Expert Systems with Applications 39 (2012) 10977–10989

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A knowledge-based system approach for sensor fault modeling, detection and mitigation

Jonny Carlos da Silva [a,*], Abhinav Saxena [b], Edward Balaban [a], Kai Goebel [a]

[a] NASA Ames Research Center, Intelligent Systems Division, Moffett Field, CA 94035, USA
[b] SGT Inc., NASA Ames Research Center, Intelligent Systems Division, Moffett Field, CA 94035, USA

ARTICLE INFO

Keywords:
Detection
Sensor failure
Expert system
Neural Network

ABSTRACT

Sensors are vital components for control and advanced health management techniques. However, sensors continue to be considered the weak link in many engineering applications since often they are less reliable than the system they are observing. This is in part due to the sensors' operating principles and their susceptibility to interference from the environment. Detecting and mitigating sensor failure modes are becoming increasingly important in more complex and safety-critical applications. This paper reports on different techniques for sensor fault detection, disambiguation, and mitigation. It presents an expert system that uses a combination of object-oriented modeling, rules, and semantic networks to deal with the most common sensor faults, such as bias, drift, scaling, and dropout, as well as system faults. The paper also describes a sensor correction module that is based on fault parameters extraction (for bias, drift, and scaling fault modes) as well as utilizing partial redundancy for dropout sensor fault modes). The knowledge-based system was derived from the results obtained in a previously deployed Neural Network (NN) application for fault detection and disambiguation. Results are illustrated on an electromechanical actuator application where the system faults are jam and spalling. In addition to the functions implemented in the previous work, system fault detection under sensor failure was also modeled. The paper includes a sensitivity analysis that compares the results previously obtained with the NN. It concludes with a discussion of similarities and differences between the two approaches and how the knowledge based system provides additional functionality compared to the NN implementation.

© 2012 Elsevier Ltd. All rights reserved.

1. Introduction

When sensors are performing suboptimally, the overarching functionality that uses sensor responses is potentially impaired because sensors play a vital role in ensuring optimal performance and safety in any complex engineering system. The most basic problem is how to distinguish abnormal sensor behavior from ordinary (and nominal) system behavior. This is sometimes not completely straightforward because system dynamics and external disturbances mask the sensor malfunction. Where diagnosis of the exact sensor fault is desirable, it may be further hampered by the large number of fault modes that are manifested with similar symptoms. Another challenge is how to distinguish between a sensor failure and a system failure when the two have similar fault signatures.

The following sections examine the sensor fault problem in more detail and describe potential solutions. Section 2 briefly presents the motivation for the work, based on reviews of some historical examples of system anomalies directly related to sensor failures in the aerospace domain, as well as some related works on sensor fault. Section 3 contains a brief description of the testbed and NN based approach that was used as a baseline in evaluating performance of the knowledge-based (KB) system. The evaluation was done on the basis of the most common faults in temperature and vibration sensors, as well as system faults in an electromechanical actuation system, as described in Balaban, Saxena, Bansal, Goebel, & Curran (2009). Section 4 provides the details of the KB system, built on the knowledge acquired during the NN effort. This knowledge base was developed as a combination of object-oriented modeling, semantic network, and rules. Section 5 includes a description of the validation work, and sensitivity analysis to compare the system results with the NN approach outcomes. Section 6 presents the two approaches implemented to mitigate sensor faults. The paper concludes with an evaluation of similarities and differences between the NN and KB approaches.

2. Motivation and related works

The impact of sensor failure can vary considerably with the application domain. It can range from a nuisance (e.g. when the

* Corresponding author. Permanent address: Mechanical Engineering Department, UFSC P.O. 476, Florianopolis, SC 88040-900, Brazil. Tel.: +55 48 3721 9264.

E-mail addresses: jonny@emc.ufsc.br (J.C. Silva), abhinav.saxena@nasa.gov (A. Saxena), edward.balaban@nasa.gov (E. Balaban), kai.goebel@nasa.gov (K. Goebel).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.03.026

setpoint of an air conditioner is not properly read) to equipment damage (e.g., when sensors on an assembly line are malfunctioning) or even loss of life.

2.1. Sensor failures in aerospace applications

Sensor failures in the aerospace domain often impact safety. The literature is full of examples where a sensor malfunction led to either loss of mission or loss of life. Attempts to deal with these issues through multiple redundancy often come with weight, cost, and complexity penalties and are not guaranteed to ameliorate the problem (as illustrated in the following examples):

- Mars Polar Lander (MPL) mission control lost contact with the MPL spacecraft during its final descent to the Mars surface. Subsequent attempts to reestablish contact were all unsuccessful. The MPL mission was declared lost. The failure investigation determined that the most likely cause of failure may have been the misreading by a micro-switch sensor of a landing leg deployment recoil shock as an indication of surface touchdown. This erroneous sensor reading was the signal upon which the spacecraft software relied to "assume" that surface touchdown had occurred and commanded the shutdown of the spacecraft landing retro-rockets. This is believed to have resulted in a high velocity crash of the MPL spacecraft onto the planet surface, with resulting unrecoverable damage (NASA, 2002).
- When Mars Exploration Rovers-B (Opportunity) landed on Mars on January 25, 2004, an anomalous nighttime current draw was detected. A stuck-on Instrument Deployment Device (IDD) warm-up heater was initially suspected. Unfortunately, there were only flight temperature sensors on two of the five IDD actuators and the suspected actuator with the stuck-on heater had a flight temperature sensor that failed during the rover thermal balance testing before flight. Because of the high risk for an *in situ* repair, the project opted to forego replacement of this temperature sensor (Tsuyuki et al., 2004).
- On another deep space mission, Pioneer Venus, the challenge of dealing with sensor failures was also present. It should be noted here that the Pioneer Venus probe mission is actually regarded as successful. The probes were launched towards the surface and were not designed to survive the landing (although one continued to send data for a while after impact). Nonetheless, there was unexplained behavior of the sensors that caused the loss of valuable data in the deep atmosphere Table 1 presents some anomalies found in the four Pioneer Venus probes.

Temperature sensors on all four spacecrafts showed a discontinuous drop in temperatures, while net flux radiometers showed a sudden decrease in readings concurrent with a sharp increase in atmospheric temperature. It is worth mentioning that out of 18 anomalies presented in the original table, six had direct reference to temperature sensors (NASA, 2007). While this list is by no means exhaustive, it illustrates the risks encountered when control actions are performed based on flawed sensor information. Unfortunately, despite advances in materials, communications, computer technologies, and other areas, sensors continue to be an area of concern in aerospace systems.

2.2. Related works

This section presents some of the recent applications of intelligent systems to sensor fault management, as well as the most relevant aspects of the test stand used as testbed to collect data for baseline and fault scenarios. The scenarios were, subsequently, used to develop a NN to model, detect, and disambiguate some sensor faults. The section also briefly describes the NN system itself.

Athanasopoulou and Chatziathanasiou (2009) describe a procedure for identifying sensor faults and reconstructing the erroneous measurements, in the context of a Thermal Power Plant (TPP) operation. Knowledge discovery, based on historical data, was successfully applied to deriving models that estimate the value of one variable based on other variables correlated to it. The validation part of the project was based on rules derived from the measuring equipment requirements, plant operation specifications, and personnel experience. Data mining (DM) algorithms were applied for deriving models that estimate the value of one variable based on others used as input parameters. The estimated values could be used instead of the ones recorded by a measuring instrument that is out of order. The research concluded that DM algorithms could be used successfully in reconstruction of signals from faulty sensors. The main innovation presented in the paper, comparing to the similar applications reported in the literature, relates to the fact that the reasoning algorithm was based on models derived from historical TPP operation data. This promises to allow the system to adapt, by extracting new models, to operational changes due to wear, maintenance, component replacement, or even addition of new equipment for similar intended operational usage.

Ding, Fennel, and Ding (2004) introduce an on-line sensor monitoring and fault diagnosis scheme for an electronic stability program (ESP) that consists of: an anti-lock break system (ABS), a traction control system and a yaw torque control. The basic idea for the scheme is in construction of analytical redundancy for sensors using system models. On this basis one can generate residual signal between expected and actual outputs. A fault detection and isolation (FDI) algorithm then evaluates the residual signals for abnormalities. The project applied a scheme to define residuals, which enhances the robustness of the detection system to model uncertainty and ensures a low false alarm rate, while also reducing fault sensitivity, thus only large magnitude faults can be detected. The scheme classifies driving situations as steady (straight on and steady cycle driving) or unsteady driving. Depending on the driving condition, different limits are defined for the following variables: vehicle speed, steering angle, change of steering angle, and lateral acceleration or yaw rate (depending on the sensor under consideration). The system applies different strategies for residual generation based on the driving situation. Fault detection threshold is set

Table 1
Some anomalies experienced by the Pioneer Venus Probes at and below 12.5 km altitude.

Anomaly	Probe			
	Large	North	Day	Night
Temperature sensors failed	x	x	x	x
Changes and spikes in pressure data	x	x	x	x
Failure of net flux radiometer fluxplate temperature sensors		x	x	x
Change in the indicated deployment status of the atmosphere structure temperature sensor and net flux radiometer booms		x	x	x
Erratic data from two thermocouples embedded in the heat shield		x	x	x
Erratic data from a thermistor measuring junction temperature of the heat–shield thermocouples		x	x	x

to be the maximum influence of the model uncertainties, and also depends on the driving situation. The project demonstrates implementation for an automotive system with three sensors to consider: yaw rate, lateral acceleration, and steering wheel angle. The paper presents simulations for straight-on and slalom manoeuvres. The fault detection scheme was successfully tested by a manufacturer, and is now produced and installed on different types of cars.

Figueroa et al. (2004) present a framework to model smart sensors in a rocket test facility. The framework proposes the integration of knowledge bases related to sensors, processes, actuators and other components. A key feature of the test facility is the evaluation of condition for all elements performed both autonomously and using feedback from other higher-order elements. The framework links distributed smart sensors into a coherent network using standardized interfaces and an expert system tool as the core.

3. Description of the testbed and NN approach

3.1. Testbed used for sensor fault diagnosis

The testbed described in Balaban et al. (2009) served as a starting point for this research. A ballscrew electromechanical actuator was used as the plant in the experiments performed on a test stand located at Moog Inc., where the test actuator was connected to a hydraulic load cylinder (acting as a load actuator) by a rotating horn. Control and data acquisition were performed by real-time software running on dSPACE platform.

Vibration was measured at four points on the test actuator, as shown in Fig. 1. All three axes of vibration were measured, with an additional measurement in the *Z*-direction using an accelerometer mounted directly on the nut of the ball screw. Temperature measurements were provided by a T-type thermocouple on the nut and a resistance temperature detector embedded in the stator of the motor. Load was sensed by a load cell. The position of the rod end of the test actuator was measured by a linear differential voltage transducer (LVDT). Current transducers were used on each motor phase to measure the phase currents. For data acquisition, the motor drive output an analog signal representing the torque producing current, as well as the motor velocity.

Sensor faults were injected *a posteriori*, as described in the next section. Permutations of the following conditions were used to run $2 \times 2 \times 2 = 8$ scenarios for each of the mechanical component fault cases:

- *Motion profile:* sinusoid or triangular wave.
- *Load type:* constant or spring.
- *Load level:* low (860 lbs spring force, 900 lbs constant force) or high (1725 lbs spring force, 1800 lbs constant force).

Since the above experiments produced short segments of data at fixed operating conditions, extended duration scenarios were

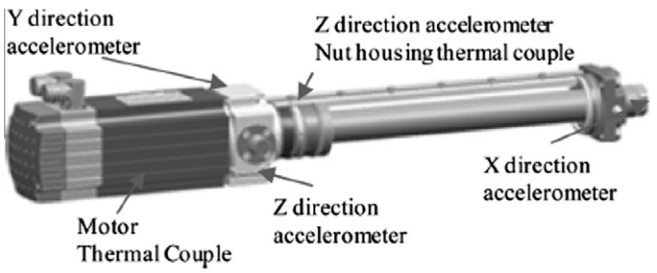
Y direction accelerometer
Z direction accelerometer
Nut housing thermal couple
X direction accelerometer
Z direction accelerometer
Motor Thermal Couple

Fig. 1. Location of sensors on electro-mechanical actuator (Balaban et al., 2009).

created by splicing data from different types of experiments. The new scenarios were designed to preserve the character of the collected data as much as possible, while extending the duration to 180 s each and varying load and position profiles. These scenarios contain two main segments – the initial part consists of nominal data, to represent a healthy system before the fault occurs, and a later part representing fault data (90 s per segment). Since the hardware limitations of the test stand required that the faults be seeded before the corresponding experiments began, nominal data was chosen from experiments conducted under the same conditions. The seeded faults included mechanical faults typically observed in EMA systems. Furthermore several sensor faults were simulated in the experimental data to generate scenarios with sensor fault data. The task of the diagnostic classifier was to distinguish between the sensor and system faults and, further, to determine the exact fault type. Specific sensor fault types were chosen based on the type of sensors used in the testbed and their common fault modes available from the literature. The sensor fault types used are listed below:

- *Bias: bias* was injected as an offset in the sensor output of the nut temperature sensor. It was specified as percentage of the average baseline temperature (80 °F), calculated over the set of nominal (no fault injected) scenarios. Gaussian noise was then introduced into the actual amount of bias added, with a signal-to-noise ratio (SNR) of 5.
- *Drift: drift* fault was also injected into the output of the nut temperature sensor. The fault was defined as a changing, mostly growing, offset and was specified by drift velocity (offset grown in a certain period of time). The effect of drift was further compounded with noise by adding several small segments of randomly varying drift slopes. The length of constant drift velocity segments was also randomized. Gaussian noise was introduced into the velocity value for this randomization, so for each segment the velocity may be somewhat different from its neighbors. The SNR for the later factor was set to 5.
- *Scaling: scaling* was simulated by amplifying or attenuating the signal by a scaling factor. The scaling factor was also added with Gaussian noise resulting in SNR of 5.
- *Loss of signal:* this fault represents a dead sensor and was simulated by replacing the sensor data from the point of fault injection with zeros.

Four types of experiments were designed. Dataset was partitioned into two parts based on the load levels, i.e. high load (~1700 lbs) and low load (~800 lbs) conditions. For the first experiment the NN was trained using only the high load data and fault detection and identification was performed on low load data that the NN never saw. The next experiment reversed the order by training on low load data and testing on high load cases. Another training set was created by combining data from both load levels. In the third experiment the NN was trained with 30% of the data set and tested on the other 70%. Finally, the NN was trained on 70% of the data and tested on the remaining 30%. The results from all these four experiments were documented and compared. Aggregated results were also used to perform a sensitivity analysis for various levels of sensor fault magnitudes. For detailed results please refer to Balaban et al. (2009). The following section presents some details of the NN implementation.

3.2. Description of the NN approach

In order to detect and isolate various fault types (both system and sensor), offline signal processing was carried out to identify signatures of each fault in the collected data. This step is called feature or condition indicator (CI) extraction and is one of the most important steps in building a successful (accurate and reliable) diagnostic system.

For a practical implementation, it is desirable that the features be not only computationally inexpensive, but also explainable in physical terms. Furthermore, they should: (a) be characterized by large between-class and small within-class variance; (b) be fairly insensitive to external variables like noise; and (c) uncorrelated with other features. Keeping these criteria in mind, a set of seven features was selected (Temperature Deviation – TD – and Drift Indicator – DI – on the ballscrew nut and motor housing thermocouples; Standard Deviation – SD – on the *X*, *Y*, and *Z* accelerometers) that were expected to detect an anomaly and distinguish between a healthy system, two actuator fault modes, and four sensor fault modes, as shown in Table 2. It should be noted that such representation of faults and features are common to several classifier approaches for diagnosis, as exemplified for instance by the dependency matrix in TEAMS (Kurtoglu, Johnson, Barszcz, Johnson, & Robinson, 2008).

All features and fault combinations, shown in Table 2, were used in the previous study using the NN as a diagnostic algorithm, with the following descriptions: TD – absolute deviation from the nominal temperature range. DI – a binary feature that assumes the value of one, if a finite rate of change of temperature is detected and zero otherwise, and SD – standard deviation of the signal within one sampling window. The combination of faults and features shown in Table 2 also provides a starting point for the knowledge-based approach for diagnosis, by providing fault signatures and trends to be encoded in the knowledge base. A general description of the previously developed Neural Network is provided below to give the reader an appreciation of that approach.

A multicategory classifier was implemented using a three-layer NN. The first layer consisted of nine nodes, with tan-sigmoid transfer functions, one for each feature in the input feature vector. The hidden layer had five nodes with logsigmoid transfer functions and the output layer had seven nodes with logsigmoid transfer functions—one for each of the seven classification categories. All input features were continuous, real-valued, and were standardized to have zero-mean and unit variance (Duda, Hart, & Stork, 2000). Binary targets were assigned such that of the seven output bits only the correct category bit would be set 1 and the rest would remain 0. Initial weights for the network were chosen based on standardized input ranges in order to ensure uniform learning (Duda et al., 2000). Networks were trained using the resilient backpropagation (RPROP) algorithm (Riedmiller & Braun, 1993).

The sensor-feature pairs were determined to uniquely cover a fixed number of component and system faults scenarios. This meant that if the NN is presented with similar faults in a different sensor than what was used for training, it will be challenging for it to identify a correct sensor-fault combination in its current topology. Adding more fault scenarios in this fashion would increase the complexity of the NN and therefore this approach has poor scalability due to additional offline analysis and efforts required. Furthermore, with this approach the NN would require training with the new scenario data which may not always be available when a system is deployed. A more generic approach to expand the NN capability would be to redesign the NN where the characteristics of various sensor fault types are learned irrespective of which sensors are used and augment it with the system fault scenarios of interest. The results from the NN approach are further discussed in the next section, along with the results from the KB diagnostic work.

4. Knowledge-based approach for fault detection

This section describes the knowledge-based diagnostic approach for sensor fault detection and disambiguation. Using a symbolic AI approach, an expert system was developed based on the knowledge generated during the development of NN-based diagnoser.

4.1. Knowledge-base description

This expert system was developed in CLIPS (Giarratano & Riley, 1994), using a combination of object-oriented modeling, semantic networks, and rules. This particular choice of developmental framework was guided by the following factors:

- CLIPS Object-Oriented Language (COOL) module allows to take full advantages of object-oriented modeling;
- The representation paradigm was chosen based on previous experiences in developing expert systems to different engineering domains, including hydraulic system design, cogeneration power plant design, and natural gas transportation modeling – to name a few (Matelli, Bazzo, & Silva, 2009, 2011; Silva & Back, 2000);
- The combination of object-oriented modeling, semantic network, and rules in an incremental approach allows modularity, expandability, and robustness in the knowledge base (Giarratano & Riley, 1994; Gonzalez & Dankel, 1993);
- The framework allows for a rapid prototyping by the knowledge engineer, which was of a benefit in this case, given previous experience and time limitations.

This section explains the main knowledge representation elements and the system verification. Fig. 2 presents the UML activity diagram corresponding to the system functions.

From Fig. 2, the main functions can be summarized as:

- Load experiment file, set of five files with load level indicator (described below).
- Update nominal values, based on the load level, since the baseline changes with the load.
- Process time-series data corresponding to each sensor.
- Extract features every half second within a 1 s long sliding window.
- Detect main failure modes per sensor: bias, drift, scaling, and dropout.

Table 2
Fault versus feature matrix expanded from Balaban et al. (2009).

Faults/features	TDnut	TDmot	SDx	SDy	SDz	DI-Tnut	DI-Tmot
Return channel ball jam	x	x	x	x	x		
Spall			x	x	x		
Nut thermocouple drift	x					x	
Nut thermocouple bias	x						
Motor thermocouple drift							x
Z Accel scaling					x		
X Accel complete failure			x				

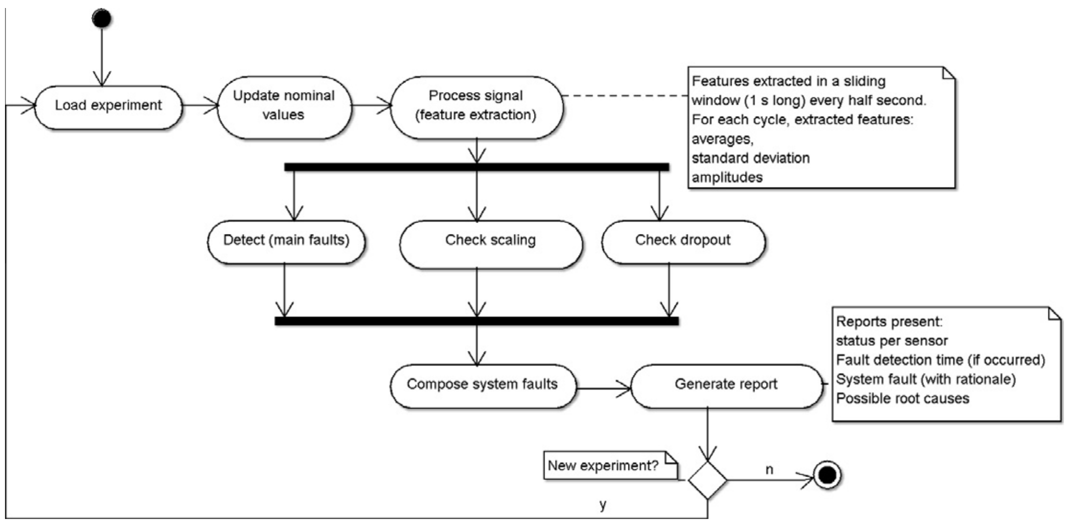

Fig. 2. Expert system UML activity diagram.

– Compose system faults, based on the status of different sensors, reason about system failure modes: jam or spall.
– Generate a report with fault detection time and rationale for selecting the particular type of fault.

In Fig. 2, each block represents a set of rules and methods that perform the aforementioned functions. As a key system function, feature extraction calculates and stores the following features for each cycle (i.e. half-second window): average standard deviation, mean value, and signal amplitudes. The features are extracted in a one second long window sliding every half a second. Therefore, for a 30 s experiment file used here as the basis to analyze sensor and system faults, feature extraction creates three vectors, with 60 values each. These vectors were modeled as CLIPS multislot attributes (Giarratano & Riley, 1994).

The main system function, i.e. sensor fault detection, is performed by three different processes, bias and drift fault detections are aggregated in the same method, whereas scaling and dropout detection are modeled separately, with the reasons for such choices explained below.

Due to the number of tests conducted to verify the system (discussed later in the paper), it was important to keep a proper record on each test, thus the Experiment class is created to identify inputs and generate reports. Such a feature also allows to execute multiple experiments in the same system session, and to replicate each experiment, if necessary. In order to perform the above functions, allowing the manipulation of the different types of sensors and a series of experiments, a considerable amount of knowledge was modeled in the system object hierarchy. Fig. 3 shows the class-object diagram composing the KB core.

As the main structure in object-oriented modeling, Fig. 3 depicts the KB system class diagram. As seen in Fig. 3, *Sensor* is the main system class (with only its key attributes presented here). It has two subclasses to model specific sensors, i.e. accelerometers and thermocouples. Although these sensors are quite different, due to the amount and type of information available in the current knowledge base it was decided for both classes to have the same attributes. The specific method to *detect* bias and drift was redefined, thus the need to create such sub-classes, once again, here object-oriented modeling played an important role by allowing easy implementation of polymorphism, i.e. allowing two types of sensors to perform the detection method in different manners. The reason for such distinction is that while for accelerometers the key parameter to fault detection is the standard deviation trend, for thermocouples such parameters are nominal value and amplitudes. Table 3 gives the rationale for each of the key attributes of the sensor class.

As the system is required to perform fault detection for two load levels, defined as high and low, there are typical-values attributes for both levels. This vector attribute specifies five key parameters for each sensor: nominal value, average standard deviation (Average-SD), standard deviation of standard deviation, maximum and minimum amplitudes. These values are previously calculated using the *baseline-values* method, by feeding the system with baseline scenarios for both load levels. As presented in Fig. 3, every experiment identifies the corresponding load level. Therefore, based on this load level the system sets the five nominal parameters, to be used as baseline for each experiment. It is worth mentioning that, although the approach implementation here is defined for only two load levels, due to the *baseline-values* method the system structure can be expanded to cover more levels.

With regard to fault isolation, the system provides two classes: Fault and Possible-Causes. The main attribute types and descriptions of the Fault class are specified from reasoning based on the status of each sensor within the context of the overall system. For example, in order to identify the jam fault it is necessary to have an off-nominal pattern in all three accelerometers and a change in both thermocouples, whereas the spall fault typically manifests itself via a faulty pattern in accelerometers *X*, *Y*, and *Z* only.

Another important aspect of the system is documentation of possible root causes. A comprehensive survey on different failure mechanisms for each type of sensor is presented in Balaban et al. (2009). However, in the previous approach identification of such failure mechanisms was not implemented. In order to provide this feature in the current effort, the *Possible-Causes* class is modeled. Although not shown in Fig. 3, the system does incorporate eight instances of the *Possible-Causes* class corresponding to the combination of four failure modes: bias, drift, scaling and dropout with two types of sensors: thermocouples and accelerometers. Therefore, depending on the result, the system presents a description of the most common root causes for that specific fault-sensor combination. A root cause isolation module was not implemented, since it was considered beyond the scope of this effort.

5. Diagnostics results

Different fault scenarios were considered for validation. Based on the experience in Balaban et al. (2009), it was expected that

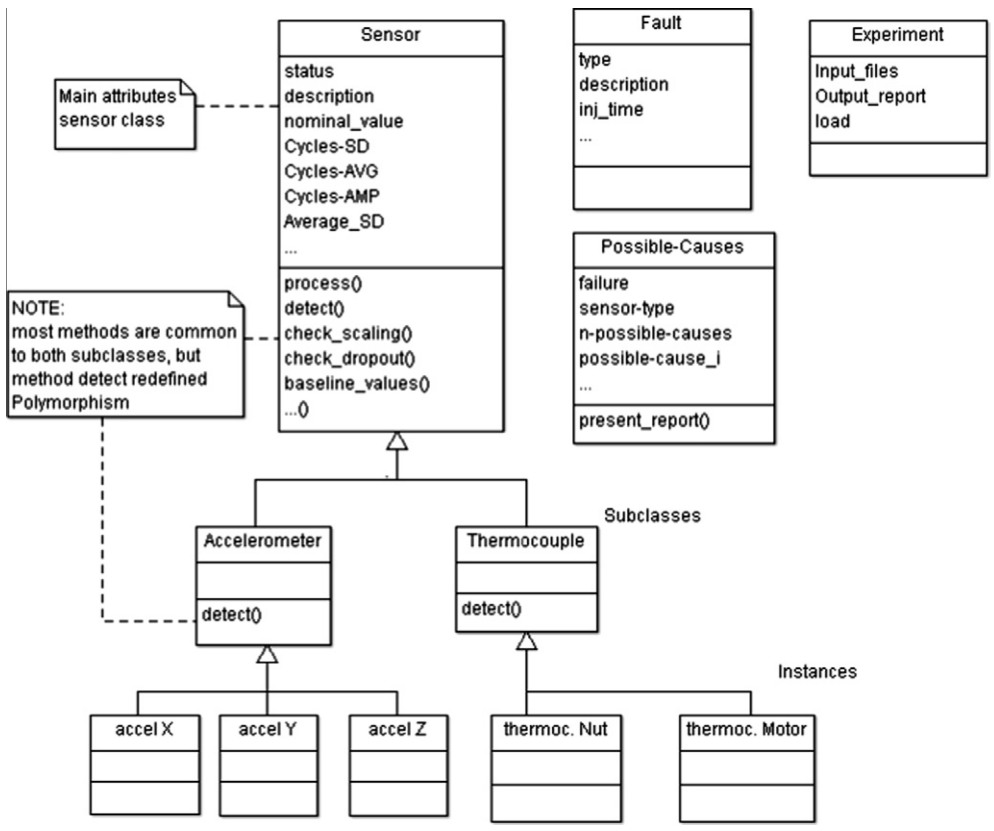

Fig. 3. Class (instance) diagram.

Table 3
Main sensor class attributes.

Attribute	Description
Status	Nominal by default. This is a key attribute for describing the fault, assuming different values according to the system inference, e.g. bias, drift, and scaling
Nominal value	Defines the nominal sensor value based on the typical values of attributes and the load level, explained below
Description	Multislot attribute to compose the sensor faulty scenario, in a user-friendly manner, including status value and fault detection time
Cycles-SD	Multislot to store the standard-deviation per cycle, used to detect bias and drift in accelerometers, to replicate the analysis seen in Table 2
Cycles-AVG	Multislot to store the mean values calculated per cycle
Cycles-AMP	Multislot to store the signal amplitude values calculated per cycle, used to detect the scaling fault
Average-SD	Mean standard-deviation obtained from the baseline-values method, applied for both load levels

the ability to detect each fault should vary considerably with values of fault parameters. Therefore it was necessary to perform sensitivity analysis to verify whether such behavior would be observed for the KB approach as well. Raw data that included baseline files and system fault scenarios (jam and spall) were available for different load levels. Fig. 4 depicts generation of fault scenarios used to produce the test cases with different fault magnitudes for the validation process.

The baseline files were used to set up nominal feature values, via *baseline-values* method, and to generate, via simulation, fault sensor scenarios for four failure modes. In order to perform the sensitivity analysis, the latter were set up using different sensors and fault parameters. The original system fault scenarios considered the occurrence of fault from the start, thus these new files were also merged with the corresponding baseline signatures to simulate the occurrence of faults at different times. Simulated fault injection was accomplished via AMESim$^{®}$ (LMS Imagine, 2008). Sensor faults were inserted as signal changes with constant parameters. The focus of this work lies in identifying fault detection and isolation capabilities and therefore the following four performance indicators, shown in Table 4, were recorded (note that True Negatives (TN) were not recorded).

In order to replicate, as close as possible, the test conditions from Balaban et al. (2009), different types of tests were performed. The results are grouped according to failure mode. First, drift fault test results are presented. Fault was injected in the actuator nut thermocouple with different drift slope values.

Fig. 5 presents the results of 22 drift sensitivity experiments based on 11 distinct slope values and 2 load levels. High load is depicted by the diamond symbol and low load is depicted with the circle symbol. The *x*-axis shows increasing slope values while the *y*-axis shows the categorization into FN, MC, and TP. There was no FP observed in any of the experiments. As the slope increases, the ease with which KB system is able to resolve the presence of drift improves from FN (at the smaller slope value) to TP at higher slope values. This behavior was consistent with the results

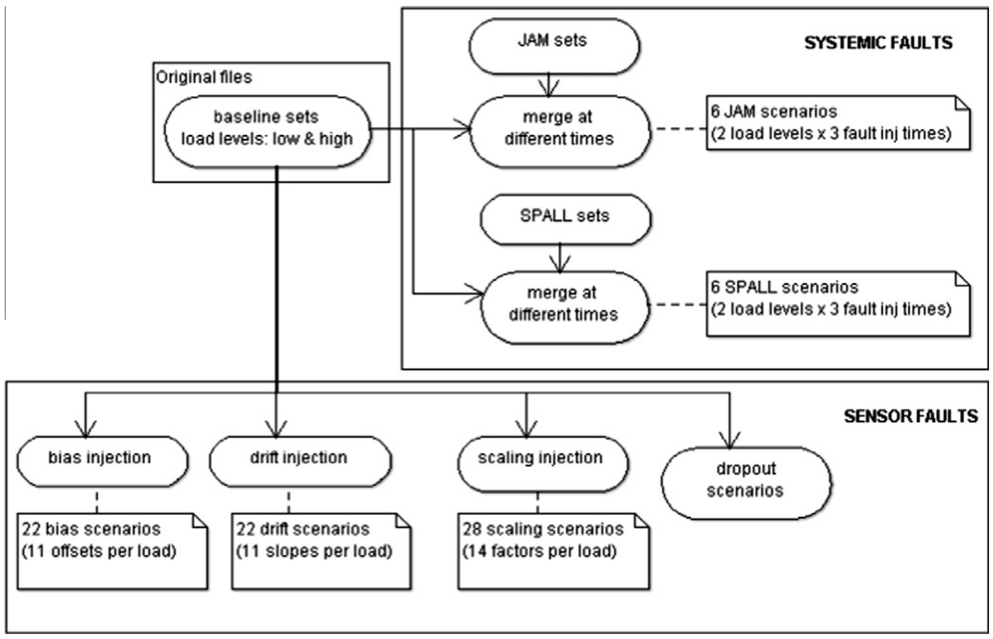
Original files
baseline sets load levels: low & high
SYSTEMIC FAULTS
JAM sets
merge at different times
6 JAM scenarios (2 load levels x 3 fault inj times)
SPALL sets
merge at different times
6 SPALL scenarios (2 load levels x 3 fault inj times)
SENSOR FAULTS
bias injection
drift injection
scaling injection
dropout scenarios
22 bias scenarios (11 offsets per load)
22 drift scenarios (11 slopes per load)
28 scaling scenarios (14 factors per load)

Fig. 4. Activity diagram to generate inputs for verification.

Table 4
Experiment results description.

Results	Description
True positive (TP)	Fault is properly detected and identified
False positive (FP)	Fault is incorrectly detected, i.e. there is no fault, but the system triggers a response. Such a result can cause unnecessary delays or sometimes even mission abort
False negative (FN)	There is a fault, but the system is unable to detect it. In some critical applications, FN can result in catastrophic failures
Misclassification (MC)	Fault is detected but incorrectly identified, for example the actual fault is bias, but the system classified it as drift

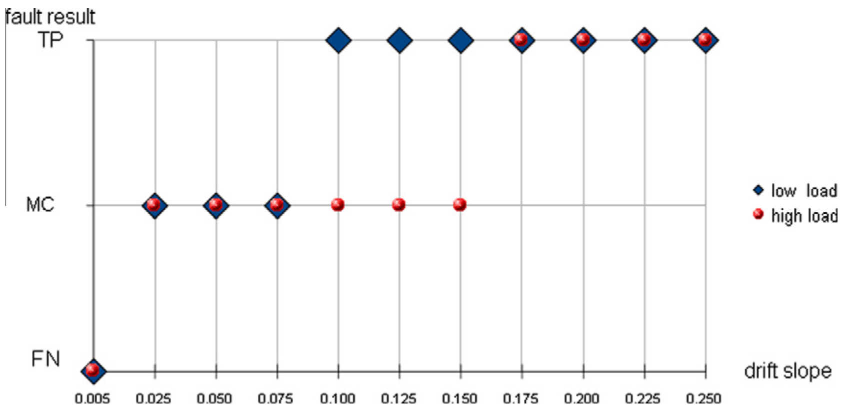

Fig. 5. Sensitivity analysis – drift fault.

obtained from the baseline NN approach. Where the system generated MC, the fault was detected, but misclassified as bias. As presented in Balaban et al. (2009), the NN diagnoser was slightly sensitive to small drifts. For small drifts it became difficult to disambiguate between drift fault and nominal behavior, resulting in a higher false negative rate. As Fig. 5 shows, the KB system also resulted in MC or FN for low drift velocities. Therefore, both systems have similar behavior in this respect. A closer examination of the raw signals explains this outcome (Fig. 6). For the sake of clarity, only four curves for different slopes are presented.

In Fig. 6 the baseline signal (solid red[1] line), with time in seconds, is a signal with relatively high noise values. The fault was injected at the start. Given a trajectory that is long enough, the fault would likely be detected eventually, irrespective of how small the slope value is (as long as it is increasing steadily). Small but increasing fault values are often interpreted as incipient faults

[1] For interpretation of color in Figs. 6 and 10, the reader is referred to the web version of this article.

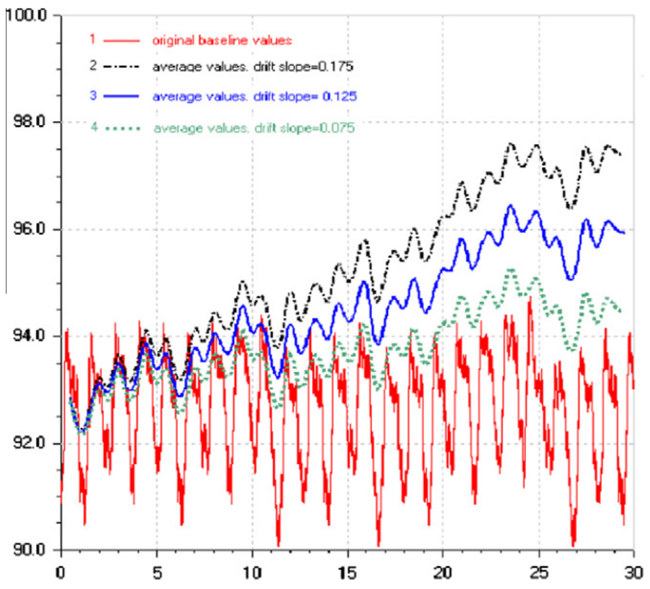

Fig. 6. Thermocouple signal values (°F) – high load level (baseline and average fault signals).

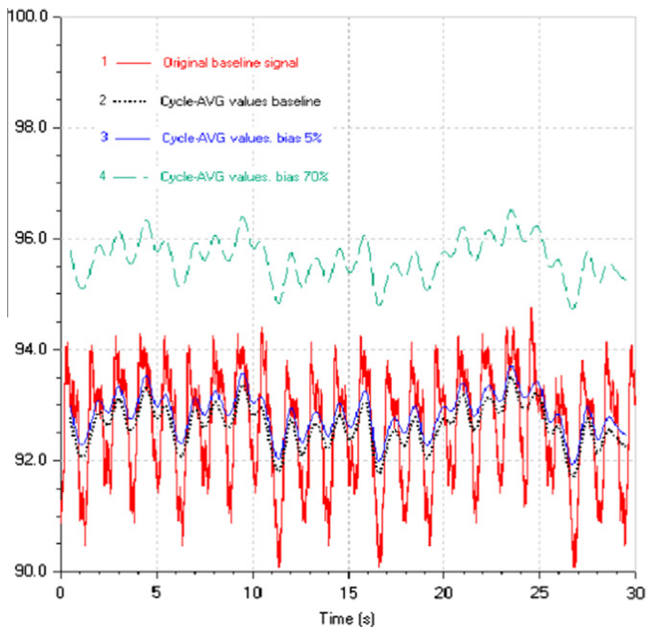

Fig. 8. Nut thermocouple baseline and average values (°F) bias fault – high load level.

(Frank, 1990) where faults are initially small and not easy to detect. The noise present in the system sometimes throws off the KB system output and results in false positives. In other cases a fault presence is identified (true positive) but the fault gets misclassified. Therefore, to improve the robustness of the system against noisy patterns and abnormal but random behaviors (e.g., signal spikes), the KB system waits and temporarily averages the estimates to remove the effects of noise. However, this results in longer detection times since now the system has to wait to observe if the feature remains out-of-range for a predefined time duration before a fault is declared. Consequently for drift faults the time to detect the drift also depends on the magnitude of drift slope, as discussed later.

The dependence on parameter values is also present in other failure modes. Fig. 7 shows the analysis for the bias fault (which was injected on the nut thermocouple). The same baseline signal as shown in Fig. 6 was used. The *x*-axis shows the increase in the bias factor (%), ranging from 5% to 100% peak-to-peak magnitude offset.

In this case, no MC was observed. At bias factors of 10% or more for the high load case and 20% or more for the low load case, the system properly identified the fault. Fig. 8 shows some corresponding signals. Here again, only a few curves corresponding to the tests, referred in Fig. 7, are presented.

As previously described in Table 3, the system calculates the signal mean values every half second. This feature is stored as cycle-AVG attribute. Fig. 8 shows the original baseline signal, the cycle-AVG values for baseline, and for two bias signals, considering 5% and 70%. As can be seen, although a 5% bias clearly shifts the signal from its original mean value, noise in the bias signal prevents proper detection.

For better comparison with the baseline results (see Table 2), scaling was injected in accelerometer *X*. Scaling factors of 0.1–2.0 were used to compare with the NN results. Since for scaling factors close to 1 the fault signal is very similar to the original signal more scaling values with smaller increments (0.05) were used to obtain a finer picture of the sensitivity analysis. Similar to the bias fault, no MC was observed, and the expert system was able to properly identify the fault for scaling factors higher than 1.2 (for high load) and 1.6 (low load), the system also identified the fault for scaling factors equal to or lower than 0.95. Fig. 9 shows the analysis for the scaling fault with variations relative to the baseline (scale factor of 1).

When the dropout fault was injected in accelerometer *Z*, the KB system properly detected it. System faults (jam and spall – as seen in Fig. 4), were combined to produce scenarios with different fault injection times. Both system faults (jam and spall) were detected for both load levels when injected at 0, 10 and 20 s (12 different experiments in total). As any mechanical fault, it is expected that

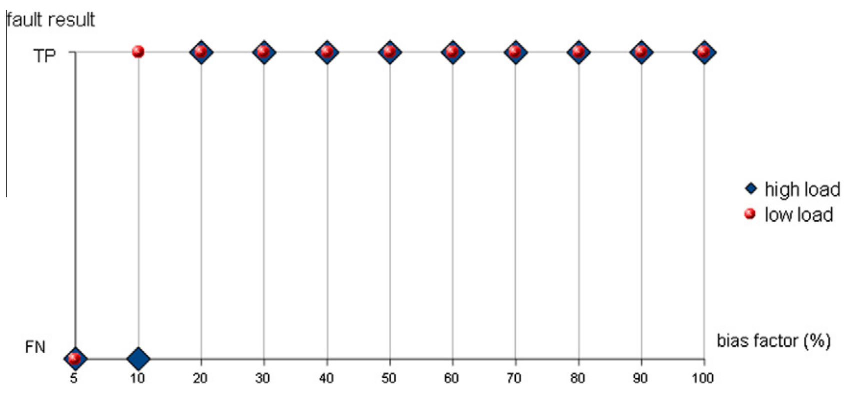

Fig. 7. Sensitivity analysis – bias fault.

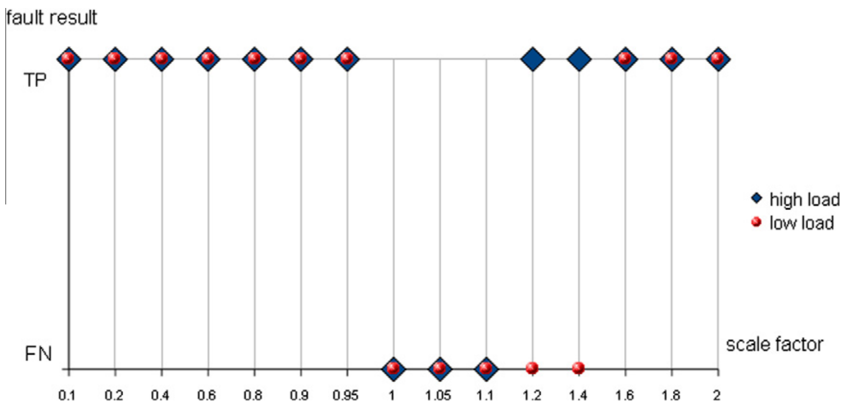

Fig. 9. Sensitivity analysis – scaling fault.

jam and spall, if present in a lesser degree, would not be detected as easily, however those signals were not available from the baseline experiments.

The sensitivity analysis results show that the overall system performance is similar to the NN approach, i.e. emulated sensor bias, drift and scaling resulted in similar detection characteristics. It should be noted that the knowledge representation scheme adopted in the KB models each sensor class with its own fault detection method. Although this requires a higher degree of tuning, the KB system can deal with a broader set of fault/sensor combinations (as shown in Table 5).

Table 5 highlights the two main differentiating points between the NN and KB approaches. The specific implementation of NN in Balaban et al. (2009) worked well given the problem description at that time (the four sensor and two system fault scenarios). However, if addition of another fault type is desired, the NN-based approach would require additional training data and a complete retraining of the NN (and possibly even its restructuring). The KB approach is more generic because it can accommodate more sensor faults of the types that are already modeled by just a new instantiation of the respective sensor class. Therefore, as depicted in Table 5, the KB is well equipped to detect and diagnose all four types of sensor faults on any of the thermocouples and scaling and dropouts on any of the accelerometer signals. This generalization provides additional flexibility for the diagnoser designer for practical implementation with relatively little additional effort. Furthermore, since the fault reasoning is performed for all fault types independently, the KB system is able to handle multiple simultaneous faults. These attributes can be seen as advantages over the previously implemented NN approach while the performance is comparable. It is, however, important to point out that with this paper we do not intend to establish a theoretical comparison between these two AI techniques, in terms of their applicability, but rather intend to point out differences in the scope of both systems in their current implementation. The development of the expert system has relied on the knowledge generated during that specific NN implementation, this does not mean that the NN based approach could not be implemented in other ways to incorporate these additional functionality if needed. It must be noted that the NN offers an advantage because the classification boundaries are learned directly from the data. For KB systems, this would require acquisition of additional background knowledge, which is a time consuming tuning process.

As a key issue in dealing with sensors as the "weak link" in a system, we evaluated detection of system fault under the possibility of presence of sensor faults. Two different approaches were considered to mitigate the effect of sensor faults while detecting system faults, described in the next two sections.

6. Fault mitigation

6.1. Heuristic approach – utilizing sensor redundancy

During the system design phase it is often desirable to understand how many sensors need to be fielded for proper fault detection and isolation. Although some approaches facilitate sensor design and placement (Kurtoglu et al., 2008), oftentimes, this decision is made in a somewhat ad-hoc fashion. Alternatively, it would be desirable to evaluate how robust the system is to sensor failure in order to assess its chances of survival in adverse conditions. Therefore, the prototype KB approach was expanded to reason on system faults given a reduced set of sensors. Of course, it should be expected that elimination of critical sensors comes with a penalty. This penalty could manifest itself in a lower detection accuracy and/or in longer detection times. It must be noted that the detection time depends on signal-to-noise ratio and confidence on the relevant sensor readings. Therefore, the detection time, a feature already calculated by the detection method but not used in the inference process, was added as a key evaluation parameter. For example, instead of relying on three accelerometers and two thermocouples to detect the jam fault, rules were added to reason based on only four of these sensors, provided that their fault

Table 5
Comparison in terms of detected fault/sensor scenarios.

Neural Network		Expert system	
Sensor	Fault modes	Sensor	Fault modes
Nut Thermocouple	Drift, bias	Nut and Motor Thermocouples	Drift, bias, scaling, dropout
Accelerometer *Z*	Scaling	*X*, *Y* and *Z* Accelerometers	Scaling, dropout
Accelerometer *X*	Dropout		
Total: 4 fault/sensor scenarios + 2 system faults The other sensors (Accel Y and Motor Thermocouple) are only used to identify jam and spall faults (see Table 2)		Total: 14 fault/sensor scenarios + 2 system faults + possible identification of multiple sensor fault scenarios	

Table 6
Comparing expert system output for a jam fault scenario with and without a sensor fault present.

Jam fault without faulty sensor	Jam fault with a faulty sensor
`List of faulty/change sensors:`	`List of faulty/change sensors:`
`(s1 status change at 21.5)`	`(s1 status change at 21.5)`
`(s2 status change at 22.5)`	`(s2 status change at 22.5)`
`(s3 status change at 22.5)`	`(s3 status change at 22.5)`
`(s4 type temperature_motor status change at 20.5)`	`(s4 type temperature_motor status change at 20.5)`
`(s5 type temperature_nut status change at 20.0)`	`(s5 type temperature_nut status dead failure detected at 10.0)`
`Fault type: Channel ball jam`	`Fault type: Channel ball jam`
`Description: (sensors s1 s2 s3 s4 and s5 status change within 2 s of each other)`	`Description: (sensors s1 s2 s3 s4 status change and s5 status dead failure detected at 10.0)`

detection times were close to each other, e.g. within a 2 second range. The system fault description was modified to consider such sensor faults. The next table compares the reports generated by the KB prototype for a system fault scenario with and without the presence of a faulty sensor.

In Table 6, the symbols s1–s5 correspond to the five sensors: accelerometers *X*, *Y* and *Z*, and thermocouples on the motor and on the nut, respectively. In the above scenarios, the system analyzed the same data files, with the exception that in one of them (the nut thermocouple signal) was modified to simulate a dropout failure at 10 s. All of the remaining sensors kept their original baseline signals for 20 s, before injecting jam fault.

As shown in Table 6, the system reports each sensor status separately, as it is calculated by the *detection* method according to the

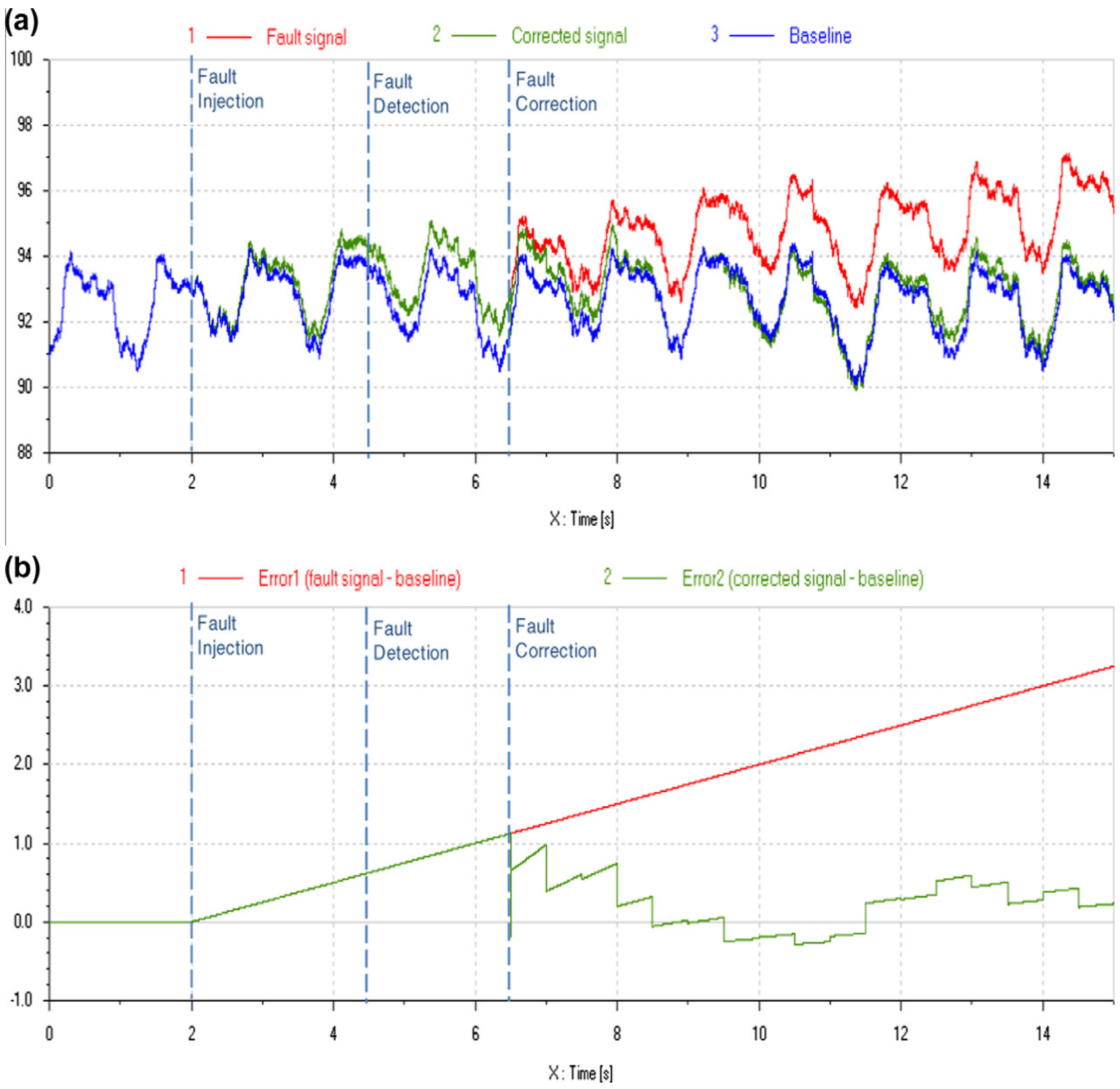

Fig. 10. Sensor correction results (a – signals, b – errors) (°F).

sensor class (i.e. thermocouple or accelerometer), plus a fault type description. In order to accommodate reasoning with faulty sensors, the knowledge base structure was expanded to include the corresponding fault detection times as parameters in the reasoning process. Furthermore, as characteristics of each sensor are different, the occurrence of a system fault, such as jam, triggers changes in sensors at similar, but not exactly the same time steps. Thus, a 2 s range, chosen based on data processing constraints, was used as an acceptable time window to distinguish between a sensor fault and a system fault. Notice here the fact that for identifying both system faults more than one sensor was needed. This implies that the signatures from all these sensors should change approximately at the same time as the fault actually appears in the system. Whereas in case of a sensor fault only the signature from that faulty sensor changes and the rest remain unchanged. In one rare case all sensors signatures can change simultaneously, which is when all sensors fail at the same time the likelihood of which is extremely low next to none. Capitalizing on this fact if a sensor fault is detected at any time its output may be ignored in making a decision about the system fault appearing at a later time. Therefore, by treating the faulty sensor output as a 'don't care' input the inference may be based on the remaining sensors only. This was possible in this case due to built in redundancy into the system, but may not be always possible. This function is addressed by a set of rules represented by the block "compose system fault", in Fig. 2, which analyses the status of each sensor being defined by the detection methods and identifies a system fault, if such fault exists.

6.2. Algorithmic approach – sensor correction

As discussed earlier, an important feature of a robust system is its ability to perform in the presence of failed sensors. This can be potentially accomplished through in-situ sensor value correction. In order to implement such a feature, new attributes were added in the KB's sensor class, in order to represent each fault parameter. For example, bias-level, slope, and scale-factor corresponding to bias, drift, and scaling respectively. The detection method was expanded to incorporate the extraction of these fault parameters.

Based on analysis shown earlier, the drift fault is the most difficult fault to detect because it has a temporal element. Therefore, in order to implement the slope calculation, instead of defining a single attribute to represent the calculated slope, the system models slope as a vector, whose calculation starts 2 s after the detection time. This lag was required in order to have a satisfactory slope calculation in the presence of noise, because signal features are extracted in half-second windows, for each measure of slope. Thus, in order to calculate the slope at time *t*, the system considers the average of slopes calculated based on the last four points (i.e.: t; $t - 0.5$; $t - 1.0$; $t - 1.5$). This definition was obtained empirically by calculating slope based on different number of points. As such, this approach may require further work to be applied to other data sets, since it may be influenced by signal noise amplitude and period. Averaging improves slope calculation even under considerable noise. Slope calculation improves with increasing time. At the beginning of the correction phase, the calculated slope is not very good, it turns out that its part on the correction method (i.e. error1 + slope.dt) is not that significant, therefore the correction is not impacted. Because, as shown in Fig. 10b, the total error is much smaller as drift starts. The correction method subtracts the calculated error from the signal. Fig. 10 presents the results of the sensor correction using the calculated slope.

Fig. 10a depicts a drift signal (slope = 0.250) inserted at 2 s (solid red line). The green line shows the corrected signal. The fault is detected at 4.5 s. Two marks on *X*-axis indicate fault injection and detection times. As mentioned above, the correction method starts 2 s after fault was detected. Until correction takes place, both fault and corrected signals are identical, therefore in Fig. 10a, only the green line is visible up to $t = 6.5$ s because it covers up the fault

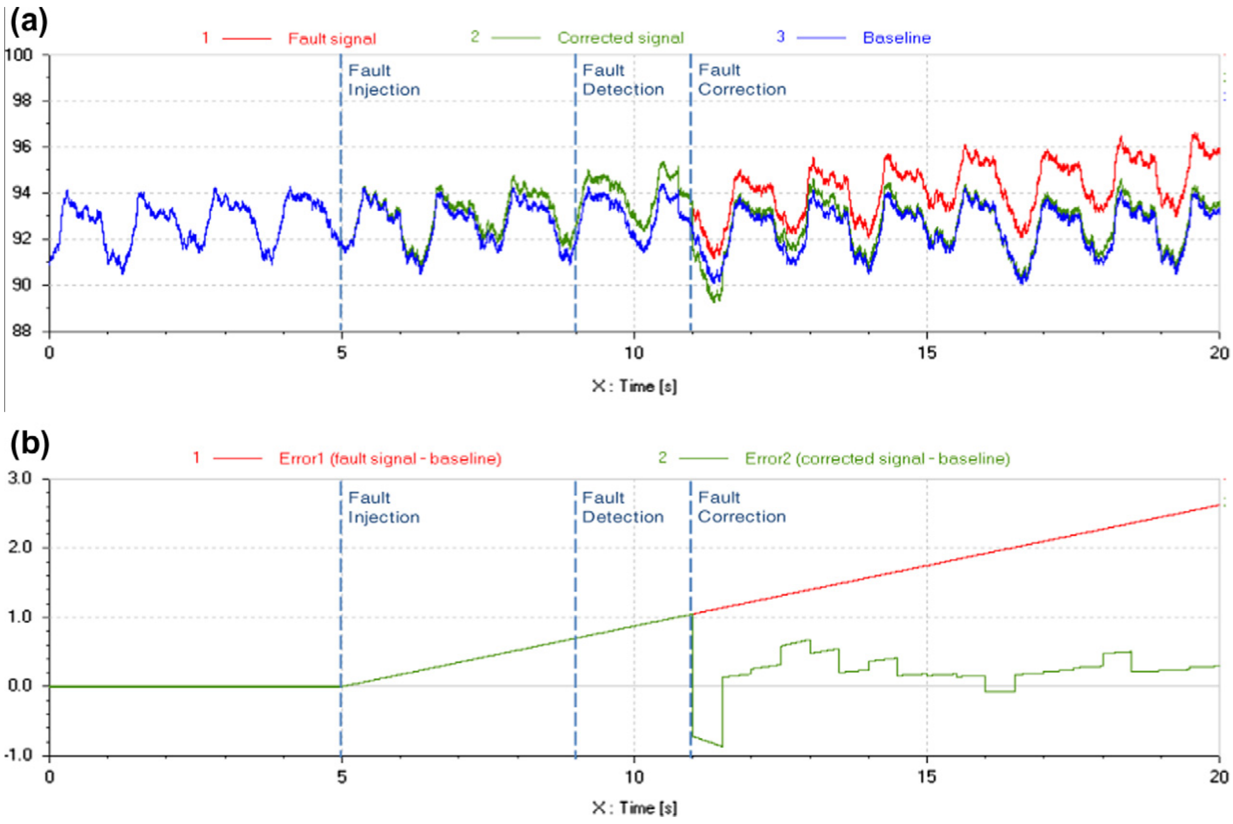

Fig. 11. Sensor correction results (a – signals, b – errors) (°F).

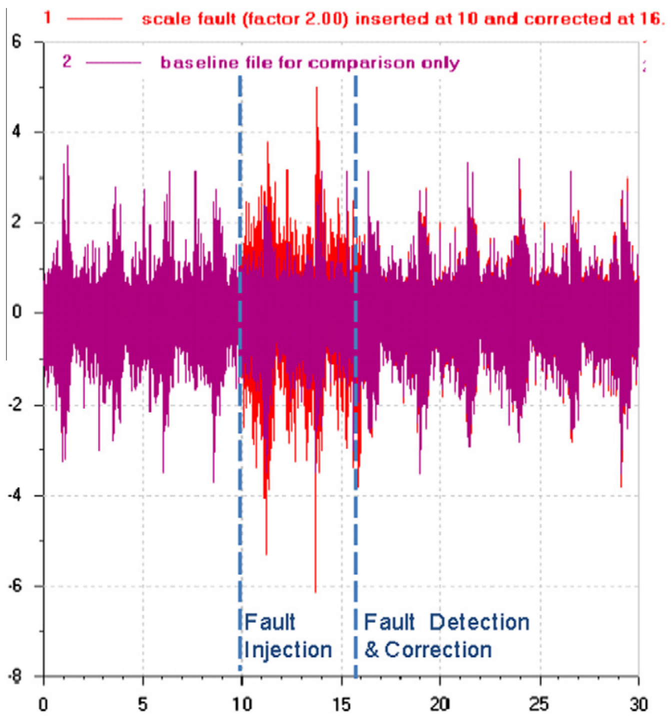

Fig. 12a. Sensor correction – scaling ($s = 2.0$).

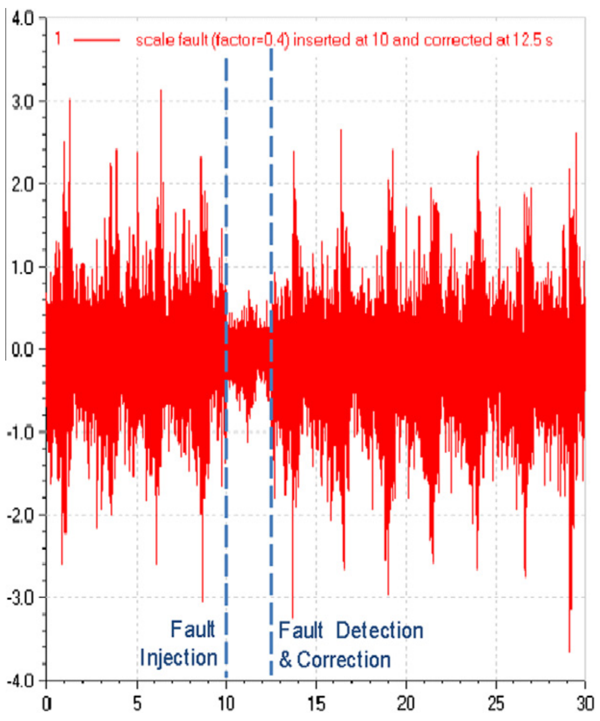

Fig. 12b. Sensor correction – scaling ($s = 0.4$).

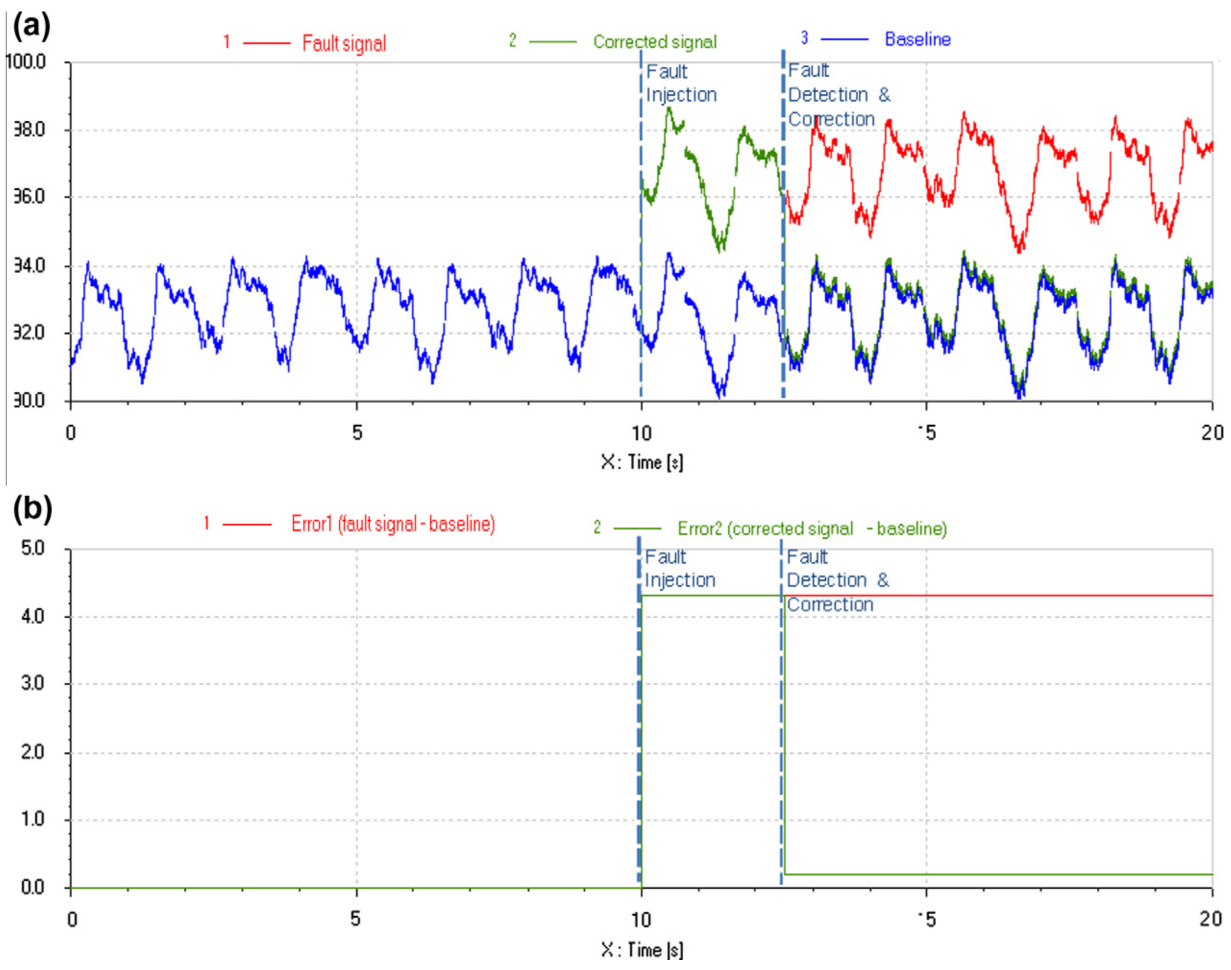

Fig. 13. Correction results for bias fault (a – signals, b – errors) (°F).

signal before this time. To evaluate the quality of signal correction, the original baseline signal (blue line) is also shown. In addition, Fig. 10b presents the errors corresponding to fault and corrected signals. Tests were conducted for different values of slope and fault injection times, including slope magnitudes above the detection. As reported in Section 4, the time to fault detection (and therefore to correction) depends on the fault magnitude. Fig. 11 shows a test with one of the smaller slope values.

Fig. 11a presents the result of sensor correction, with a drift inserted at 5 s (slope of 0.175). The KB system takes a longer time to detect it (in this case at 9 s, as marked on *X*-axis) because of the smaller slope value, but once the correction method starts (at 11 s), the error decreases.

Similar approaches were implemented for the other two fault modes, bias and scaling. Fig. 12 presents results of scaling fault experiments with two different magnitude factors ($s = 2.0$ and $s = 0.4$).

Figs. 12a and 12b show results of sensor fault correction for scaling faults with scaling factors of 2.0 and 0.4, respectively. For a better presentation, only Fig. 12a shows the baseline signal also. In both experiments, fault was inserted at 10 s. As can be seen, the system takes a shorter time to detect downscaling fault (in this example 2.5 s) compared to upscaling. The reason for that can be explained by the manner in which the detection method was implemented. To identify a scaling fault, the signal amplitude needs to be out of its nominal range, defined by the maximum and minimum amplitudes values, for at least four cycles. Due to this signal profile, an upscaling fault can manifest in an amplitude within the nominal range. More tests would be necessary to confirm similar behavior with an exhaustive set of different signal profiles. The correction method for bias fault follows a similar approach, with the results presented in Fig. 13.

Fig. 13a shows a fault signal (red solid line) with a bias inserted at 10 s. The green line represents the corrected signal. As shown in Fig. 13b, once the correction process takes place (at 12.5 s), the error decreases to acceptable values, i.e. an absolute error of 0.4 is smaller than the limit of detection found in the bias sensitivity analysis, seen in Fig. 7.

7. Conclusion

The paper presents the results of deploying a knowledge-based system for detection, identification, and disambiguation of various sensor and system faults in an electromechanical actuator system. Furthermore, the paper presents some approaches for mitigation of the most common sensor faults: bias, drift, scaling and dropout. Based on the sensitivity analysis, the KB system performance showed similar results compared to a NN-based inference system implemented previously. In addition, the system expanded on the previous work in three aspects. First, the KB system broadened the scope, both in terms of sensor/fault combinations and functionality (i.e. the system can handle more sensor-fault pairs without significant modifications). Secondly, the KB system demonstrated robustness of fault detection in the presence of sensor dropouts, since it was able to reason about system fault cases even when a key sensor signal was absent. Thirdly, the KB system was able to calculate fault parameters and correct sensor fault signals for the fault types discussed. More complex superimposed fault modes were not tested and remain to be investigated in the future. Finally, it is important to mention that it would be premature to draw conclusions that this approach is generic enough to handle all problems of this type, although the results obtained give hope that health management systems could be designed in ways that are more robust to sensor faults.

Acknowledgments

This project was developed under Grant 4095/10-3, CAPES Foundation, Brazil. The authors would also like to thank UFSC – Federal University of Santa Catarina (Brazil), and NASA Ames Research Center for their support in this project.

References

Athanasopoulou, C., & Chatziathanasiou, V. (2009). Intelligent system for identification and replacement of faulty sensor measurements in Thermal Power Plants (IPPAMAS: Part 1). *Expert Systems with Applications, 36*, 8750–8757.

Balaban, E., Saxena, A., Bansal, P., Goebel, K., & Curran, S. (2009). Modeling, detection, and disambiguation of sensor faults for aerospace applications. *IEEE Sensors Journal, 9*(12).

Ding, E. L., Fennel, H., & Ding, S. X. (2004). Model-based diagnosis of sensor faults for ESP systems. *Control Engineering Practice, 12*, 847–856.

Duda, R. O., Hart, P. E., & Stork, D. G. (2000). *Pattern classification* (second ed.). New York: Wiley, pp. 654.

Frank, P. (1990). Fault diagnosis in dynamic systems using analytical and knowledge based redundancy – A survey and some new results. *Automatica, 26*, 459–470.

Figueroa, F., Schmalzel, J., Morris, J., Solano, W., Mandayam, S., & Polikar, R. (2004). A framework for intelligent rocket test facilities with smart sensor elements. In *Sensors for industry conference, New Orleans, Louisiana, USA*.

Giarratano, J., & Riley, G. (1994). *Expert systems – Principles and programming* (second ed.). PWS Publishing Company.

Gonzalez, A. J., & Dankel, D. D. (1993). *The engineering of knowledge-based systems – Theory and practice*. Prentice-Hall, Inc.

Kurtoglu, T., Johnson, S. B., Barszcz, E., Johnson, J. R., & Robinson, P.I. (2008). Integrating system health management into the early design of aerospace systems using functional fault analysis. In *International conference on prognostics and health management*.

Matelli, J. A., Bazzo, E., & Silva, J. C. (2009). An expert system prototype for designing natural gas cogeneration plants. *Expert Systems with Applications, 36*, 8375–8384.

Matelli, J. A., Bazzo, E., & Silva, J. C. (2011). Development of a case-based reasoning prototype for cogeneration plant design. *Applied Energy, 88*, 3030–3041.

LMS Imagine (2008). *AMESim Reference Manual, Rev 8A*.

NASA (2002). *Probabilistic risk assessment guide for NASA managers and practitioners*. Available from: <http://www.hq.nasa.gov/office/codeq/doctree/praguide.pdf> Accessed January 2012.

NASA (2007). *Extreme Environments technologies for future space science missions – Final report*. Available from: <http:// vfm.jpl.nasa.gov/files/EE-Report_FINAL.pdf> Accessed January 2012.

Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of IEEE international conference on neural networks, San Francisco, CA* (pp. 586–591).

Silva, J. C., & Back, N. (2000). Shaping the process of fluid power system design applying an expert system. *Research in Engineering Design, 12*, 08–17.

Tsuyuki, G. T., Avila, A., Awaya, H. L., Krylo, R. J., Novak, K. S., & Phillips, C. J. (2004). Mars exploration rover: Thermal design is a system engineering activity. In *SAE – International conference on environmental systems*.