IJRCS

International Journal of Robotics and Control Systems
Vol. 3, No. 4, 2023, pp. 804-823
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

ASCEE

Wireless Sensor Networks Fault Detection and Identification

Rastko R. Selmic [a,1,*], Jake Scoggin [b], Stephen Oonk [c], Francisco Maldonado [c]

[a] Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada
[b] University of Connecticut, Connecticut, USA
[c] American GNC Corporation, Simi Valley, California, USA
[1] rastko.selmic@concordia.ca
[*] Corresponding Author

ARTICLE INFO

Article History
Received August 28, 2023
Revised September 20, 2023
Accepted October 17, 2023

Keywords
Fault detection;
Wireless sensor networks;
Sensor faults.

ABSTRACT

We have developed and experimentally tested a set of models for the detection and identification of sensor faults that commonly occur in wireless sensor networks. Considered faults include outlier, spike, variance, high-frequency noise, offset, gain, and drift faults. These faults affect the system operations and can endanger operators, final users, and the general public. The fault detection models are divided into two classes: data-centric models, which only analyze a single data stream, and system-centric models, which consider the overall system. For data-centric models, we use the magnitude, the gradient, and the variance of raw sensor data to model faults. For system-centric models, we introduce variogram-based techniques that allow faults to be detected by comparing readings from multiple sensors that measure related phenomena. For data-centric and system-centric sensor fault detection, we show how a few model parameters affect the sensitivity of wireless sensor network fault models. We present simulation and experimental results that illustrate the fault detection and identification models. The system is intended for health monitoring applications of the NASA Stennis Space Center (SSC) test stands and widely distributed support systems, including pressurized gas lines, propellant delivery systems, and water coolant lines. The testbed consists of Coremicro® reconfigurable embedded smart sensor nodes capable of wireless communication, a network-capable application processor, a wireless base station, the software that supports sensor and actuator health monitoring, a database server, and a smartphone running a health monitoring Android application.

1. Introduction

Wireless sensor networks (WSNs) can range from a small handful to hundreds or thousands of sensor nodes with various sensors monitoring diverse physical phenomena. WSNs are used to facilitate the automation of factories, monitor off-shore drilling equipment, quantify the health of oil refineries, and many other applications. In health monitoring applications, a WSN can measure anything from the efficiency of a turbine to the pressure in a pipe. Faulty sensor readings in these applications can lead to unnecessary shutdowns of plants or disruptions in monitored processes. While sensor technology advancements can reduce fault occurrences, they cannot be completely eliminated. Therefore, it is important to develop models and techniques that differentiate between legitimately

unhealthy conditions and faulty sensor readings in WSNs. This paper explores fault detection and identification techniques and develops mathematical models for such faults in WSNs.

Sensor and actuator faults in complex distributed electro-mechanical systems are well studied [1, 2], and in industrial wireless sensor networks [3]. In [1], a systematically characterized taxonomy of common sensor data faults that occur in deployed sensor networks and the detailed approaches commonly taken to model these faults are provided. These features include characteristics specific to the data, system, or environment pertaining to the system of interest. According to [1], the data features are usually statistical in nature. A confident diagnosis of any single fault may require more than one of these features to be modeled. The fault detection techniques most commonly used include mean and variance (determine expected behavior via regression models or correcting faulty sensor values), correlation (regression methods), gradient (rate of change), and spatial or temporal distance (determine if data is faulty). Table 1 shows common fault detection techniques that include statistical, nearest neighbor, clustering, and classification methods.

Table 1. Common fault detection techniques.

Ref	Method	Pros	Cons
[4, 5]	Statistical	Mathematically justified models	Fails if data do not fit assumed distribution
[6, 7]	Classificat.	Provides an exact set of faults	Computationally expensive, requires proper kernel choice
[9, 10]	Nearest neighbor	No assumptions on data distribution	Computationally expensive in large networks, dependent on input parameters
[11]	Clustering	No previous knowledge of data statistics needed, adapt to new data	Cluster width must be defined

Statistical methods create a model of healthy data and then classify any behavior that does not fit that model as a fault. For example, a statistical model of an outdoor temperature sensor may assume that temperature should be high during the day and low at night and register a fault anytime that trend is not observed. In this case, a fault could indicate a malfunction in the sensor network or an environmental irregularity, like a storm. Statistical methods have the advantage of being based on justifiable mathematical models but are not robust to phenomena that do not fit assumed distributions.

Fault classification methods take the opposite approach, and instead of creating mathematical models for healthy data, they use mathematical models for faulty data. Incoming data from the WSN is analyzed for behavior that corresponds to the faulty behavior previously modeled. The main advantage of classification methods is that they allow users to specify an exact set of faults to be considered, while the main disadvantage is that they usually cannot detect faults not previously observed.

Nearest neighbor methods forgo data models in favor of using a sensor-to-sensor comparison as a fault metric. These methods identify sets of sensors whose data should be similar (i.e., the spatially nearest sensors). Anytime these neighbors report significantly different readings from each other, a fault is registered. The main advantage of nearest-neighbor methods is that they allow faults to be detected without a mathematical model for healthy or faulty data. The main disadvantage is that they require appropriate neighbors to be defined, and mistakes in deciding which sensors are related can result in poor data comparisons.

Clustering-based methods compare clusters of nodes rather than sensors. During a learning period, sets of nodes are grouped into clusters, which are then compared to each other to establish which clusters are measuring related data. During fault detection, each cluster compares its data with its related clusters, and faults are registered whenever two related clusters do not measure similar data. Clustering methods are less computationally expensive than nearest-neighbor methods. The main disadvantage of clustering methods is the difficulty of appropriately defining clusters.

Table 2 contains results from the literature on fault detection and split them into categories useful for mathematical modeling. In [12], principal component analysis (PCA) is used during a learning period to model healthy data behavior with the first four eigenvectors. Using these eigenvectors as a baseline for healthy behavior, any activity that falls outside of the healthy model is considered an outlier. In [7], a support vector machine models healthy behavior and detects outliers in real time for a WSN. In [13], possible outliers are sorted and ranked periodically, speeding up the process of outlier discovery for a high-dimensional dataset.

Table 2. Fault detection in sensor networks research results.

Ref	Method	Pros	Cons
		Outliers	
[12]	PCA	Few false negatives, even with substantial drift and foreground variations	Local predictions not robust to faulty sensors
[7, 8]	SVM Classification	High accuracy, few false alarms	Computationally complex
[13]	Hash-Based Ranking Scheme	Detects top outliers with small dataset	Computationally complex
		Noise	
[14]	Nearest Neighbor	Works with small number of neighbor nodes	Assumes normally distributed noise
[15]	Nearest Neighbor, Clustering	Determines minimum neighborhood size	Requires fusion node
[16]	Nearest Neighbor, Statistical	Error probability decreases monotonically as the number of sensors increases	Computationally complex
		Stuck Sensors	
[17]	Clustering with random calls	Prevents faulty sensors from having disproportionately heavy loads	Less robust to localized events
[18]	Clustering	Relatively simple comparison process	Requires fusion node
[19]	Nearest Neighbor, Classification	Removes faulty and malicious nodes from network	Requires knowledge of fault probability in healthy sensors
		Calibration	
[20]	Nearest Neighbor	Allows a large number of nodes to be autonomously calibrated from a small base set	Requires dense sensor deployment
[23]	Nearest Neighbor	Blindly calibrates network, does not require dense sensor deployment	Assumes linear calibration functions

Calibration methods focus on identifying and accounting for offsets and gains in a WSN. In [20], it is assumed that the phenomena being measured should have similar behavior everywhere, and the characteristics of a small base set are used to model the characteristics of the entire network. In [23], it is assumed that the measured phenomena have some linear correlation between a sensor and its nearest neighbor. This information is used to calibrate the sensors in the WSN.

Previous fault detection techniques center mainly on simulation-based tests for algorithm development. In contrast, in this paper, we provide an overview of multiple fault models and test various fault detection models on an actual wireless network comprised of industrial-grade sensors. This allows us to demonstrate functional differences between models in real-world applications and to confirm that the fault models are capable of improving the health monitoring capabilities of an industrial system.

Other techniques for fault detection and identification include machine learning-based methods [24]. In [25], a digital equivalent of the sensor was developed using a generative adversarial network. The trained model works as a digital copy of the real sensor and watches for possible faults in the real sensors. Recent convolutional neural network-based sensor fault detection methods were developed in [26]. The convolutional neural network was used to detect and identify the sensor fault, while a bank of convolutional autoencoders was used to reconstruct the correct sensor data.

This paper studies fault detection and identification in WSNs. Faults included in this study are outlier, spike, variance, high-frequency noise, offset, gain, and drift faults. These faults affect the system operations and endanger operators, final users, and the general public. We have developed a set of fault detection models for WSNs and implemented them on a system consisting of multiple wireless sensor nodes, fault detection software, a server, and a client (Fig. 1). The system is intended for health monitoring applications of the NASA Stennis Space Center (SSC) test stands and widely distributed support systems, including pressurized gas lines, propellant delivery systems, and water coolant lines [27, 28, 29, 30]. The concepts presented here can also be used in other distributed industrial systems with a large number of networked electromechanical devices. The system within which our fault models were tested contains distributed software resources for providing ubiquitous information capability. The main system blocks are shown in Fig. 1 and include: (a) Server (developed in [31]); (b) Network Capable Application Processor (NCAP); (c) Communication Module; and (d) set of Transducer Interface Modules (TIMs) previously developed using the Coremicro Reconfigurable Embedded Smart Sensor Node (CRE-SSN) [32, 33]. The TIMs monitor sensors that measure physical phenomena such as temperature, pressure, and flow rate. They communicate measured data to the NCAP, which analyzes the data for faults. The NCAP records the data (as well as any detected faults) in a server where it can be accessed later by a user.

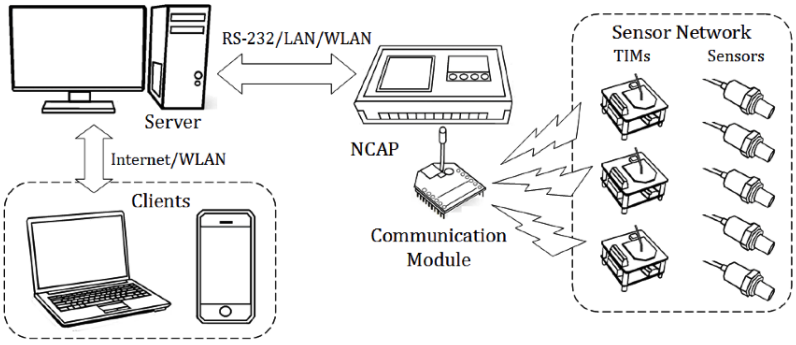

Fig. 1. Server-NCAP-TIM configuration.

2. Mathematical Models

Fig. 2 illustrates the sensor network model. Sensor readings for node j are given by

$$Z_j(k) = \left[z_j^1(k)\ z_j^2(k)\ \cdots\ z_j^i(k)\ \cdots\ z_j^N(k)\right]^T, \tag{1}$$

where individual sensors on a node are identified as $i = 1, 2, \ldots, N$, [34]. We assume that each node has an equal number of sensors, *N*, with output values at time instance k given by $z_j^i(k)$ (*i*-th sensor on *j*-th node at time *k*). A true value of a measured physical variable is $u_j^i(k)$ and a faulty measurement of the *i*-th sensor on the *j*-th node at time instance *k* is $\hat{z}_j^i(k)$.

There are two general categories for fault-detection techniques: data-centric and system-centric [1]. Data-centric techniques focus on a single data stream to identify faults, while system-centric techniques consider the whole system to detect faults. The two types of techniques need not be used exclusively. A computationally inexpensive data-centric technique may be used to identify abnormal behavior on a sensor stream, at which point a more expensive system-centric technique can be implemented that compares the sensor data to related neighboring nodes, verifying whether the abnormal data reflects an abnormal environment or a faulty sensor reading.

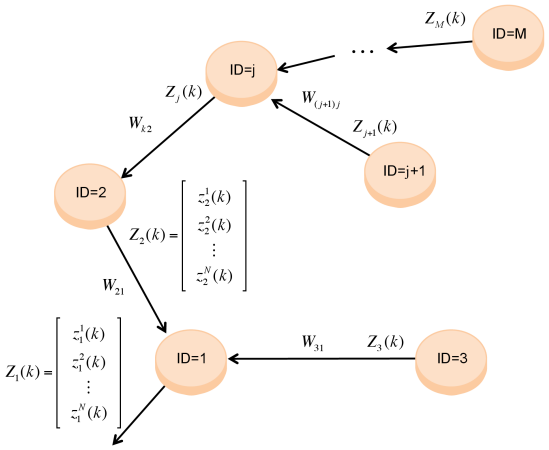

Fig. 2. Sensor network weighted graph.

2.1. Data-Centric Techniques

For each type of fault, there are defining characteristics, e.g., magnitude for outlier fault or variance for noise fault. In our approach, we identify a range for that characteristic under normal operating conditions, and we say that a fault occurs any time the characteristic of interest falls outside of the expected range.

The energy constraint in WSNs limits communication and computation complexities, while one of the leading hardware constraints is memory. For a large network, storing every data sample collected by a sensor is infeasible. Therefore, online fault detection algorithms with a sliding data window are preferable ("one-pass algorithms" [35]). A sliding window considers the last W_{sl} data samples from a sensor data stream. Data samples are stored for the duration of the sliding window and then erased. This method is robust to changes in the "normal" behavior of sensor data.

A less robust and less costly method of defining data norms is to use a learning period. A standard assumption is that sensor data are healthy during the learning period. By analyzing a data stream over the learning period, normal behavior can be defined. The advantage is that normal behavior has to be calculated only once. However, the method is not robust to data streams whose behavior changes with time. For example, the pressure in a pipe may be low during a learning period but rise when gas is pumped into it. Using a sliding window allows algorithms to update the expected behavior of a data stream as its normal behavior changes while still detecting true faults in the data stream.

2.1.1 Outlier Fault

We define an outlier as a single data sample whose value is *significantly* (defined by the user) outside of the range defined by previous data samples, see Fig. 3 [36]. To quantify this "*no-fault range*", we define $\bar{Z}_j^i$ as an upper bound of expected data (i.e., the highest value we expect from the *i*-th sensor on the *j*-th node) and $\underline{Z}_j^i$ as a lower bound of expected data.

The upper and lower bounds $\bar{Z}_j^i$ and $\underline{Z}_j^i$ are given by

$$\bar{Z}_j^i(k) = \max\{z_j^i(p)\} + c_{out}|\max\{z_j^i(p)\} - \min\{z_j^i(p)\}| \tag{2}$$

$$\underline{Z}_j^i(k) = \min\{z_j^i(p)\} - c_{out}|\max\{z_j^i(p)\} - \min\{z_j^i(p)\}|, \tag{3}$$

for $p = k_0, k_0 + 1, ..., k_0 + W_{ln} - 1$, where k_0 is the first data point in the learning period and W_{ln} is the number of data points in the learning period. We define c_{out} to be a positive constant that determines the outlier detection sensitivity. For instance, $c_{out} = 0.2$ allows the signal to vary

20% of the range above and below the data extrema observed during the learning period. Increasing c_{out} lowers outlier detection sensitivity by allowing some faulty readings to go undetected but while reducing the false alarm rate.

When using a sliding window, $\bar{Z}_j^i(k)$ and $\underline{Z}_j^i(k)$ are recalculated periodically. Bounds can be recalculated every time new data is received. For a sliding window of a length W_{sl}, every time new data is received, the sliding window is updated, and the bounds become functions of the current data sample. The running index p in this case is $p = k - W_{sl}, k - W_{sl} + 1, ..., k - 1$. The benefit of using such bounds is that if the range of expected data changes, the bounds will adapt to them. However, frequent updating of the signal bounds increases the computational cost and is a natural trade-off between the false alarm rate and computational cost.

We say that an outlier has occurred when the following condition is met:

$$z_j^i(k) > \bar{Z}_j^i(k) \text{ or } z_j^i(k) < \underline{Z}_j^i(k). \tag{4}$$

An example of an outlier is shown in Fig. 3.

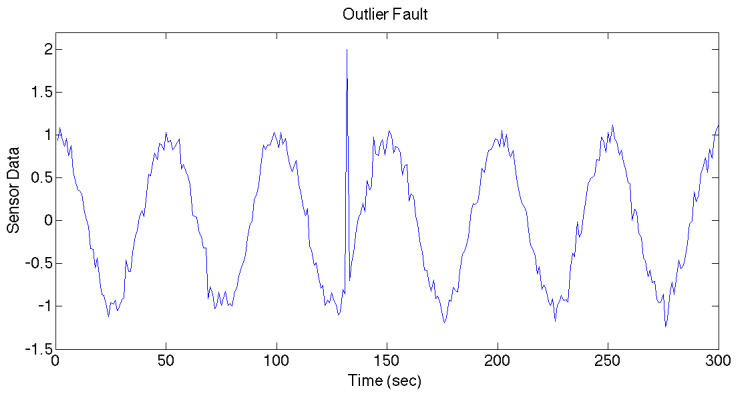

Fig. 3. Outlier occurring between 100 and 150 seconds.

2.1.2 Spike Fault

We use the term spike to refer to a small number (r_s) of data points that rise or fall more rapidly than the data during the healthy sensor behavior [37]. Fig. 4 shows a spike fault where data returns to normal behavior after the spike. To define a spike, we look at the gradient of data points and define the following: $\bar{R}_j^i$ being the upper bound on the gradient, which may be ascribed to healthy data on the *i*-th sensor at the *j*-th node; and r_s being the number of successive samples required to have an unhealthy gradient before a spike occurs.

As with outliers, the bound we use to determine healthy behavior is based on a learning period W_{ln} or a sliding window and is defined as

$$\bar{R}_j^i(k) = (1 + c_{spk})\text{max}\{|z_j^i(p) - z_j^i(p-1)|\} \tag{5}$$

for $p = k_0, k_0 + 1, \cdots, k_0 + W_{ln}$ in case of a fixed learning period and $p = k - r_s - W_{sl}, k - r_s - W_{sl} + 1, \cdots, k - r_s - 1$ in case of a sliding window; c_{spk} is a positive constant that determines the detection sensitivity.

When using a sliding window, the gradient bound is a function of sample time k, and it is assumed that data in the sliding window are healthy. The end of the sliding window should be set more than r_s samples away from the current data sample.

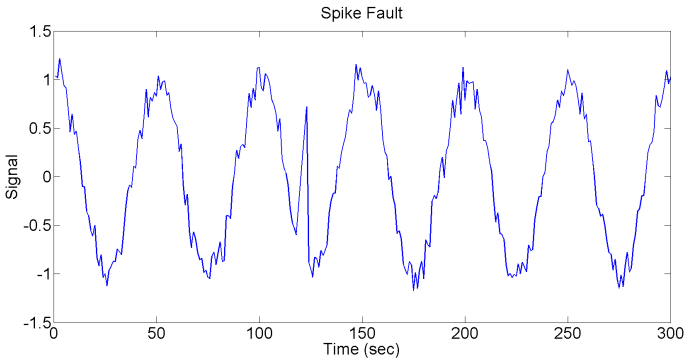

Fig. 4. Spike fault occurring between 100 and 150 seconds.

The number of successive samples, r_s, required for a spike fault is influenced by the phenomena being measured and the sampling rate. Some phenomena, such as light intensity, may be expected to produce signals with sharp gradients [1] while others are expected to change smoothly. For a slowly varying signal, a spike could indicate a fault in the sensor or that the sampling rate is too slow. Regardless, a spike in a data stream is a characteristic of interest and rapid detection is important. A small r_s will hasten the detection of spikes, which is crucial in time-constrained applications. Conditions for spike detection are given by:

$$\begin{aligned} z_j^i(k) - z_j^i(k-1) &> \bar{R}_j^i \\ z_j^i(k-1) - z_j^i(k-2) &> \bar{R}_j^i \\ &\vdots \\ z_j^i(k-(r_s-1)) - z_j^i(k-r_s) &> \bar{R}_j^i \end{aligned} \tag{6}$$

or

$$\begin{aligned} z_j^i(k) - z_j^i(k-1) &< -\bar{R}_j^i \\ z_j^i(k-1) - z_j^i(k-2) &< -\bar{R}_j^i \\ &\vdots \\ z_j^i(k-(r_s-1)) - z_j^i(k-r_s) &< -\bar{R}_j^i \end{aligned} \tag{7}$$

where (6) and (7) represent criteria for upward and downward spikes respectively. The necessity for the two sets of conditions, instead of just taking the absolute value of the gradient, is because a spike fault model requires r_s successive points to have either abnormally large positive or negative gradients. Moreover, this eliminates false positive spike identification when high-frequency noise is present, resulting in sensor readings having large gradients in random directions.

2.1.3 Variance Fault

The term variance fault is used to describe a set of data whose values tend to differ from the mean of that set by an abnormally large or small amount [38], where the threshold parameters are chosen by the user. Defining the length of the window over which variance is calculated as W_v, and the expected value of data over that window as $\overline{z_j^i(p)}$, the variance is then given by

$$V_j^i(k) = \frac{1}{W_v - 1} \sum_{p=k-W_v}^{k-1} \left(z_j^i(p)\right)^2 - \left(\overline{z_j^i(p)}\right)^2. \tag{8}$$

There are two types of variance faults: low and high variance faults. A low variance fault (also called a stuck-at fault) occurs when the variance of a data stream is abnormally low, i.e., when a signal is stuck at a certain value. A high variance fault occurs when the variance is abnormally high. The model parameters include $\underline{V}_j^i$ and $\overline{V}_j^i$, a lower and an upper bound on signal variance under healthy conditions. Using a learning period, the variance fault thresholds are given by:

$$\overline{V}_j^i = (1 + c_{var})\max\{V_j^i(p)\}, \tag{9}$$

$$\underline{V}_j^i = (1 - c_{var})\min\{V_j^i(p)\}, \tag{10}$$

where $p = k_0, k_0 + 1, \cdots, k_0 + W_{ln}$, the length of the learning period is W_{ln}, and c_{var} is a constant that determines the variance sensitivity. Using a sliding window, the index p belongs to a set of W_{sl} points as $p = k - W_{sl}, k - W_{sl} + 1, \cdots, k - 1$. We say that a variance fault has occurred if

$$V_j^i(k) > \overline{V}_j^i(k) \text{ or } V_j^i(k) < \underline{V}_j^i(k). \tag{11}$$

Increasing the learning or sliding window size provides a larger sample size for determining the variance. If the phenomena being measured are slow-varying throughout the learning period, it will result in a more accurate representation of the steady-state variance of the system. However, if the phenomena are fast-varying over the learning period, signal changes will be reflected in the healthy variance parameters. A small window size will emphasize the role of variance as imperfections in the system but will also increase false positives. Fig. 5 and Fig. 6 show low variance and high variance faults, respectively.

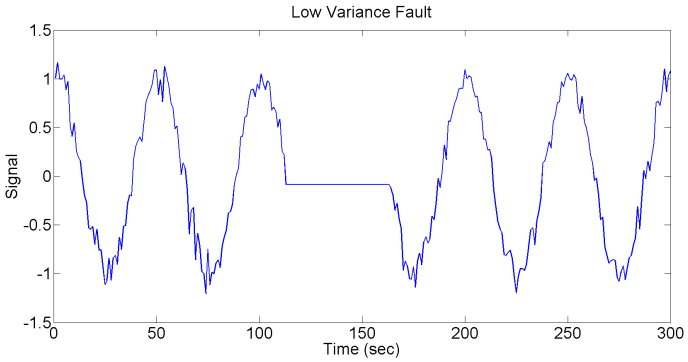

Fig. 5. Low variance fault occurring between 120 and 160 seconds.

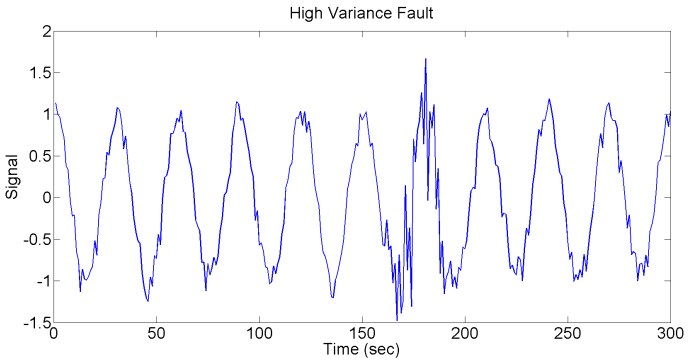

Fig. 6. High variance fault occurring between 150 and 200 seconds.

There are several possible causes of variance faults. The term ”noise” is often used to describe some high-variance faults in the case of signals with particular mathematical attributes. Several types

of noise exist, such as white noise, pink noise (equal power in bandwidths that are proportionally wide; see below), and violet noise (increases with frequency). Noise is usually classified by its behavior in the power spectrum, and we use the Fast Fourier Transform (FFT) to detect and isolate a noise fault.

2.1.4 High-Frequency Noise Fault

Noise classification and quantification are usually done in the frequency domain. We use the FFT to obtain the signal power spectrum, Fig. 7. Here, we describe a method for detecting undesired high-frequency signals in the power spectrum and how to distinguish the presence of unexpected high-frequency useful data and high-frequency noise [39].

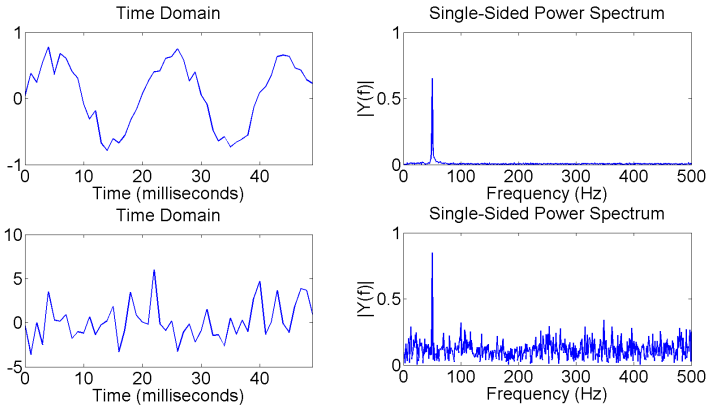

Fig. 7. Time domain and power spectrum for a sinusoid with a small amount of white noise (top) and a large amount of white noise (bottom).

Fig. 7 illustrates how the FFT can be used to detect noise in a sensor signal. Fig. 7 shows a power spectrum of a 50 Hz sinusoid with added white noise at different power levels. The uniform effect of white noise over the power spectrum can be observed. Conversely, if the high variance fault is caused by a new high-frequency signal added to the data stream, only a specific high-frequency band will be affected.

We define the window of data on which the FFT is performed as W_{fft}. The variables that define a high-frequency noise fault include $\overline{Y}_j^i$ – the maximum power level classified as noise in the power spectrum, and $\overline{\nu}_j^i$ – the highest frequency of a useful signal with a power spectrum contribution above the noise level $\overline{Y}_j^i$.

To define what constitutes a meaningful contribution to the power spectrum, we compute the power spectrum of a signal over a range of frequencies from $\nu = 0$ to $\nu = \nu_{max}$. We assume that the highest frequency (ν_{max}) is significantly higher than any frequency, contributing more than noise to the power spectrum. In particular, we assume the highest frequency component is at least W_{pow} higher than any frequency, which makes a meaningful contribution to the power spectrum. The maximum power level, which noise is expected to contribute, $\overline{Y}_j^i$, is then based on the power spectrum contributions of the last W_{pow} frequency components in the FFT:

$$\overline{Y}_j^i = \max\{Y_j^i(\nu)\} + c_{fft}|\max\{Y_j^i(\nu)\} - \min\{Y_j^i(\nu)\}|, \tag{12}$$

for $\nu \in (\nu_{max} - W_{pow}, \nu_{max})$ and where c_{fft} is a parameter that is used to tune the sensitivity of the high-frequency noise model. Any frequency component with a power spectrum contribution at or below $\overline{Y}_j^i$ is considered noise. Any frequency component that contributes a power level above $\overline{Y}_j^i$ is

said to have a meaningful contribution to the data stream. Examining the power spectrum of a signal over a healthy learning period allows one to define the highest contributing frequency component under healthy conditions as the highest frequency ν with a power spectrum contribution above $\overline{Y}_j^i$:

$$\overline{\nu}_j^i = \max\{\nu_j^i \mid Y_j^i(\nu) > \overline{Y}_j^i\}. \tag{13}$$

A high-frequency noise fault is then defined as the phenomenon wherein the power spectrum of a signal has a meaningful contribution at a frequency higher than the highest frequency expressed during the learning period:

$$Y_j^i(\nu) > \overline{Y}_j^i, \text{ for } \nu > \overline{\nu}_j^i \ . \tag{14}$$

2.2. System-Centric Faults

System-centric techniques use data from multiple sensors in the system to detect faults [21, 22]. The following techniques require at least one healthy sensor to be correlated to the faulty sensor. This, in turn, requires that the nodes in the WSN have multiple sensors that measure the same (or related) phenomena, i.e., that nodes are densely deployed to the point of oversampling the phenomena of interest [23].

To determine whether or not two sensors are sampling related data, we use variograms. Variograms allow us to quantify the correlation of phenomena at one point in the system with phenomena at another point in the system. A variogram $\gamma_{j,l}^i$ between sensors j and l is defined as:

$$\gamma_{j,l}^i = \frac{1}{2W_{vgm}} \sum_{p=k_0}^{W_{vgm}-k_0} \left(z_j^i(p) - z_l^i(p)\right)^2, \tag{15}$$

where W_{vgm} is the window of data where the variogram is being calculated. In the event of spatial correlation among sensors, the variogram is a function of radius r around sensor j:

$$\gamma_j^i(r) = \frac{1}{2W_{vgm}|\Omega_j(r)|} \sum_{q \epsilon \Omega_j(r)} \sum_{p=k_0}^{W_{vgm}-k_0} \left(z_j^i(p) - z_q^i(p)\right)^2$$

where r is the radius around sensor j, $\Omega_j(r)$ is the set of all neighbors of sensor j within radius r, and $|\Omega|$ is cardinality of the set Ω. A small variogram implies a high correlation among the sensors. We say that if the variogram between a sensor and its neighbors is smaller than some threshold Γ, then the sensor reading is related to its neighbors.

2.2.1 Offset Fault

An offset fault occurs when sensor data values are *offset* from the true phenomenon being measured by a constant amount:

$$\hat{z}_j^i(k) = f\left(u_j^i(k)\right) + \beta_0, \tag{16}$$

where the function f represents a nonlinear sensor model, $u_j^i(k)$ is a true value being measured on the i-th sensor on j-th node, and β_0 is the offset. Fig. 8 shows an example of an offset fault. To determine the offset value, one usually requires either a ground truth value of the sensor readings or a precise sensor model f, see [27]. We instead assume that the sensor network is distributed densely enough that any sensor has at least one related neighbor. An offset is defined as the steady-state condition of a constant difference between the readings of a sensor and the readings of its related neighbors.

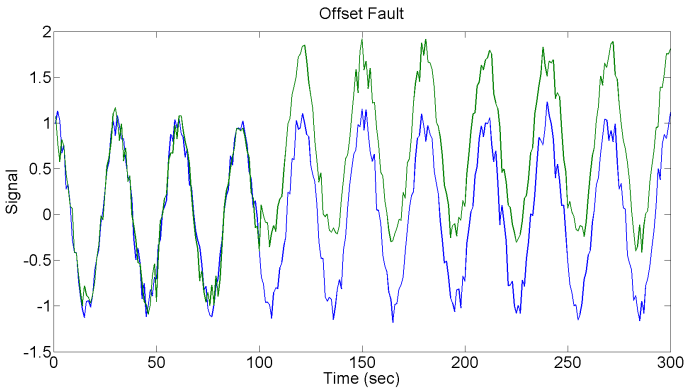

Fig. 8. The green signal has a constant offset compared to the blue signal.

To model an offset, we consider the difference between a sensor's readings and the average of the readings of its related neighbors:

$$\Delta_j^i(k) = z_j^i(k) - \frac{1}{|\Omega_j|}\sum_{q\epsilon\Omega_j} z_q^i(k). \tag{17}$$

A constant offset is modeled as a *slow-varying* difference between a sensor's measurements and the true value being measured. To distinguish between a constant offset and a time-changing offset (which may be considered a drift fault, see below), the variance, var $\left(\Delta_j^i(k)\right)$, over a window W_{oft} is observed. The conditions for an offset fault are that the difference between a sensor's readings and the readings of its related neighbors be above the threshold value and that the variance of the difference is sufficiently small, i.e., $\Delta_j^i(k) \geq c_{oft}$ and var $\left(\Delta_j^i(k)\right) \leq \beta_{oft}$, where c_{oft} and β_{oft} are model parameters.

2.2.2 Gain Fault

A gain fault occurs when the sensor data for a certain measured phenomenon differ from the healthy sensor data by a constant ratio (see Fig. 9) [40]. In the event of a gain fault, we expect the gain of a sensor's readings compared to the average of its neighbor's readings, $\eta_{j,gn}^i$, to be significantly lower or higher than 1. This ratio is defined as

$$\eta_{j,gn}^i(k) = \frac{z_j^i(k)}{\frac{1}{|\Omega_j(k)|}\sum_{q\epsilon\Omega_j} z_q^i(k)}. \tag{18}$$

To quantify the variation of the sensor's gain, we consider the var $\left(\eta_{j,gn}^i(k)\right)$ over a window W_{gn}. A gain fault has occurred when $|\eta_{j,gn}^i(k) - 1| \geq c_{gn}$ and var $\left(\eta_{j,gn}^i(k)\right) \leq \beta_{gn}$, where c_{gn} and β_{gn} are small, adjustable parameters.

2.2.3 Drift Fault

A drift fault occurs when sensor readings drift away from the real values by an amount that increases with time, Fig. 10 [41]. Given $\Delta_j^i(k)$, a drift fault model represents a steady increase in time, i.e., an offset increment $\lambda_j^i(k, L) = \Delta_j^i(k) - \Delta_j^i(k - L)$ is a function of time increment L:

$$\lambda_j^i(k, L) = \lambda_j^i(L) = c_{dft}. \tag{19}$$

If c_{dft} varies with time, then (19) models a drift that is increasing according to the integral of c_{dft}.

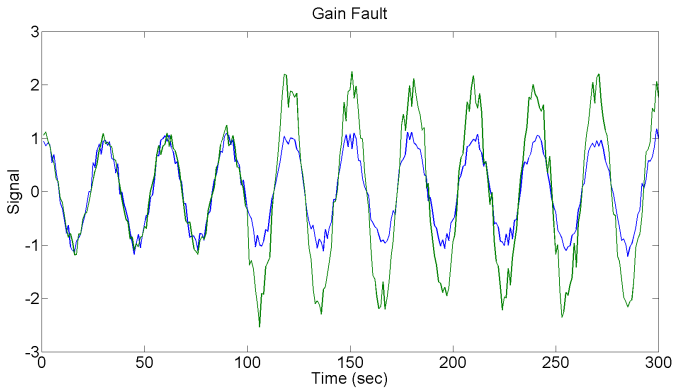

Fig. 9. The green signal has a different gain than the blue signal.

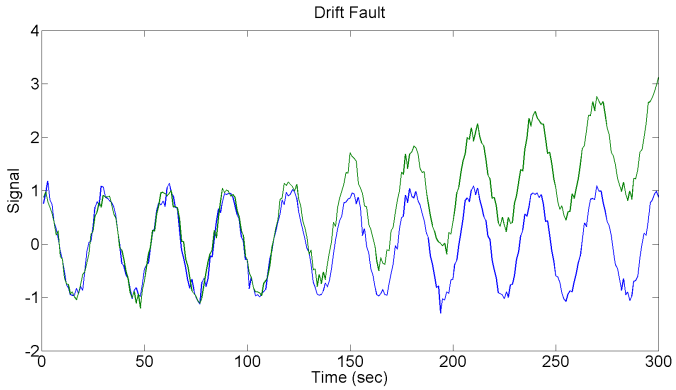

Fig. 10. The offset between the signals is steadily increasing, resulting in a drift fault.

3. Experimental Results

We gathered data from a wireless sensor network composed of industrial-grade sensors to test our models. Fig. 11 shows the experimental testbed that was used for sensor fault detection, fault information processing, and data storage. The testbed consists of Coremicro® Reconfigurable Embedded Smart Sensor Nodes (CRE-SSNs) [33] capable of ZigBee wireless communication, an NCAP with a ZigBee wireless base station, NCAP software that supports sensor and actuator health monitoring, a database server, and a smartphone running a health monitoring Android application. Each node can read and wirelessly transmit data from up to four sensors simultaneously. The sensors measure their environment and report these measurements to the wireless sensor nodes. The nodes wirelessly transmit the sensor data to the ZigBee wireless base station, which reports its readings to the NCAP, where software processes and records the data.

As an example of healthy sensor behavior, 10 sets of 6400 data samples (each approximately 1 minute long) were recorded. The experiment was carried out under two assumptions: (1) the phenomenon being measured (temperature) was slow-varying throughout all datasets, and (2) the health of the system was constant throughout all datasets. The obtained sensor data is shown in Fig. 12.

The characteristics of interest, other than the raw sensor data, are the gradient, variance, offset, gain, and drift of each sensor. Note that Data Set 1 has a spike just before 20 seconds and this spike in raw data results in a spike for all of the characteristic plots simultaneously. A single sensor fault may result in multiple fault types being flagged.

3.1. Data-Centric Parameters

Each data-centric fault model has two main parameters: a threshold parameter and a window size. Higher values for the threshold constants c_{out}, c_{spk}, and c_{var}, hereafter referred to collectively as c_{xxx}, are expected to correspond to lower numbers of faults being flagged for a given dataset, regardless of whether a learning period or a sliding window is used. The data in Fig. 12 was analyzed with various values of c_{xxx}, and results for two different data sets are shown in Fig. 13 and Fig. 14. A window size of 100 samples was used for both the learning period and the sliding window experiments.

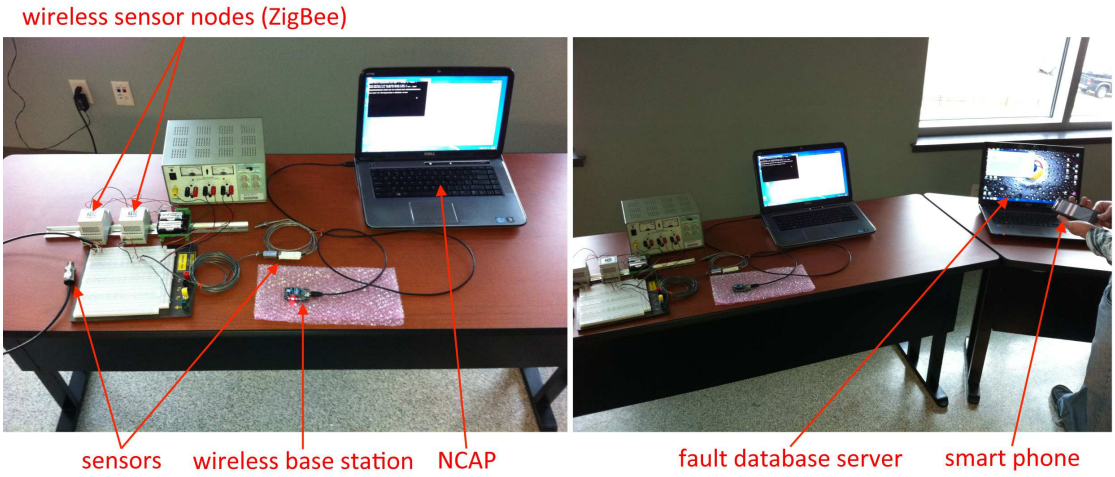

Fig. 11. The experimental testbed used to gather data is seen above.

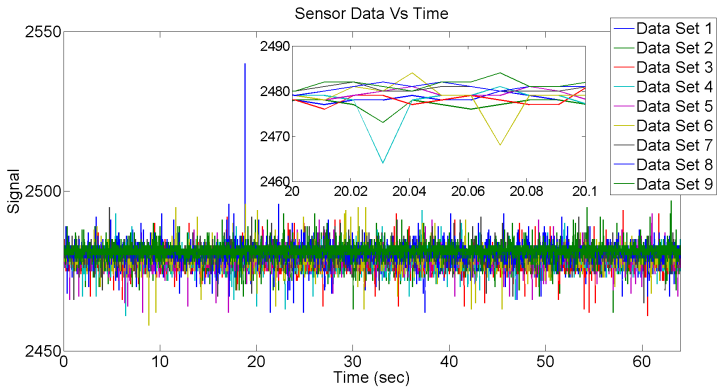

Fig. 12. Datasets used to experiment with model parameters. The insert shows a close-up of the data from 20 to 20.1 seconds.

A constant value of $c_{xxx} = 0.2$ was used in both the learning period and the sliding window experiments. The largest window size examined has 210 points, and data streams were analyzed for faults from the 211th point onwards. Fig. 15 shows the number of faults detected versus the window size when using a learning period to define normal behavior. Fig. 16 shows the number of faults detected versus the window size when using a sliding window to define normal behavior.

Note that a large window size results in more variance faults detected. This shows that the variance of these data streams over the latter part of the learning period was not indicative of the typical variance for the sensors. When a sliding window is used, larger window sizes result in fewer variance faults and fewer outlier and spike faults, as was expected.

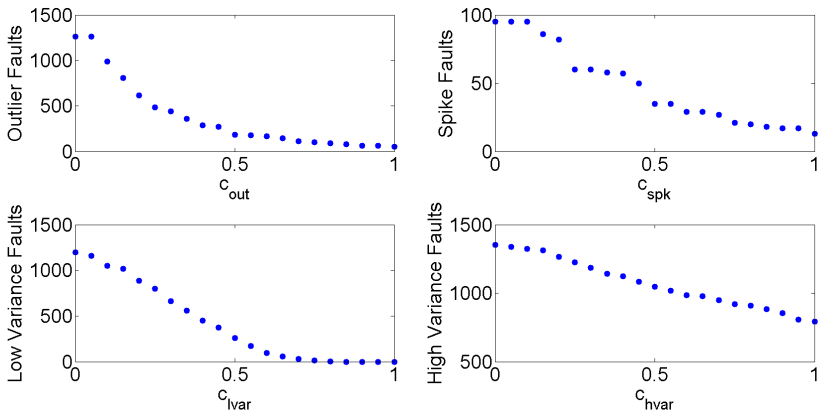

Fig. 13. Number of data-centric faults detected for various values of c_{xxx} with $W_{ln} = 100$: (i) Top left: outliers; (ii) Top right: spikes; (iii) Bottom left: low variance; and (iv) Bottom right: high variance.

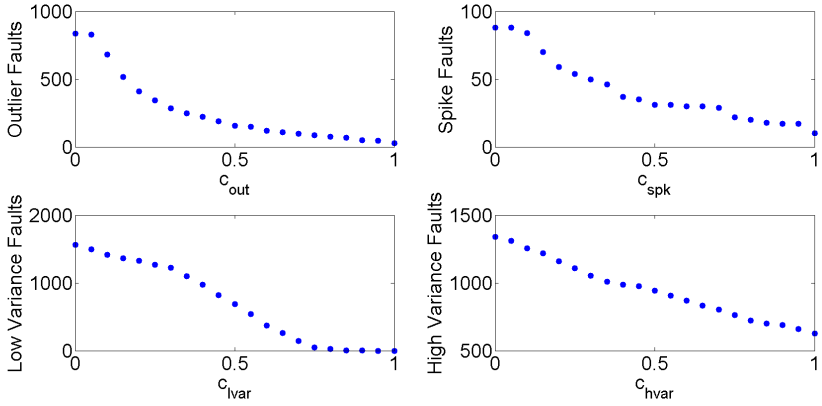

Fig. 14. Number of data-centric faults detected for various values of c_{xxx} with $W_{sl} = 100$: (i) Top left: outliers; (ii) Top right: spikes; (iii) Bottom left: low variance; and (iv) Bottom right: high variance.

3.2. High-Frequency Noise Parameters

High-frequency noise is modeled similarly to the other data-centric faults; however, the frequency domain analysis imposes certain constraints. Namely, using the Fast Fourier Transform (FFT) places constraints on the window size used (power of two). While a dataset may be "padded" with zeros to bring the total number of points up to the nearest power of two, this padding skews the noise levels of the power spectrum.

The high-frequency noise fault model has three main parameters: the threshold parameter c_{fft}, the window of data used W_{fft}, and the window of frequencies assumed to be purely attributed to noise at the upper end of the power spectrum W_{pow}. As with the other threshold parameters, increasing the value of c_{fft} is expected to decrease the number of high-frequency noise faults detected for a given dataset. Using a larger W_{fft} is expected to yield a better approximation of a data stream's behavior and is also expected to correspond with a lower number of faults detected. Finally, using a larger W_{pow} assumes that a larger range of frequencies can be attributed to noise during the learning period. Because the datasets under consideration are approximating a slow-varying phenomenon (assuming the temperature was constant during data collection), this results in a better approximation of the typical noise level for a dataset. Therefore, we expect larger values of W_{pow} to correspond with fewer high-frequency noise faults detected in a given dataset. Fig. 17, Fig. 18, and Fig. 19 show the effects of varying c_{fft}, W_{fft}, and W_{pow}, respectively. To cover a wide range of values for W_{fft} (4 - 1024

points), the value of W_{pow} was set dynamically to one-fourth of W_{fft} rather than statically set to a constant value.

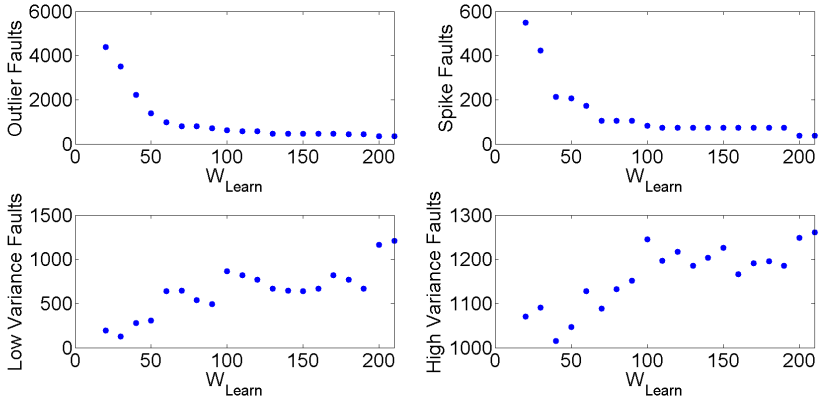

Fig. 15. Number of data-centric faults detected for various values of W_{ln} with $c_{xxx} = 0.2$: (i) Top left: outliers; (ii) Top right: spikes; (iii) Bottom left: low variance; (iv) Bottom right: high variance.

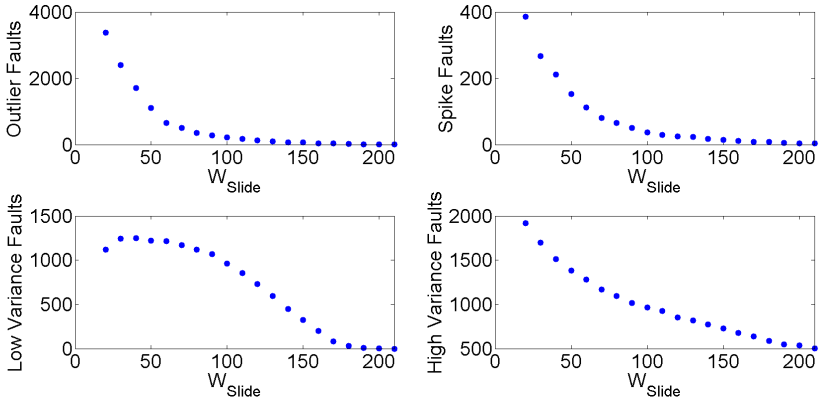

Fig. 16. Number of data-centric faults detected for various values of W_{sl} with $c_{xxx} = 0.2$: (i) Top left: outliers; (ii) Top right: spikes; (iii) Bottom left: low variance; and (iv) Bottom right: high variance.

3.3. System-Centric Paramaters

The variogram is used in system-centric techniques where the parameter Γ controls the size of a set of correlated neighboring sensors. Increasing Γ is expected to correspond with a higher number of related neighbors for a sensor. Each of the system-centric models has two main parameters: a threshold parameter representing the magnitude of the fault, (c_{oft}, c_{gn}, and c_{dft}) and a parameter representing the consistency of the fault over time (β_{oft}, β_{gn}, and β_{dft}). Hereafter, we collectively refer to the threshold parameters for system-centric models as c_{yyy} and the consistency parameters for system-centric models as β_{yyy}.

Larger values of β_{yyy} are expected to correspond with fewer faults being detected and represent looser constraints on the long-term behavior of the fault. For instance, a high β_{dft} means that as long as a data stream is consistently rising while its related neighbors are not, a drift fault will be flagged. A lower β_{dft} represents stricter conditions that the offset between the faulty sensor and its related neighbors be linearly increasing with time. Similarly, the magnitude of β_{gn} determines how constant the gain of a sensor must be compared to its related neighbors, with a lower β_{gn} imposing the stricter constraints of a constant gain. We examine the effect of both varying the threshold and consistency parameters in Fig. 20 and Fig. 21, respectively.

Varying the threshold parameter for each fault has the expected effect. Fig. 20 shows the number of detected faults when threshold parameters are changed. Fig. 21 shows the number of detected faults versus the consistency parameter. Note that for offset and drift faults, a higher consistency parameter results in more faults being detected, as expected. For gain faults, however, the consistency parameter has a different effect. No gain faults are detected when the consistency parameter is set to zero, but for any value higher than zero, the same number of gain faults are detected. This behavior was found regardless of the values of the threshold parameter c_{gn}. Such step behavior suggests that for gain faults, the threshold parameter has a much greater effect on the sensitivity than the consistency parameter.

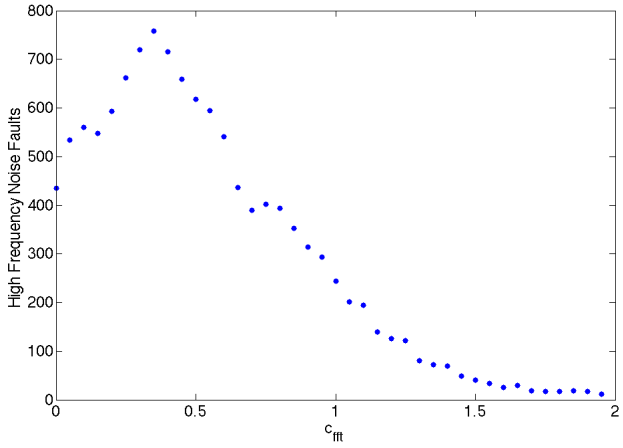

Fig. 17. Number of high-frequency noise faults detected for various values of c_{fft} with $W_{fft} = 256$ and $W_{pow} = 32$.

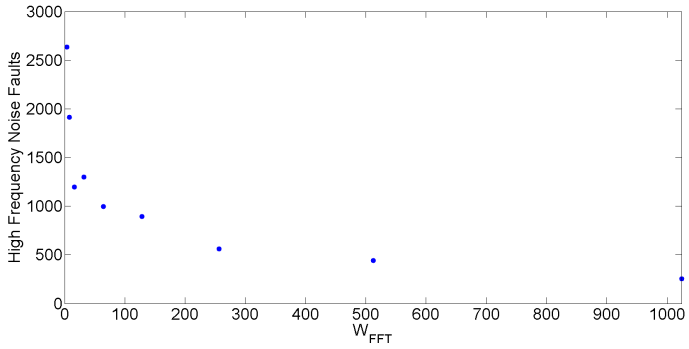

Fig. 18. Number of high-frequency noise faults detected for various values of W_{fft} with $c_{fft} = 0.2$ and $W_{pow} = W_{fft}/4$.

While these models have shown satisfactory performance with our testbed, please note that there is always a trade-off between sensitivity and false positives in fault detection. There is no universal value of the fault detection parameters presented here that would work for any application. Setting proper values of those parameters requires the operator's experience and intuition related to the specific system. Some parameters, such as upper and lower data limits for outlier faults, can be adjusted online. However, the window size and other sensitivity parameters are based on experience. Future research will study intelligent learning techniques where algorithms will set all fault detection parameters.

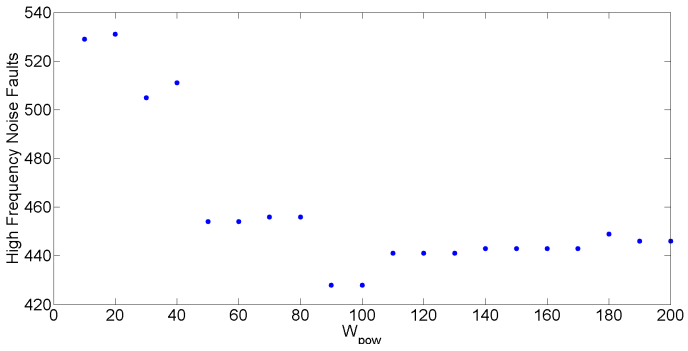

Fig. 19. Number of high-frequency noise faults detected for various values of W_{pow} with $W_{fft} = 512$ and $c_{fft} = 0.2$.

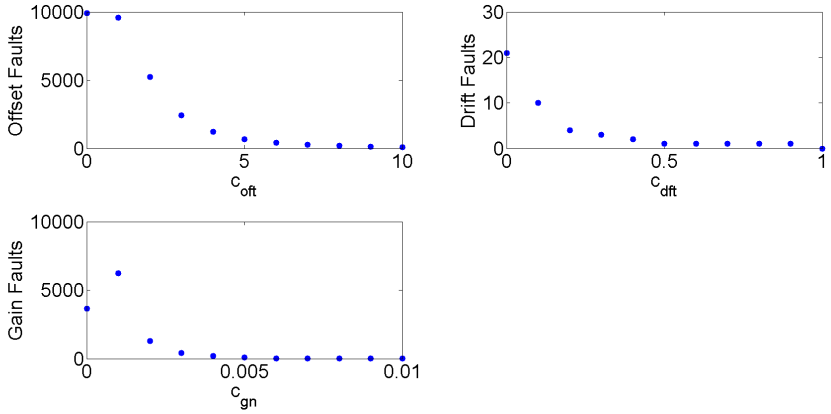

Fig. 20. The number of system-centric faults versus the threshold parameter c_{yyy} with $\beta_{yyy} = 1$; (i) Top left: offset; (ii) Top right: drift; and (iii) Bottom left: gain.

4. Conclusion

We have developed models for common faults in WSNs. It is important to be able to differentiate between true data changes and sensor faults to have a full understanding of a system's health. The models developed here are intended as a *library* of common fault types to be considered when gathering and analyzing data from a WSN. The developed models are designed to work for a wide variety of sensors and applications by tuning and adjusting model parameters. The effects of adjusting these parameters for an industrial temperature sensor were explored. The sensitivity of the models was evaluated using experimental testbed results. While the various fault models were evaluated on the WSN, the methods presented here apply to any distributed electro-mechanical system consisting of multiple sensors, actuators, and networks. The network communication does not need to be wireless.

Future work will create a corresponding library of models for common equipment failures in a system. The models of equipment failures can then be compared to the sensor fault models developed here, and algorithms that differentiate between the two can be developed. Differentiating between true physical system anomalies and sensor faults will allow for more precise monitoring of a system's health. Using novel machine-learning methods for learning healthy sensors' data and later detecting faults is another future research topic. The learning methods can be used to learn the system model or just data. The machine learning techniques can also be considered as a tool to automatically adjust various fault detection parameters currently set manually.

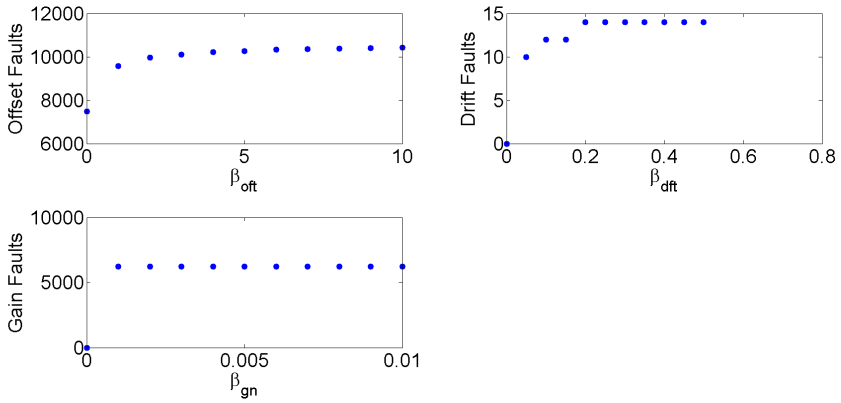

Fig. 21. The number of system-centric faults versus the consistency parameter β_{yyy} with $c_{yyy} = 0.2$; (i) Top left: offset; (ii) Top right: drift; and (iii) Bottom left: gain.

Author Contribution: All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] K. Ni, N. Ramanathan, M. N. H. Chehade, L. Balzano, S. Nair, S. Zahedi, G. Pottie, M. Hansen, M. Srivastava, and E. Kohler, "Sensor network data fault types," *ACM*, vol. 5, no. 3, pp. 1–29, 2009, https://doi.org/10.1145/1525856.1525863.

[2] M. Russell, G. Lecakes, S. Mandayam, and S. Jensen, "The intelligent valve: A diagnostic framework for integrated system-health management of a rocket-engine test stand," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 4, pp. 1489–1497, 2011, https://doi.org/10.1109/TIM.2010.2101350.

[3] G. Kaur, P. Chanak and M. Bhattacharya, "Obstacle-Aware Intelligent Fault Detection Scheme for Industrial Wireless Sensor Networks," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 10, pp. 6876–6886, 2022, https://doi.org/10.1109/TII.2021.3133347.

[4] W. Wu, X. Cheng, M. Ding, K. Xing, F. Liu, and P. Deng, "Localized outlying and boundary data detection in sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1145–1157, 2011, https://doi.org/10.1109/TKDE.2007.1067.

[5] L. Bettencourt, A. Hagberg, and L. Larkey, "Separating the wheat from the chaff: Practical anomaly detection schemes in ecological applications of distributed sensor networks," in *Proceedings of the IEEE International Conference on Distributed Computing in Sensor Systems*, 2007, https://doi.org/10.1007/978-3-540-73090-3_15.

[6] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. Bezdek, "Quarter sphere based distributed anomaly detection in wireless sensor networks," in *Proceedings of the IEEE International Conference on Communications*, 2007, https://doi.org/10.1109/ICC.2007.637.

[7] M. S. Mohamed and T. Kavitha, "Outlier detection using support vector machine in wireless sensor network real-time data," *International Journal of Soft Computing and Engineering*, vol. 1, no. 2, pp. 68–72, 2011, https://www.ijsce.org/portfolio-item/a035041211/.

[8] S. Zidi, T. Moulahi, and B. Alaya, "Fault detection in wireless sensor networks through SVM classifier," *IEEE Sensors Journal*, vol. 18, no. 1, pp. 340–347, 2018, https://doi.org/10.1109/JSEN.2017.2771226.

[9] J. W. Branch, C. Giannella, B. Szymanski, R. Wolff, and H. Kargupta, "In-network outlier detection in wireless sensor networks," in *Knowledge and Information Systems*, vol. 34, pp. 23–54, 2013, https://doi.org/10.1007/s10115-011-0474-5.

[10] K. Zhang, S. SHi, H. Gao, and J. Li, "Unsupervised outlier detection in sensor networks using aggregation tree," in *Proceedings of Advanced Data Mining and Applications*, 2007, https://doi.org/10.1007/978-3-540-73871-8_16.

[11] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. Bezdek, "Distributed anomaly detection in wireless sensor networks," in *Proceedings of the IEEE International Conference on Communication Systems*, 2006, https://doi.org/10.1109/ICCS.2006.301508.

[12] J. Gupchup, A. Terzis, R. Burns, and A. Szalay, "Model-based event detection in wireless sensor networks," *Computing Research Repository*, vol. abs/0901.3923, 2009, https://doi.org/10.48550/arXiv.0901.3923.

[13] Y. Wang, S. Parthasarathy, and S. Tatikonda, "Locality sensitive outlier detection: A ranking driven approach," in *Proceedings of the 27th IEEE International Conference on Data Engineering*, 2011, pp. 410–421, https://doi.org/10.1109/ICDE.2011.5767852.

[14] G. Jin and S. Nittel, "Ned: An efficient noise-tolerant event and event boundary detection algorithm in wireless sensor networks," in *Proceedings of the 7th International Conference on Data Management*, 2006, pp. 153–153, https://doi.org/10.1109/MDM.2006.110.

[15] X. Luo, M. Dong, and Y. Huang, "On distributed fault-tolerant detection in wireless sensor networks," *IEEE Transactions on Computers*, vol. 55, no. 1, pp. 58–70, 2006, https://doi.org/10.1109/TC.2006.13.

[16] S. Kar, R. Tandon, H. Poor, and S. Cui, "Distributed detection in noisy sensor networks," in *Proceedings of the IEEE International Symposium on Information Theory*, 2011, pp. 2856–2860, https://doi.org/10.1109/ISIT.2011.6034097.

[17] H.-T. Pai, J.-T. Sung, and Y. S. Han, "Adaptive retransmission with balanced load for fault-tolerant distributed detection in wireless sensor networks," *Journal of Information Science and Engineering*, vol. 23, pp. 1141–1154, 2007, https://jise.iis.sinica.edu.tw/JISESearch/pages/View/PaperView.jsf?keyId=49_845.

[18] T.-Y. Wang, L.-Y. Chang, D.-R. Duh, and J.-Y. Wu, "Fault-tolerant decision fusion via collaborative sensor fault detection in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 2, pp. 756–768, 2008, https://doi.org/10.1109/TWC.2008.060653.

[19] S. H. Oh, C. O. Hong, and Y.-H. Choi, "A malicious and malfunctioning node detection scheme for wireless sensor networks," *Wireless Sensor Network*, vol. 4, no. 3, pp. 84–90, 2012, https://doi.org/10.4236/wsn.2012.43012.

[20] V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak, "A collaborative approach to in-place sensor calibration," in *Proceedings of the Second International Workshop on Information Processing in Sensor Networks (IPSN)*, 2003, pp. 301–316, https://doi.org/10.1007/3-540-36978-3_20.

[21] M. M. Gharamaleki and S. Babaie, "A new distributed fault detection method for wireless sensor networks," *IEEE Systems Journal*, vol. 14, no. 4, pp. 4883–4890, 2020, https://doi.org/10.1109/JSYST.2020.2976827.

[22] P. -Y. Chen, S. Yang and J. A. McCann, "Distributed real-time anomaly detection in networked industrial sensing systems," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3832–3842, 2015, https://doi.org/10.1109/TIE.2014.2350451.

[23] L. Balzano and R. Nowak, "Blind calibration of networks of sensors: Theory and algorithms," in *Networked Sensing Information and Control*, 2008, pp. 9–37, https://doi.org/10.1007/978-0-387-68845-9_1.

[24] S. U. Jan, Y. D. Lee, and I. S. Koo, "A distributed sensor-fault detection and diagnosis framework using machine learning," *Inf. Sci.*, vol. 547, pp. 777–796, 2021, https://doi.org/10.1016/j.ins.2020.08.068.

[25] M. N. Hasan, S. U. Jan, and I. Koo, "Wasserstein GAN-Based Digital Twin-Inspired Model for Early Drift Fault Detection in Wireless Sensor Networks," *IEEE Sensors Journal*, vol. 23, no. 12, pp. 13327–13339, 2023, https://doi.org/10.1109/JSEN.2023.3272908.

[26] D. Jana, J. Patil, S. Herkal, S. Nagarajaiah, and L. Duenas-Osorio, "CNN and convolutional autoencoder (CAE) based real-time sensor fault detection localization and correction," *Mech. Syst. Signal Process.*, vol. 169, 2022, https://doi.org/10.1016/j.ymssp.2021.108723.

[27] F. Figueroa, J. Schmalzel, J. Morris, M. Turowski, and R. Franzl, "Integrated system health management: Pilot operational implementation in a rocket engine test stand," in *Proceedings of the AIAA Infotech@Aerospace 2010*, 2010, https://doi.org/10.2514/6.2010-3454.

[28] F. Figueroa, J. Schmalzel, R. Aguilar, M. Shwabacher, and J. Morris, "Integrated system health management (ishm) for test stand and j-2x engine: Core implementation," in *Proceedings of the 44th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, 2008, https://doi.org/10.2514/6.2008-5135.

[29] F. Figueroa, J. Schmalzel, M. Walker, M. Venkatesh, R. Kapadia, J. Morris, M. Turowski, and H. Smith, "Integrated system health management: Foundational concepts, approach, and implementation," NASA Stennis Space Center, Tech. Rep., 2009, https://doi.org/10.2514/6.2009-1915.

[30] F. Figueroa, J. Schmalzel, R. Aguilar, M. Shwabacher, and J. Morris, "Tutorial integrated systems health management (ISHM)," in *Proceedings of the NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2011)*, 2011, https://ntrs.nasa.gov/api/citations/20110012035/downloads/20110012035.pdf.

[31] N. Vosburg, R. Selmic, S. Oonk, and F. Maldonado, "Intelligent distributed and ubiquitous health management system: Data storage and processing," in *Proceedings of the AIAA Infotech@Aerospace*, 2013, https://doi.org/10.2514/6.2013-5218.

[32] S. Oonk, F. Maldonado, and T. Politopoulos, "Distributed intelligent health monitoring with the coremicro reconfigurable embedded smart sensor node," in *Proceedings of the International AUTOTESTCON Conference*, Anaheim, California, 2012, pp. 233–238, https://doi.org/10.1109/AUTEST.2012.6334523.

[33] "Coremicro reconfigurable embedded smart sensor node (cre-ssn) product brochure, https://americangnc.com/images/CRE-SSN%20Brochure.pdf, accessed Feb. 2015." American GNC Corporation, Tech. Rep., 2012.

[34] Y. Ju, G. Wei, D. Ding and S. Liu, "A novel fault detection method under weighted try-once-discard scheduling over sensor networks," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 3, pp. 1489–1499, 2020, https://doi.org/10.1109/TCNS.2020.2980362.

[35] C. Han, L. Xu, and G. He, "Mining recent frequent item sets in sliding windows over data streams," *Computing and Informatics*, vol. 27, pp. 315–339, 2008, https://www.cai.sk/ojs/index.php/cai/article/view/252.

[36] B. Xu, Y. Guo, L. Wang, and J. Zhang, "A Novel Robust Gaussian Approximate Smoother Based on EM for Cooperative Localization With Sensor Fault and Outliers," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–14, 2021, https://doi.org/10.1109/TIM.2020.3034977.

[37] S. Safavi, M. A. Safavi, H. Hamid, and S. Fallah, "Multi-Sensor Fault Detection, Identification, Isolation and Health Forecasting for Autonomous Vehicles," *Sensors*, vol. 21, no. 7, p. 2547, 2021, https://doi.org/10.3390/s21072547.

[38] J. M. Rivera-Velazquez, L. Latorre, F. Mailly, and P. Nouet, "A New Algorithm for Fault Tolerance in Redundant Sensor Systems Based on Real-Time Variance Estimation," *IEEE Sensors Journal*, vol. 22, no. 15, pp. 15410–15418, 2022, https://doi.org/10.1109/JSEN.2022.3186636.

[39] H. Lee, Y. Kim and C. O. Kim, "A Deep Learning Model for Robust Wafer Fault Monitoring With Sensor Measurement Noise," *IEEE Transactions on Semiconductor Manufacturing*, vol. 30, no. 1, pp. 23–31, 2017, https://doi.org/10.1109/TSM.2016.2628865.

[40] C. Attaianese, M. D'Arpino, M. D. Monaco and L. P. Di Noia, "Modeling and Detection of Phase Current Sensor Gain Faults in PMSM Drives," *IEEE Access*, vol. 10, pp. 80106–80118, 2022, https://doi.org/10.1109/ACCESS.2022.3195025.

[41] Z. Deng, H. Wang and R. Wang, "An Improved Unscented Kalman Filter for Interrupted and Drift Sensor Faults of Aircrafts," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–10, 2023, https://doi.org/10.1109/TIM.2023.3235423.