

$$\chi = (\chi_{TQX}, \chi_Q, \chi_W), \quad \text{where}$$

$$\chi_{TQX} : T \times Q \times X \to \Lambda \tag{8}$$
$$\chi_Q : Q \to \mathcal{Q} \tag{9}$$
$$\chi_W : W \to \Sigma \tag{10}$$

Kokar [5] introduced a set of necessary and sufficient conditions, or 'consistency postulates' that the pair $\mathcal{G}, \boldsymbol{\chi}$ must satisfy in order to be a valid representation of the general dynamical system. In this paper, since the transfer of the QDS, δ is probabilistic, the consistency postulates have been redefined in a probabilistic sense. The modified consistency postulates can be stated as follows:

Definition Let $\mathcal{D}$,$\mathcal{G}$ and $\boldsymbol{\chi}$ represent a GDS, QDS and an abstraction function respectively. Then the pair $(\mathcal{G}, \boldsymbol{\chi})$ form a consistent representation in a probabilistic sense if, $\forall q, x, t$,

$$\gamma(\chi_Q(q)) = \chi_W(g(q)) \tag{11}$$
$$\chi_Q\left(f(t, q, x)\right) \sim \delta\left(\chi_Q(q), \chi_{TQX}(t, q, x)\right) \tag{12}$$

where $X \sim P$ means the random variable X is distributed according to the probability distribution P.

A. Abstraction

Theorem 3.1 (Kokar [5]): Let $W_\pi = W_1, ..., W_n$ be a finite partition of a GDS's output space W, given by $\chi_W^{-1} : \Sigma \to W_\pi$. Let Q_π describe a partition of Q defined as an inverse image of W_π through g,

$$Q_\pi = g^{-1}(W_\pi),$$

and let TQX_π describe a partition of $T \times Q \times X$ defined as an inverse image of Q_π through f,

$$TQX_\pi = f^{-1}Q_\pi.$$

Then Q_π is a maximal admissible partition of Q, and TQX_π is an admissible partition of $T \times Q \times X$ [5].

If the system model, i.e the equations governing the general dynamical system is known, the critical hypersurfaces or partitions can be analytically evaluated using theorem 3.1 and utilized as delineated in the preceding section.

However, in the absence of model equations, this scheme is of little practical use, unless

1) there is an alternate means of constructing the phase space purely from output, without using the model equations,

2) there is an alternate means of arriving at the proposed partition without information about the state transition function f and the output function g.

The next two subsections delineate a method for achieving these ends in an approximate way.

1) Phase Space Construction: Starting from the output signal captured by suitable instrumentation, a pseudo phase space can be constructed from delay vectors using Taken's theorem [6]. The embedded phase space can be denoted by

$$\boldsymbol{x}(k) = [x_{k-\tau}, ..., x_{k-m\tau}],$$

where τ is the time lag, and m is the embedding dimension. Takens' theorem guarantees that, at least in the noise-free case, a system of state dimension s may be embedded using a maximum of m_T lags where $m_T \geq 2s + 1$.

In order to find optimum values of the embedding parameters m and τ, the literature reports many optimization routines. In this case, following [7] a differential entropy based method is used to simultaneously estimate the optimal set of embedding parameters (m^*, τ^*). The method employs a single criterion - the entropy ratio between the phase space representation of a signal and an ensemble of its surrogates.

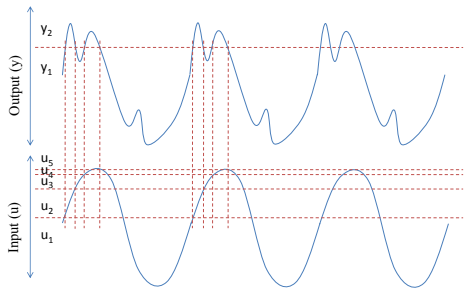


Fig. 1. Partitioning Scheme

2) Partitioning: Time series sensor data are obtained from the input and output data streams of the dynamical system $\mathcal{D}_0$ under nominal condition under different input conditions. Let $\mathcal{Y} = \{y_1, y_2, ...\}, \quad y_k \in \Sigma$ denote the discretized output sequence. A *D*-Markov machine is next constructed, with states defined by symbol blocks of length D from $\mathcal{Y}$. The reader is referred to references [1] and [2] for an in-depth description of the procedure.

The state space is constructed from the output space using Taken's theorem as discussed in the last section. In the very next step, this phase space and the input space are individually discretized. The crux of the method is to place the partitions in such a way, that there is a change in alphabet in both, at the exact same instant there is a change in the already discretized output sequence. The phase space and input variables hold the last symbol till there is a change in the output state sequence. Synchronization of input and output is essential for this step. Periodicity, or at least semi-periodicity guarantees that the number of phase space and input symbols will not explode. The partitioning scheme is illustrated in Fig. 1.

Remark A partition constructed this way is admissible, but not maximal, since this partition is a subpartition of the original partition proposed in Theorem 3.1.

Let $\mathcal{U} = \{u_{(1)}, u_{(2)}, ...\}$ denote the discretized input data sequence. Similarly let $\mathcal{Q} = \{q_{(1)}, q_{(2)}, ...\}$ denote the discretized state variable sequence. It may be noted that the state space can be multi-dimensional depending on the embedding dimension m^*.

Once the input and state space are both discretized, they can be combined to form the discretized augmented input space $\Lambda = \{\lambda_{(1)}, \lambda_{(2)}, ...\}$, where each $\lambda_{(i)} = \{q_{(i)}, u_{(i)}\}$.

The transition function used in the current methodology has been designed to be stochastic. $\delta : \mathcal{Q} \times \Lambda \rightarrow Pr\{\mathcal{Q}\}$ gives the probability distribution of transition from state q_i to $\{q_1, q_2, .., q_f\}$ on receiving an input λ_j. A grammar constructed in this way has the advantage over the context sensitive grammar described in [8] in that, the number of production rules may become inconveniently large in case of a context sensitive grammar.

However, the function $\gamma : Q \rightarrow \Sigma$, which maps the current state q_i to the current output symbol σ_i is completely deterministic. This is really an artifact of the state construction procedure [1].

B. Identification

It is assumed that inputs and outputs are time-synchronized. The state transition function δ can be expanded into two dimensional matrices δ^{λ_i}, indexed by the input variable alphabets. That means

$$\delta = \{\delta^{\lambda_1}, \delta^{\lambda_2}, ..., \delta^{\lambda_m}\} \tag{13}$$

where $\delta^{\lambda_i} : \mathcal{Q} \times \lambda_i \rightarrow Pr\{\mathcal{Q}\}$ maps the current state and input to the probability distribution over all possible states. The algorithm for estimating the matrices δ^{λ_i} is straightforward and involves counting the frequency of each transition in the training phase. Since the state transition matrices are constructed simply by counting, this method is ideal for implementing in the sensor electronics for real-time prognoses.

The training algorithm has to make sure that the probability values of δ^i converge. The convergence depends on the length of the input-output symbol sequences. In this work, a stopping rule [1] has been used for detecting the optimal data length. Stationarity of the underlying process in the slow time scale [1] is assumed and essential for this step.

In the training phase, it has to be ensured that the grammar $\mathcal{G}$ is trained with sufficient input data belonging to a particular equivalence class. This is the so-called *coverage problem.*

C. Anomaly Detection Scheme

Figure 2 gives a schematic representation of the anomaly detection philosophy. Input and output time series data from the actual plant is discretized to form symbol sequences and is fed to the trained fixed structure automaton. The discretization should be performed using the same partitioning as was done during the training phase.

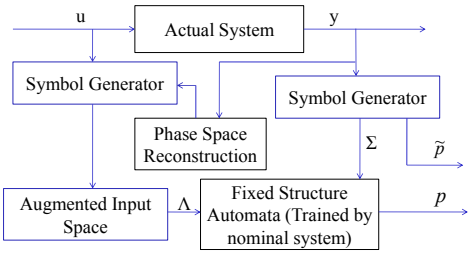


Fig. 2. General Anomaly Detection Scheme

It should be noted that, the FSA uses the output from the actual system in addition to the input, and hence cannot serve as an independent 'system identification' procedure in the classical sense of the term. The automata can serve as a system emulator only when the state transition function δ is fully deterministic, i.e. given the current state q_j and the current input symbol λ_i,

$$\delta(q_j, \lambda_i) = \begin{bmatrix} p_{q_1} & p_{q_2} & \dots & p_{q_{|\Sigma|}} \end{bmatrix}^T \quad (14)$$

$$\text{where} \quad p_{q_k} = 1 \quad \text{for one and only one } k \quad (15)$$

$$= 0 \quad \text{otherwise} \quad (16)$$

In the current scheme, the state transition probability vectors $\pi_{q_j}^{\lambda_i}$, which are the rows of the state transition matrix δ, serve as feature vectors, and are used for the purpose of anomaly detection. An extremely convenient feature of using state transition probabilities as feature vectors, and using stochastic methods to define distances between nominal and off-nominal behavior of plants is that this technique is very robust to noise.

In the present study, a novel intuitive **Pseudo-Learning Technique** of utilizing the stochastic state transition function δ is proposed, for the purpose of Anomaly Detection. In this method, the actual state transitions inside the fixed-structure automaton during the anomaly detection phase occur according to the output symbol sequence obtained from the actual system, but at each instance of state transition, the trained automaton produces a *State Transition Probability vector* π_n [9], which is characteristic of the nominal system.

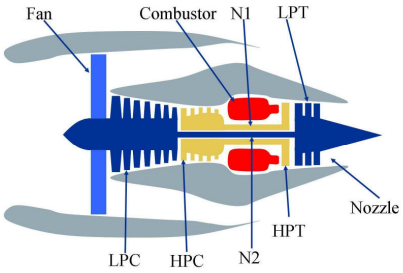
It may be noted, that the pattern vector π_n, produced by the trained automaton, is characteristic of the nominal behavior of the plant given the past history of input, state and output. The current (possibly off-nominal) condition of the plant is characterized by another state probability vector $\tilde{\pi}_n$. This is defined for the actual system output at an instant n, for which only one element of the vector will be 1, rest are zeros. The next step is to use the sequences of instantaneous State Probability vectors $\{\boldsymbol{\pi}_n\}$ and $\{\tilde{\boldsymbol{\pi}}_n\}$ obtained at each time instant, to construct a pattern vector. This is followed by calculation of mean State Probability vectors $\boldsymbol{p}$ and $\tilde{\boldsymbol{p}}$ from the collections $\{\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, ..., \boldsymbol{\pi}_n\}$ and $\{\tilde{\boldsymbol{\pi}}_1, \tilde{\boldsymbol{\pi}}_2, ..., \tilde{\boldsymbol{\pi}}_n\}$ respectively over time instants 1,2,...,n. It may be noted that in an ideal case, $\boldsymbol{p}$ should converge to $\tilde{\boldsymbol{p}}$, while they should start to diverge from each other as the fault progresses. Thus the difference of these two probability vectors, $\boldsymbol{p} - \tilde{\boldsymbol{p}}$ is a natural choice for constructing the pattern vector corresponding to that specific fault condition.

Once the pattern vectors for a fault condition are obtained, a suitable classification algorithm, such as a support vector machine can be utilized to create the hyperplane separating the nominal patterns from the possibly off-nominal pattern vectors.

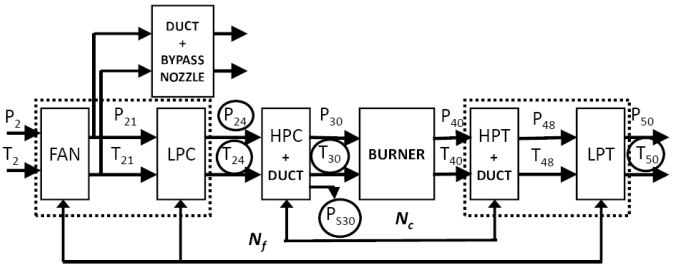
IV. Validation on the C-MAPSS test-bed

The C-MAPSS simulation test-bed, developed at NASA, is built upon the model of a commercial-scale two-spool turbofan engine and its control system. While the details of the model are available in [10], a brief outline of C-MAPSS is provided here for completeness of the paper. The engine under consideration produces a thrust of approximately 400,000 N and is designed for operation at (i) altitudes from sea level up to 12,200 m, (ii) Mach numbers from 0 to 0.90, and (iii) sea-level temperatures from approximately -50^oC to 50^oC. The throttle resolving angle (TRA) can be set to any value in the range between $0°$ (minimum power) and $100°$ (maximum power).

As seen in Fig. 3(a) and 3(b), the simulation test-bed of the gas turbine engine system consists of high pressure compressor (HPC), combustor, and high pressure turbine (HPT), which form the core of the engine model; this subsystem is also referred to as the gas generator. In the turbofan engine, the engine core is surrounded by the fan and Low pressure compressure (LPC) in the front and an additional low pressure turbine (LPT) at the rear; and fan, LPC and LPT are mechanically connected by an additional shaft. The hot exhaust gas, called the core airflow, passes through the core and LPT and then exits through the nozzle; and the rest of the incoming air passes through the fan and bypasses, or flows around the engine.



(a) Gas turbine engine schematic



(b) Gas turbine engine model configuration

Fig. 3. C-MAPSS engine simulation test-bed

A gain-scheduled control system is incorporated in the engine system, which consists of (i) a fan-speed controller for a specified throttle-resolver angle (TRA), (ii) three high-limit regulators for maintaining core-spool speed, engine-pressure ratio, and HPT exit temperature, (iii) the fourth limit regulator for the HPC exit static pressure, (iv) acceleration and deceleration limiters for the core-spool speed, and (v) a comprehensive logic structure that integrates these control-system components. Given the inputs of TRA, altitude (a) and Mach number (M), the interactively controlled component models at the simulation test-bed compute nonlinear dynamics of real-time turbofan engine operation. Both steady-state and transient operations are simulated in the continuous-time setting. The entire test-bed code is written on Matlab and Simulink platform.

This paper addresses estimation of those faults that cause efficiency degradation in engine components. In the current configuration of the C-MAPSS simulation test-bed, there are 13 health parameter inputs, namely, efficiency health parameters (ψ), flow health parameters (ζ) and pressure ratio modifiers, that simulates the effects of faults and/or degradation in the engine components.

For the engine's five rotating components (i.e., Fan, LPC, HPC, HPT and LPT), the ten health parameters are: (a) fan (ψ_F, ζ_F), (b) low pressure compressure (ψ_{LPC}, ζ_{LPC}), (c) high pressure compressor (ψ_{HPC}, ζ_{HPC}), (d) high pressure turbine (ψ_{HPT}, ζ_{HPT}), and (e) low pressure turbine (ψ_{LPT}, ζ_{LPT}). Out of these, the fan efficiency (ψ_F) and flow modifier (ζ_F) has been selected as the health parameter for fault detection in this paper. The LPC exit/ HPC inlet pressure and temperature sensors P_{24} and T_{24} respectively have been used for monitoring this fault.

A. Details of the Experiment

Time series data have been collected for different sensors under persistent excitation of TRA inputs that have truncated triangular profiles. To simulate different operating conditions, each TRA input profile has been designed to have a wide range of mean values, amplitude and frequency of excitation. Specifically, the algorithm has been tested for a mean TRA angle of 40^o, 60^o and 80^o, with amplitude ranging from $\pm 1^o$, $\pm 2^o$ and $\pm 3^o$ and the frequency of input excitation varying between $0.1Hz$, $0.06Hz$ and $0.04Hz$. Also the effect of altitude and aircraft speed have been taken into account by collecting data while the aircraft is at sea-level (i.e. altitude $a = 0.0$, Mach number $M = 0.0$) when the engine is on the ground for fault monitoring and maintenance by the engineering personnel, as well as when it is in flight at $10000ft$ with Mach number $M = 0.3$. So the entire training set comprises of $3 \times 3 \times 3 \times 2 = 54$ operating conditions

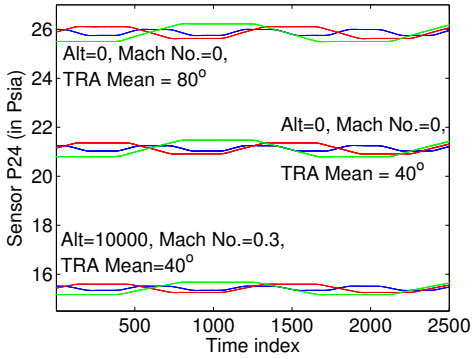


Fig. 4. Sensor reading at several operating conditions; in each set, blue implies input TRA oscillation amplitude = $\pm 1^o$ and frequency = $0.1Hz$; red implies amplitude = $\pm 2^o$ and frequency = $0.06Hz$; green implies amplitude = $\pm 3^o$ and frequency = $0.04Hz$;

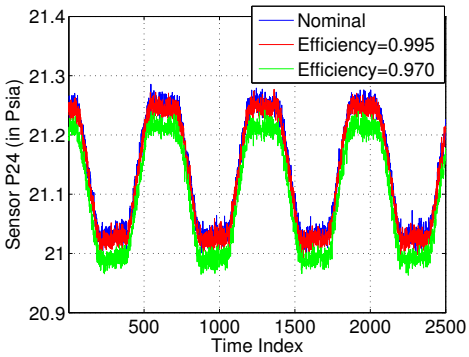


Fig. 5. Effect of fan efficiency degradation on the P24 sensor data. For all three fault conditions, SNR=40 dB

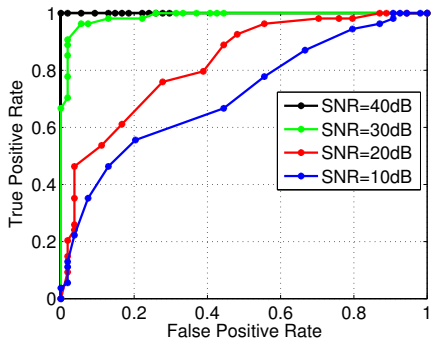
The engine simulation is conducted at a frequency of 66.67 Hz (i.e., inter-sample time of $15ms$) and the length of the simulation time window is 150 seconds, which generate $10,000$ data points for each training or test case, out of which the last $8,000$ data points are used to reduce the effects of initial transience.

An engine component C is considered to be in nominal condition when both ψ_C and ζ_C are equal to 1. Fault is injected in the fan by simultaneously reducing both ψ_C and ζ_C by same amount in the results reported in this paper. For both training (i.e., forward problem) and testing (i.e., inverse problem), time series data from all sensors are generated with ψ_F and ζ_F ranging from 1.0 to 0.96 (i.e., 4% relative loss in fan efficiency) in steps of 0.005. Figure 5 shows the effect of efficiency degradation on a relatively noisy ($SNR = 40dB$) pressure signal ($P24$) at the outlet of the low pressure compressor.

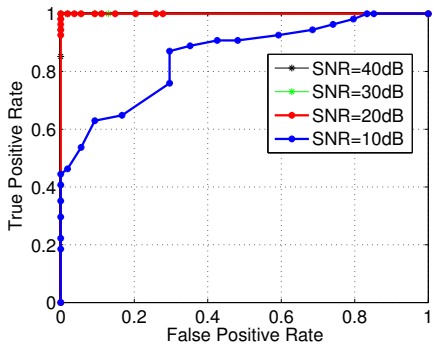
B. Results and Discussion

The training set comprises of the input signal profile of TRA and the output signal profile of $P24$ for all 54 operating conditions. The output data $P24$ from all these operating conditions are first normalized and then concatenated to form the complete output set. This data is then discretized using a maximum entropy partitioning [1]. The number of states in the PFSA is selected to be 15. Following the procedure outlined in section III-B, the augmented input space is constructed by discretizing the input and phase space. In this case, the output itself suffices as the phase space, i.e. $m^* = 1$. The input specific probabilistic state transition matrices are next constructed, which concludes the training of the PFSA.

In the validation part, the input and output data corresponding to a single fault level (for example, when the fan



(a) Fan Efficiency = 0.995



(b) Fan Efficiency = 0.970

Fig. 6. Receiver-Operator Characteristics of the Symbolic Identification Classifier applied to data contaminated by various noise levels

efficiency level is, say, 0.995) but for all different input and operating conditions, is fed into the algorithm. The pattern vector cluster corresponding to this fault condition is calculated according to the algorithm described in section III-C. The success or failure of the algorithm depends on the distinguishability of these patterns from the pattern cluster generated by the machine when the engine was running in its nominal health state, albeit at different operating conditions.

A support vector machine with linear kernel is used to classify the nominal from the off-nominal cases. The validation is done by choosing one of the data sets as test data and using the remaining data as the training data, and noting whether it could be classified correctly. This is repeated for all the data sets to yield a True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR) and False Negative Rate (FNR). Here "positive" denotes nominal condition and "negative" denotes off-nominality. Thus false positive implies a missed event, i.e. a faulty engine is classified as healthy, and a false negative implies a false alarm, i.e. a healthy engine is misclassified as faulty.

In order to estimate the robustness of the technique with noisy data, the data is contaminated with additive white gaussian noise. The SNR in dB is decreased from $100dB$ in the first run to $10dB$ in steps of $10dB$.

Figure 6 shows the result of the SVM classifier when applied to pattern vectors corresponding to different fan efficiencies. It may be noted (Fig.6a) that even a fan efficiency decrease of 0.5% can be correctly detected (TPR$=100\%$, FPR$=0\%$) for data contaminated with noise with $SNR = 40dB$. Even for $SNR = 30dB$, the ROC curve closely approaches the left hand top corner. There is a significant decrease in performance, for $SNR = 20dB$ and beyond.

The corresponding result for a relatively larger fault, (corresponding to a fan efficiency decrease of 3% is shown in Fig. 6b. As expected, the noise tolerance for this classification is much higher, since the fault signature is also less subtle. It can be seen that a correct classification can be performed even when $SNR = 20dB$.

V. Summary, Conclusions and Future work

In this paper, some of the critical and practical issues regarding the problem of health monitoring of multi-component human-engineered systems have been discussed, and a syntactic method has been proposed. The two primary features of this proposed concept are: (i) *Symbolic identification* and (ii) *Pseudo-learning technique*.

The reported work is a step toward building a real-time data-driven tool for estimation of parametric conditions in nonlinear dynamical systems. Further theoretical, computational, and experimental work is necessary before the SDF-based anomaly detection tool can be considered for incorporation into the instrumentation and control system of commercial-scale plants. For example, real flight record data, and environmental condition data need to be incorporated to investigate the effects of atmospheric temperature and pressure variations on the fault detection algorithm. At the same time, the following theoretical aspects are under investigation:

- Development of a multi-dimensional partitioning for a MIMO system, which should be computationally inexpensive.
- Estimation of a theoretical bound on the error incurred in this process of anomaly detection.

References

[1] Ray A. Symbolic Dynamic Analysis of Complex Systems for Anomaly Detection. Signal Processing. 2004;84(7):1115–1130.

[2] Rajagopalan V, Ray A. Symbolic Time Series Analysis via Wavelet-Based Partitioning. Signal Processing. 2006;86(11):3309–3320.

[3] Khatkhate A, Ray A, Keller E, Gupta S, Chin S. Symbolic Time Series Analysis for Anomaly Detection in Mechanical Systems. IEEE/ASME Trans on Mechatronics. 2006 August;11(4):439–447.

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

[5] Kokar MM. On Consistent Symbolic Representations of General Dynamic Systems. IEEE Transactions on Systems, Man and Cybernetics. 1995;25(8):1231–1242.

[6] Takens F. Detecting strange attractors in turbulence. vol. 898/1981. Springer Berlin / Heidelberg; 1981.

[7] Gautama T, Mandic DP, Hulle MMV. A differential entropy based method for determining the optimal embedding parameters of a signal. IEEE International Conference on Acoustics, Speech, and Signal Processing. 2003;6(6-10):VI – 29–32.

[8] Martins JF, Dente JA, Pires AJ, Mendes RV. Language Identification of Controlled Systems: Modeling, Control, and Anomaly Detection. IEEE Transactions On Systems, Man, And CyberneticsPart C: Applications And Reviews. 2007 May;.

[9] Narendra KS, Thathachar MAL. Learning Automata An Introduction. Prentice-Hall; 1989.

[10] Frederick DK, DeCastro Jonathan A, Litt JS. Users Guide for the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS); 2007. NASA/TM2007-215026.