Received August 3, 2014, accepted September 6, 2014, date of publication October 13, 2014, date of current version October 27, 2014.

Digital Object Identifier 10.1109/ACCESS.2014.2362756

Fault Diagnosis in Hybrid Electric Vehicle Regenerative Braking System

CHAITANYA SANKAVARAM[1,2], (Member, IEEE), BHARATH PATTIPATI[1], (Member, IEEE), KRISHNA R. PATTIPATI[1], (Fellow, IEEE), YILU ZHANG[2], (Senior Member, IEEE), AND MARK HOWELL[2,3], (Senior Member, IEEE)

[1]Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269 USA
[2]General Motors Global Research and Development, Warren, MI 48090-9055 USA
[3]Trimble Navigation, Christchurch 8024, New Zealand

Corresponding author: C. Sankavaram (chaitanya.sankavaram@gmail.com)

This work was supported by the National Science Foundation (NSF) through the Division of Electrical, Communications and Cyber Systems under Grant CPS/ECCS-0931956 and Grant GOALI/ECCS-1001445.

ABSTRACT Regenerative braking is one of the most promising and environmentally friendly technologies used in electric and hybrid electric vehicles to improve energy efficiency and vehicle stability. This paper presents a systematic data-driven process for detecting and diagnosing faults in the regenerative braking system of hybrid electric vehicles. The diagnostic process involves signal processing and statistical techniques for feature extraction, data reduction for implementation in memory-constrained electronic control units, and variety of fault classification methodologies to isolate faults in the regenerative braking system. The results demonstrate that highly accurate fault diagnosis is possible with the classification methodologies. The process can be employed for fault analysis in a wide variety of systems, ranging from automobiles to buildings to aerospace systems.

INDEX TERMS Automotive systems, distance measure, fault classification, inference, multiple fault diagnosis, regenerative braking system.

I. INTRODUCTION

Hybrid electric vehicles (HEVs) employ regenerative braking to improve fuel economy and vehicle stability. The primary function of the regenerative braking system (RBS) is to convert kinetic energy (from the wheels) into electrical energy during braking mode and store it in batteries for later use in propelling the vehicle [1], [2]. Typically, these systems employ electric motors for both traction and regenerative braking aspects and involve continuous interactions among mechanical/electrical components and their controllers (electronic control units, ECUs) for reliable vehicle operation and control. Faults occurring in these systems may significantly degrade the performance and efficiency of vehicle operation and control. Hence, it is crucial to characterize the nominal and faulty behavior via simulation-based fault injection experiments (also known as *software-in-the-loop* experiments), and subsequently, design detection and inference methodologies for timely diagnosis of faults.

Fault diagnosis methods can be broadly classified into three major categories, viz., physics-based modeling, data-driven and knowledge-based approaches [3]. The *physics-based modeling approach* uses a mathematical representation of the system. This approach employs consistency checks between the sensed measurements and the outputs of a mathematical model. The expectation is that inconsistencies are large in the presence of malfunctions and small in the presence of normal disturbances, noise and modeling errors. Two main methods for generating the consistency checks are based on observers [4], [5] (e.g., Kalman filters, reduced-order unknown input observers, interacting multiple models, particle filters) and parity relations [6] (dynamic consistency checks among measured variables from physical or information redundancy relations). This approach is applicable to systems, where satisfactory physics-based models of the system and an adequate number of sensors to observe the state of the system are available. The main advantage of a model-based approach is its ability to incorporate a physical understanding of the process into the process monitoring scheme. However, it is difficult to apply the model-based approach to large-scale systems because it requires detailed analytical models in order to be effective [3], [7], [8].

A *data-driven approach* is preferred when system models are not available, but system monitoring data for nominal and degraded conditions is available. Neural network and statistical classification methods [9], [10] are illustrative of data-driven techniques. In these cases, experimental data from an operating system or simulated data from a black-box simulator will be the major source of system knowledge for diagnosis. Significant amount of data is needed from monitored variables under nominal and faulty scenarios for data-driven analysis [3], [7], [8].

The *knowledge-based approach* uses graphical models such as dependency graphs (digraphs), Petri nets, multi-signal (multi-functional) flow graphs, and Bayesian networks for diagnostic knowledge representation and inference [11]. A nice feature of knowledge-based approach is that it can embed qualitative and expert knowledge into the model. However, acquiring, managing and maintaining knowledge is a challenge with this approach.

The existing literature suggests that major research effort on HEVs has been on the design and control methodologies for improved energy regeneration [12], optimal braking strategies [13] and system level control to improve energy/fuel efficiency [14]–[16], while significantly less attention has been given to fault analysis in HEV systems. Song et al. [17] discussed a fault-tolerant powertrain topology for series hybrid electric vehicles. They considered short-switch and open-switch faults that compromise the reliable operation of motor drive systems. Parsa and Toliyat [18] proposed a control strategy that provides fault-tolerance to five-phase permanent-magnet motors. It was concluded that the system operated safely under loss of up to two phases without any additional hardware. In [19], a simple on-board fault diagnosis scheme based on reference frame theory is discussed to detect electric motor faults in HEVs during start-up and idle conditions. An observer-based residual generation method for detecting current sensor faults is presented in [20]. Jayabalan et al. [21] proposed the use of statistical moments of higher orders to detect open-circuit and short-circuit faults in cascaded multiple converter systems. A parity relation-based residual generation method for fault detection and isolation of actuators in electric vehicle is discussed in [22].

The existing fault diagnosis approaches for HEVs are component-centric and do not explicitly consider communication aspects and system level interactions. This paper considers hardware, software, and communication faults in the regenerative braking system (RBS) of a HEV, and presents a systematic data-driven methodology for detecting and isolating faults. The faults studied include parametric and sensor-related faults (physical system faults), software logic faults, and network communication faults (missing messages, too many messages, and outdated message faults). A series-parallel drivetrain configuration [23] with regenerative braking is considered. This configuration typically consists of two energy sources for propelling the vehicle: (*a*) an internal combustion engine and an electric generator; and (*b*) an electric motor with battery as the energy storage device [23]. At low speed/power demand, the vehicle runs in electricity-only mode driven by the motor (*series mode*). When the power demand at the wheels is higher, engine together with the motor propels the vehicle (*parallel mode*). More details about different powertrain configurations can be found in [24].

The RBS is modeled using Powertrain System Analysis Toolkit (PSAT), a vehicle simulation software tool [25]. Although the model is designed using PSAT software, it is converted to run independently in MATLAB/Simulink. This model is segregated into multiple ECUs and hardware components to construct a realistic network of embedded systems. The resulting model consists of a powertrain controller (supervisory controller making the high-level decisions that affect the general state of the powertrain), component controllers (ECUs), and the powertrain model (engine, battery, motor, clutch, differential and so on). Communication among the controllers (ECUs) occurs via a controller area network (CAN) bus and the communication aspects of the model are simulated using the Vector CANoe software [26], [27]. The MATLAB/Simulink and Vector CANoe co-simulation environment allowed us to inject faults, evaluate the operational and faulty behavior of the system by monitoring the residuals, and infer/diagnose the failed components.

The fault diagnostic process for RBS primarily involves the following steps: firstly, feature extraction/data reduction techniques are employed to extract the most informative features from the simulation data; secondly, trending and/or threshold-based fault detection tests are designed to detect the occurrence of fault(s); and finally, pattern recognition-based classification techniques [9], [10], distance-based measures and hidden Markov model (HMM) based inference algorithms [28] are applied to isolate faults in the RBS.

The paper is organized as follows. Section II describes the modeling of an RBS and section III presents the data-driven fault diagnostic process to detect faults in the RBS system. The simulation-based fault injection experiments conducted on the RBS model and the related experimental results are discussed in section IV. Finally, section V concludes the paper with a summary.

II. MODELING OF REGENERATIVE BRAKING SYSTEM

The regenerative braking system with two wheel drive series-parallel drivetrain configuration is modeled using PSAT. The RBS model from PSAT is then segregated into multiple ECUs and hardware components. The ensuing model consists of a driver model, a component (physical system) model and six ECUs, namely battery control unit, engine ECU, motor1 control unit, motor2 control unit, mechanical brake control unit, and powertrain controller. These individual models in MATLAB/Simulink are converted into ‘‘dll’’ files using MATLAB Real Time Workshop and then uploaded into the Vector CANoe environment. The simulation setup of ECUs and communication network of RBS is shown in Fig. 1. Communication between the ECUs is carried out via signals on the

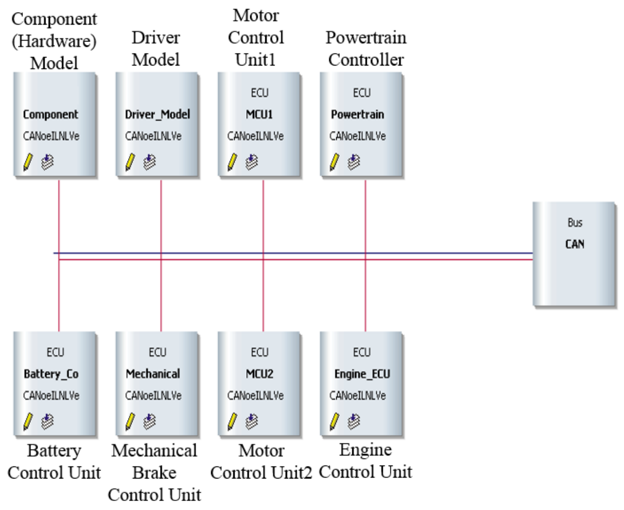


FIGURE 1. Simulation setup of regenerative braking system in vector CANoe environment.

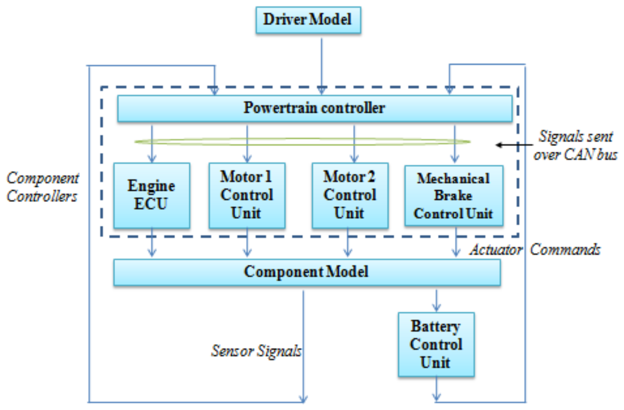


FIGURE 2. Functional flow diagram of RBS.

TABLE 1. List of signals and messages in RBS.

Message Name (From –To)	List of Signals
Motor1Param_ECU (PTC – Motor1 Control Unit)	Motor1 Torque Demand Motor1 Max. Propelling Torque Motor1 Max. Braking Torque
BattParam_ECU (Battery Control Unit – PTC)	Battery State-of-Charge
Motor2Param_ECU (PTC – Motor2 Control Unit)	Motor2 Torque Demand Motor2 Max. Propelling Torque Motor2 Max. Braking Torque
Mechanical_BrkParam_ECU (PTC – Mechanical Brake Control Unit)	Wheel Brake Torque Demand
EngineParam_ECU (PTC – Engine ECU)	Engine on command Engine Torque Demand Max. Engine Torque Min. Engine Torque

CAN bus. Table 1 summarizes the list of signals/messages being transmitted between the ECUs.

The high-level functional block diagram of the RBS model is depicted in Fig. 2. The powertrain controller (PTC) is a high-level controller that makes the supervisory control decisions associated with the vehicle powertrain and accordingly delivers the torque requests to the component controllers (low-level regulatory controllers) based on the component characteristics and its feedback. Subsequently, the low-level controllers, such as the engine controller and motor controller, regulate the corresponding components in the powertrain system. The following subsections briefly describe the functionality of individual component and controller blocks in the RBS model.

A. DRIVER MODEL

The driver model simulates the drive cycles (e.g., Urban Dynamometer Driving Schedule (UDDS) [29]) by setting accelerator and brake pedal positions to achieve the desired vehicle speed. A simple proportional and integral (PI) controller is designed to control the vehicle speed, and a suitable torque demand (τ_{dmd}) is requested that is proportional to the error between the desired and actual vehicle speed. Subsequently, the torque demand is used to request the torque from different powertrain components via the supervisory control algorithm and the requested torque from different powertrain components is regulated by control algorithms in the ECUs.

The driver torque demand at the wheels is given by,

$$\tau_{dmd} = \tau_{variation} + \tau_{loss\_veh} \tag{1}$$

where $\tau_{variation}$ is the torque demand that tracks the changes in the speed demand and allows the vehicle to achieve the desired speed, and τ_{loss_veh} is the torque loss due to vehicle inertia, aerodynamic drag and road grade.
Formally,

$$\tau_{variation} = K_p e(t) + K_i \int e(t)dt \tag{2}$$

and
$$\begin{aligned}\tau_{loss\_veh} &= \tau_{veh\_iner} + \tau_{aero\_drag} + \tau_{road\_grade} \\ &= M\frac{dV_v}{dt}R_w + \frac{1}{2}\rho C_d A_f V_v^2 R_w + C_{rr} M g R_w\end{aligned} \tag{3}$$

where k_p is the proportional gain, k_i is the integral gain, e is the error between the desired and actual vehicle speed, M is the mass of the vehicle, V_v is the velocity of the vehicle, R_w is the wheel radius, ρ is the air density, C_d is the aerodynamic drag coefficient, A_f is the frontal area, C_{rr} is the rolling friction coefficient, and g is the acceleration due to gravity.

B. POWERTRAIN CONTROLLER

The powertrain controller is the supervisory controller that makes high-level decisions, such as engine on/off, operating mode of the vehicle (e.g., propelling mode, regenerative braking mode), and accordingly delivers the torque requests to the component controllers (see Fig. 2). The inputs to the powertrain controller are: driver's torque demand, sensor signals

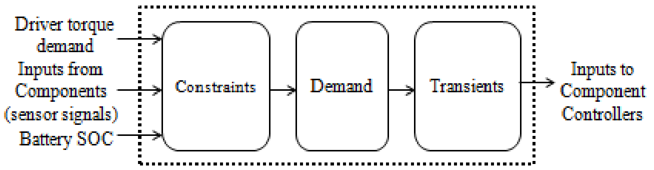

Driver torque demand
Inputs from Components (sensor signals)
Battery SOC
Constraints
Demand
Transients
Inputs to Component Controllers

FIGURE 3. **Powertrain controller.**

from the components, and the battery's state-of-charge (SOC) (Refer to Fig. 3).

PTC consists of three main sub-blocks: *(i) constraints* block, which computes the physical limits for each of the components, for instance, the maximum available torque for the engine or the maximum propelling torque for the generator; *(ii) demand block,* which has control strategies for *propelling* (that is, torque at the wheels is positive), *shifting* (gear selection strategy), and *braking* (that is, torque at the wheels is negative); and *(iii) transient block*, which provides the demand signal in the event of a transient, for example, gear shifting.

C. COMPONENT CONTROL UNITS

In the component controller, the desired torque demands from the PTC are converted into component commands. These commands are processed by the component blocks in the powertrain model. The functions of each of the individual control units are described below:

i. *Engine Control Unit:* Here, the desired torque from the engine is converted into a percentage throttle command.

ii. *Mechanical Brake Control Unit*: This control unit sends braking command to the vehicle component block to achieve the desired torque at the wheels.

iii. *Motor1/Motor2 Control Unit*: This control unit sends percentage of torque requested from the maximum allowable torque to the motor components.

iv. *Battery Control Unit*: In this control unit, the battery current and temperature from the powertrain are used by the SOC estimation algorithm to track the battery SOC (see (8)). This SOC is employed by the powertrain controller for engine on/off control. If the SOC is below a certain threshold, the engine is commanded to stay on, to sustain the battery SOC.

D. POWERTRAIN MODEL

The powertrain model consists of components that mimic the hardware components such as the engine, the battery and the motor. In order to achieve the requested torque, the commands sent from the component controller are treated as the actuator commands by the components in the powertrain model. Each component produces an internal torque that satisfies the demand from the controller, and this torque is communicated to the subsequent component. Fig. 4 shows the individual component blocks in the powertrain model (with a series-parallel drive train configuration), including the clutch, gearbox, and differential which forms the transmission. Each component in the transmission, in turn, makes an internal speed calculation, and reports the system status (e.g., engine speed, motor current/speed) to the supervisory controller. The mathematical modeling of the powertrain components is briefly discussed below [25]:

1) BATTERY MODEL

A simple 0th order (i.e., resistive) battery model is used to estimate the characteristics of the battery. The open circuit

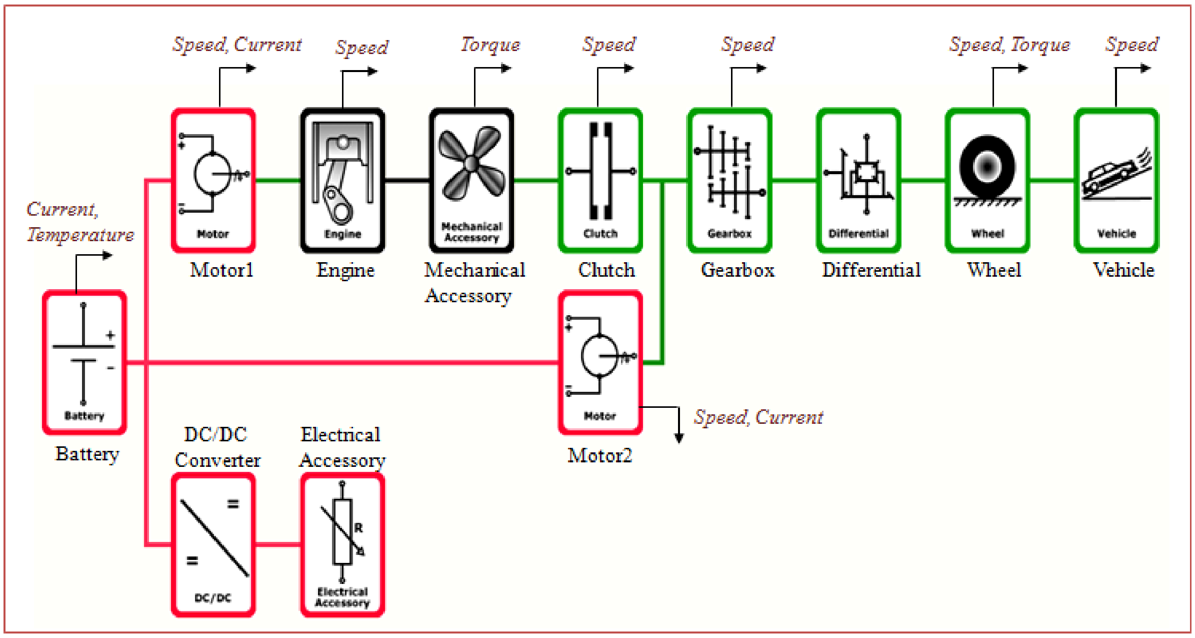


FIGURE 4. **Vehicle powertrain model with series-parallel configuration.**

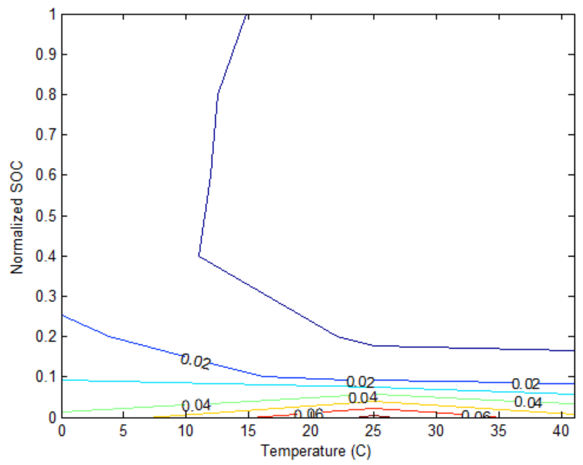


FIGURE 5. Contour plot of discharge resistance as a function of temperature and SOC.

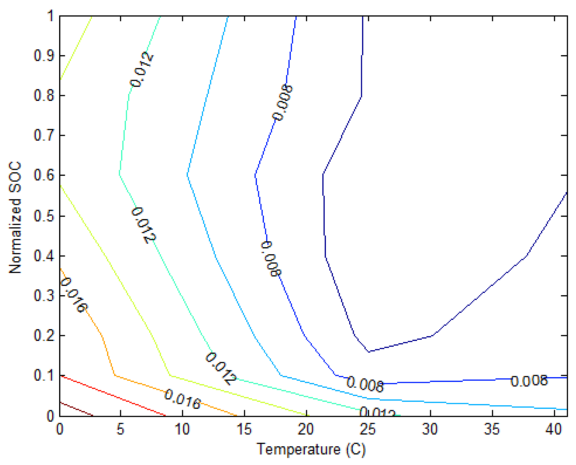


FIGURE 6. Contour plot of charge resistance as a function of temperature and SOC.

voltage of the battery (V_{oc}) is related to the battery terminal voltage via,

$$V_{batt} = V_{oc} - R_{int} I_{batt} \tag{4}$$

where V_{batt} is the effective terminal voltage of the battery, and R_{int} is the internal resistance of the battery. Both V_{oc} and R_{int} are modeled as functions of temperature and SOC. Figures 5-7 show the contours of resistance and V_{oc} as a function of temperature and SOC. The component maps in this paper were obtained using empirical data taken from the PSAT database [25]. When the current is positive, R_{int} is calculated from the charging map and when the current is negative, R_{int} is computed from the discharging map. The battery current is computed using the power supplied by the battery, P_{batt} (or the motor power, P_m) and V_{oc} as follows. From (4),

$$P_{batt} = P_m = I_{batt} V_{batt} = V_{batt} \frac{V_{oc} - V_{batt}}{R_{int}} \tag{5}$$

Solving the resulting quadratic equation for V_{batt}, we obtain

$$V_{batt} = \frac{V_{oc} + \sqrt{V_{oc}^2 - 4R_{int}P_m}}{2} \tag{6}$$

Thus,

$$I_{batt} = \frac{V_{oc} - V_{batt}}{R_{int}} = \frac{V_{oc} - \sqrt{V_{oc}^2 - 4R_{int}P_m}}{2R_{int}} \tag{7}$$

Given the battery current and the temperature, the SOC can be estimated in the battery control unit using (8),

$$SOC(t) = SOC_{initial} + \frac{1}{3600C_{max}} \int_0^t i_{batt}\,dt \tag{8}$$

where $SOC_{initial}$ is the initial SOC of the battery pack, and C_{max} is the maximum capacity in ampere-hour (varies slightly with the battery's internal temperature).

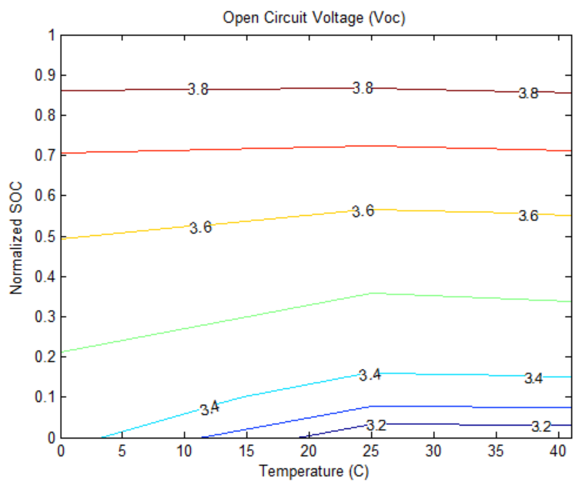


FIGURE 7. Contour plot of open circuit voltage as a function of temperature and SOC.

2) MOTOR MODEL

The electric power needed by the motor is formulated as a function of motor output speed, $\omega_{m,out}$ and output torque, $\tau_{m,out}$. The power supplied to the motor is bounded by the maximum allowable current as in (9).

$$I_m = \frac{P_m}{V_{batt}} = \frac{f(\omega_{m,out}, \tau_{m,out})}{V_{batt}} \tag{9}$$

where V_{batt} denotes the input DC voltage to the motor. The output torque, $\tau_{m,out}$ is calculated from the available torque as,

$$\tau_{m,out} = \tau_{mot,max} m_{cmd}. \tag{10}$$

This maximum torque, $\tau_{mot,max}$ is limited both mechanically and electrically. The mechanical torque limit is,

$$\tau_{max,mech} = \tau_{mot,cont} H_{index} + \tau_{mot,peak}(1 - H_{index}). \tag{11}$$

Here, $\tau_{mot,cont}$ and $\tau_{mot,peak}$ are the continuous and peak torques, which depend on the current shaft speed. Fig. 8 shows the continuous and peak torques as a function of speed.

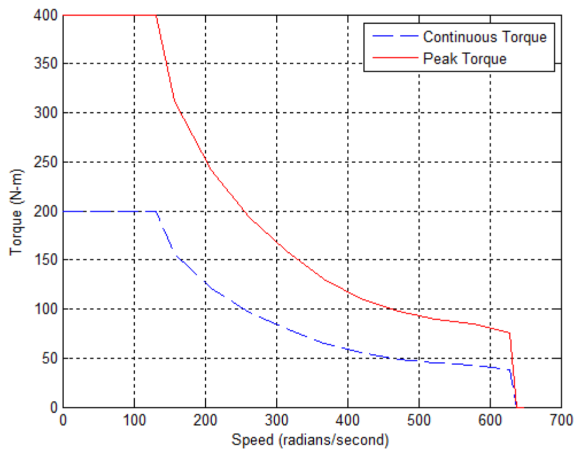


FIGURE 8. Plot of continuous and peak torques of Motor.

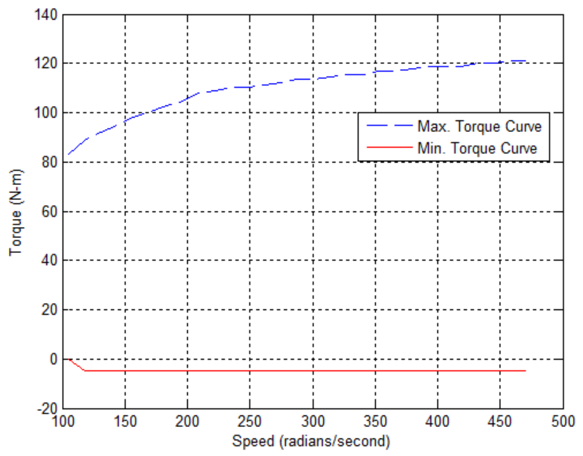


FIGURE 9. Plot of maximum and minimum torque curves of engine.

Depending on the motor operation, the available torque between the peak and continuous torques is fine-tuned via heat index, H_{index}, i.e., when heat index is 1 (i.e., motor is hot), continuous torque curve is used in the torque computation. Similarly, when the motor is at its operating temperature, the maximum torque curve is used. The dynamics of heat index are given by:

$$\frac{dH_{index}}{dt} = \frac{0.3}{t_{mot,max}} \left(\left| \frac{\tau_{m,out}}{\tau_{mot,cont}} \right| - 1 \right) \tag{12}$$

where $t_{mot,max}$ is the motor time constant. The electrical torque limit is implemented as a function of electrical power and the motor speed, i.e., $f(\omega_{m,out}, P_m)$.

3) ENGINE MODEL

The engine model simulates the engine torque. It is calculated from the maximum and minimum torque curves which are mapped as a function of speed. The torque output is given by,

$$\tau_{eng,out} = (1-\theta)\tau_{CTT} + \theta\tau_{WOT} \tag{13}$$

where θ is the normalized engine throttle from the engine control unit, τ_{CTT} is the minimum torque curve of the engine as a function of speed (closed-throttle torque curve), τ_{WOT} is the maximum torque curve of the engine as a function of speed (open-throttle torque curve). If the engine is in the *off* state or at zero throttle, the torque output is zero. Fig. 9 shows the plot of maximum and minimum torque curves as a function of speed.

4) CLUTCH MODEL

The clutch dynamics are given by,

$$J_{cl,in}\frac{d\omega_{cl,in}}{dt} = \tau_{cl,in} - \tau_{cl,out} \tag{14}$$

where $\tau_{cl,in}$ is the input torque from the engine, $\tau_{cl,out}$ is the output torque to the transmission, and $\omega_{cl,in}$ is the angular velocity on the engine side of clutch. The output torque is a function of clutch command and is given by,

$$\tau_{cl,out} = \begin{cases} \tau_{slip\_max}\ cl_{cmd}\ sign(\omega_{cl,in} - \omega_{cl,out}), & \text{if not locked} \\ \tau_{cl,in}, & \text{if locked.} \end{cases} \tag{15}$$

Here, τ_{slip_max} is the maximum friction torque between the clutch plates, cl_{cmd} is the normalized clutch command from the controller (0=unlocked, 1=locked). The clutch is locked (engaged) when the command is equal to 1 and unlocked when the clutch is open. When the clutch is locked, the angular velocities of the engine and transmission input shafts are the same.

5) GEARBOX MODEL

The gearbox is modeled as a gear ratio and a loss term. The torque supplied by the gearbox is given by,

$$\tau_{gb,out} = gb_{ratio}(\tau_{cl,out} - \tau_{gb,loss}). \tag{16}$$

Here, $\tau_{gb,loss}$ is a function of gearbox input torque ($\tau_{gb,in}$, a summation of clutch output torque and the motor1 torque), gearbox output speed ($\omega_{gb,out}$) and gearbox ratio (gb_{ratio}) i.e., $f(\tau_{gb,in}, \omega_{gb,out}, gb_{ratio})$. The gearbox output speed is,

$$\omega_{gb,out} = \frac{\omega_{gb,in}}{gb_{ratio}}. \tag{17}$$

6) FINAL DRIVE MODEL

The wheel torque and gearbox speed are influenced by the differential ratio via:

$$\omega_{gb,out} = \omega_{wh}\, fd_{ratio} \tag{18}$$

$$\tau_{wh,in} = diff_{ratio}(\tau_{gb,out} - \tau_{diff,loss}) \tag{19}$$

where the final drive ratio, fd_{ratio}, and the loss term, $\tau_{diff,loss}$ are known constants.

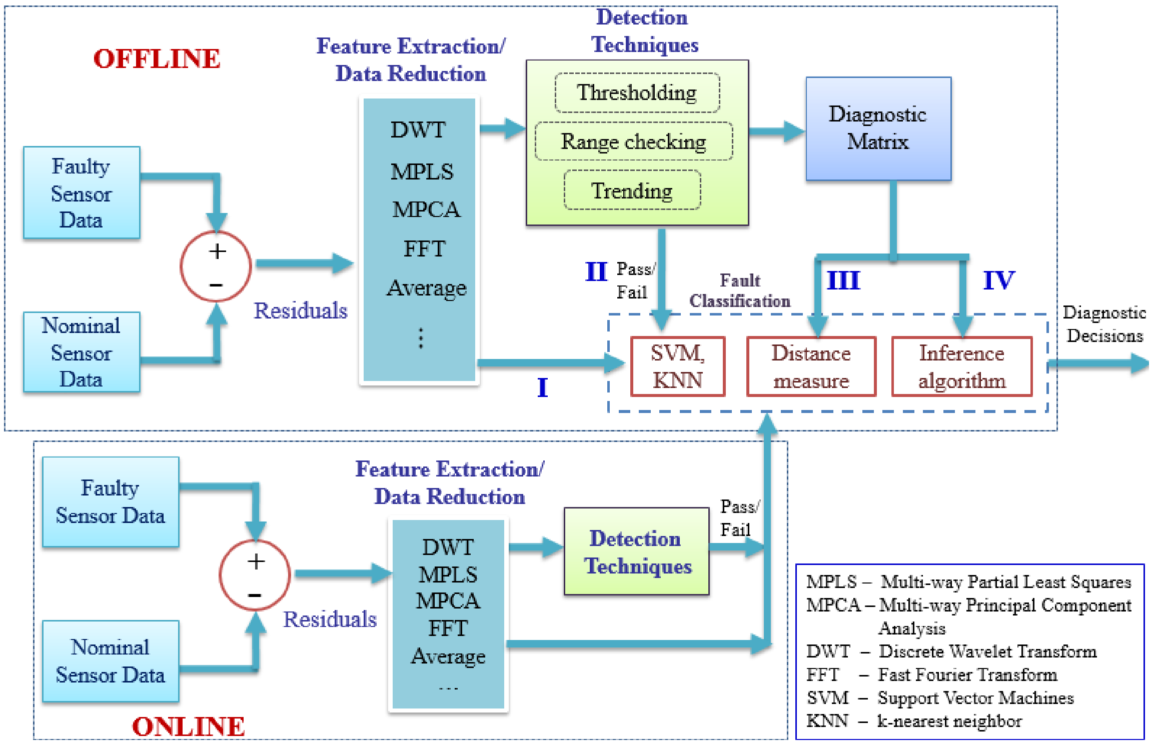


FIGURE 10. **Fault detection and diagnosis process for regenerative braking system.**

7) WHEEL MODEL

The angular speed of the wheel is calculated from the vehicle linear speed using,

$$\omega_{wh} = \frac{V_v}{R_w} \tag{20}$$

where V_v is the vehicle linear speed and R_w is the wheel radius. The wheel torque output can be written in terms of input torque, rolling resistance losses and brake torque as,

$$\tau_{wh,out} = \tau_{wh,in} - \tau_{loss} - \tau_{brake} = \tau_{diff,out} - \tau_{loss} - \tau_{brake}. \tag{21}$$

The brake torque is obtained from the maximum available brake torque via (22) below to meet the desired brake torque demand (normalized brake command) from the mechanical brake control unit. The maximum available brake torque, τ_{max_brake} is assumed to be a constant.

$$\tau_{brake} = wh_{cmd}\ \tau_{max\_brake} \tag{22}$$

8) VEHICLE DYNAMICS MODEL

The vehicle longitudinal dynamics are modeled as follows:

$$\begin{aligned} M\frac{dV_v}{dt} &= F_w - F_{grade} - F_{drag} - F_{brake} \\ &= \frac{\tau_{wh,out}}{R_w} - F_{grade} - F_{drag} - F_{brake} \end{aligned} \tag{23}$$

where V_v and M are the velocity and mass of the vehicle, respectively, and F_{grade} is the force due to weight of the vehicle and road grade angle, θ. Here θ is defined as the angle of inclination between the vehicle and the ground. The F_{grade} is calculated using,

$$F_{grade} = Mg\sin\theta. \tag{24}$$

Here, g is the acceleration due to gravity. The aerodynamic drag force is given by,

$$F_{drag} = \frac{1}{2}\rho C_d A_f V_v^2. \tag{25}$$

The frictional brake force (F_{brake}) is not modeled here. It is assumed that the brakes provide the demand required by the cycle. Appendix A summarizes the component specifications for the series-parallel powertrain configuration.

III. FAULT DETECTION AND DIAGNOSIS PROCESS

The data-driven fault detection and diagnosis (FDD) process for the RBS system consists of an offline training phase and an online testing phase, as shown in Fig. 10. The process involves three major steps: *data reduction, fault detection and fault classification*.

In the off-line training phase, feature extraction/data reduction techniques are employed on the residuals (deviation of actual measurements from the expected ones) from different fault scenarios to extract salient features that capture the most information from the data. Once the features are extracted, the training of fault classifiers can be carried out in one of the following four ways:

(i) Use the extracted features to train fault classifiers, such as the support vector machine (SVM) [30], and the k-nearest neighbor (KNN) [9]; (or)

(ii) Use fault detection techniques [31], viz., trending and thresholding to generate test results (pass/fail [P/F] test outcomes), and use these results to train the fault classifiers; (or)

(iii) Use the test results to learn the diagnostic matrix [11] (D-Matrix) i.e., fault-test dependencies that can be later used online for classifying faults using a distance metric (e.g., Hamming or Euclidean distance measure [32]); (or)

(iv) Use the learned diagnostic matrix in a factorial hidden Markov model (HMM) [33] based inference algorithm to isolate the faults.

In the online testing phase, the trained fault classification algorithms, viz., pattern classification techniques, Euclidean distance-based metric, or an inference algorithm, are used to classify the fault based on the extracted features (reduced data). The following subsections describe briefly the steps involved in the FDD process.

A. FEATURE EXTRACTION/DATA REDUCTION

Data reduction/feature extraction techniques are of significant importance in real world applications to overcome the problems associated with high-dimensional datasets, mainly due to multiple modes of system operation and sensor data collected over time [34], [35]. In Fig. 10, the feature extraction block includes several signal processing algorithms and statistical techniques, such as, multi-way partial least squares (MPLS), multi-way principal component analysis (MPCA), discrete wavelet transform (DWT), and fast Fourier transforms (FFT) to extract fine-grained information for diagnosing faults. In our experiments, feature extraction is done by applying wavelet transformation, MPLS, and statistical quantities such as time averaged or root mean square (RMS) values of the residuals.

Wavelets have proven to be extremely useful feature extractors in recovering fine-grained mode-invariant information from the data. They not only reduce the data size, but also aid in capturing salient events in a signal, such as trends, discontinuities, breakdown points, self-similarity, etc., and this characteristic of wavelets results in the generation of a strong feature set for classification [36]–[38].

Data-reduction techniques, such as MPCA and MPLS, transform highly correlated data to a smaller number of relevant and reduced set of transformed variables, called score vectors. These score vectors capture a significant amount of variation in the data. Both MPLS and MPCA techniques can be used for fault detection. For example, in the MPLS technique, if the distance of the score vectors is close to the origin of the reduced feature space, then the system is in the nominal state [39], [40]. The distance to these score vectors from the origin can be computed via Hotelling statistic [41]. In this work, MPLS technique is primarily employed for data reduction.

B. FAULT DETECTION

Fault detection is performed by monitoring the amount/rate of deviation of a parameter from its nominal value (also known as residuals). Tests such as cumulative sum (CUSUM) test [42] and simple thresholds are used for the detection of faults. Here, at each time instant k, the residuals of monitored variables are computed within a measurement window of size L, and the cumulative sum of squared residuals is calculated. Let S_{ik} be the cumulative sum of squared residual of monitored signal i at time k, mathematically written as,

$$S_{ik} = \sum_{j=k-L}^{k} r_{ij}^2 \tag{26}$$

where r_{ij} is the residual of the signal i at j^{th} time instant, and, is the difference between the actual sensor output and nominal no-fault output. In the no-fault case, the expected value of r_{ij} is zero, whereas in the presence of a fault, the r_{ij} does not have zero-mean. Hence, a simple threshold test on the cumulative sum of the squared residuals can determine whether the system is in a faulty state or nominal state, i.e., if S_{ik} exceeds a threshold T then a fault is declared (see (27)).

$$\begin{aligned} S_{ik} &> T \rightarrow \text{ fault detected} \\ S_{ik} &\leq T \rightarrow \text{ no fault detected} \end{aligned} \tag{27}$$

To detect communication faults, i.e., missing messages and too many messages, a monitoring technique based on message rate is employed. Since the messages from the controller are periodic, and the periodicity is predefined, monitoring the number of messages (message rate) that are received within a measurement window would ascertain whether the messages are being lost or if too many messages are being sent. If the message rate in an interval is below a predefined nominal threshold (Th_{low}) then it is determined that message loss fault has occurred. In the same way, if the message rate in an interval is above a predefined nominal threshold (Th_{high}) then a fault corresponding to too many messages has occurred in the system. Here, Th_{low} and Th_{high} are the low and high nominal thresholds for the message rate. When the message rate is between these two thresholds, the system is considered to be in the nominal state.

Once a fault is detected in the system, fault classification algorithms are then employed to accurately isolate the fault.

C. FAULT CLASSIFICATION

The fault classification is performed using the following techniques (see Fig. 10):

(i) Pattern classification techniques, such as the support vector machines (SVM) and k-nearest neighbors (KNN);

(ii) Euclidean distance measure; and/or

(iii) Inference algorithm.

The following subsections describe briefly the above mentioned techniques.

1) PATTERN CLASSIFICATION TECHNIQUES

Here, support vector machine and the k-nearest neighbor are employed for fault classification.

a: SUPPORT VECTOR MACHINE (SVM)

SVM is one of the most widely used supervised learning algorithms for classification. Given a set of training samples belonging to different classes, SVM finds an optimal hyperplane that maximizes the margin between the classes. Hence, SVM is also known as the maximum-margin classifier. In the case of non-linear classification, where a linear boundary is not appropriate, a kernel function is used for mapping the data onto a higher dimensional space. An optimal hyperplane is then constructed in the new space. Some of the kernel functions that are commonly used are polynomial, radial basis functions, and hyperbolic tangent [30].

b: k-NEAREST NEIGHBOR (KNN)

KNN classifier is a simple non-parametric method that classifies test vectors based on the samples from the training data. The classifier finds the k-nearest points to a test vector from the training data, and the class with the maximum *a posteriori* probability within those k points is declared as the most-likely class. Normally, k is chosen as an odd number to avoid ties [9].

2) DISTANCE MEASURE

In this method, a diagnostic matrix (D-Matrix) is generated for all the faults in the training phase based on detection techniques (or tests) such as thresholding, trending, or range-checking on the features extracted from the residuals. Once the diagnostic matrix is generated, similar tests can be applied online in the testing phase to generate binary outcomes for each of the tests. Subsequently, the distance between the code generated from the test pattern and each entry of the diagnostic matrix can be measured using distance metrics, such as the Hamming distance or the Euclidean distance [32]. The test pattern is classified as the fault that has the nearest distance with the entry in the diagnostic matrix. Mathematically, it is given as,

$$\begin{aligned}\text{Euclidean Distance} &= \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}\\ \text{Hamming Distance} &= \frac{\text{no. of positions that are different}}{\text{total number of positions}}\\ &= \frac{\#(p_i \neq q_i)}{n} \end{aligned} \tag{28}$$

where $\underline{p}$ and $\underline{q}$ are row vectors of length n.

3) INFERENCE ALGORITHM FOR FAULT DIAGNOSIS

The dynamic multiple fault diagnosis (DMFD) is a factorial HMM based inference algorithm to infer the evolution of component fault states given a set of uncertain test outcomes (pass, fail) at each epoch. Formally, the dynamic multiple fault diagnosis problem is represented as $DMFD = \{S, T, O, D, P, A\}$, where $S = \{s_1, \ldots, s_m\}$ is a finite set of components (failure sources) associated with the system. The state of component s_i is denoted by x_i, where $x_i(k) = 1$ if s_i is faulty at epoch k; and $x_i(k) = 0$ otherwise. The status of all components at epoch k is denoted by $\underline{x}(k) = [x_1(k), x_2(k), \ldots, x_m(k)]$. Similarly, $T = \{t_1, t_2, \ldots, t_n\}$ represent a finite set of available tests where $t_j(k) = 1$ represents that the test t_j is in failed state at time epoch k; and $t_j(k) = 0$ otherwise. Here $\kappa = \{0, 1, \ldots, k, \ldots, K\}$ is the set of discretized observation epochs. The initial state $\underline{x}(0)$ is assumed to be known. The observations $O = \{o_1, o_2, \ldots, o_n\}$ are the (pass/fail) test outcomes. For each component state, e.g., for component s_i at epoch k, $A = (Pa_i(k), Pv_i(k))$ denotes the set of fault appearance probability and fault disappearance probability defined as $\Pr(x_i(k) = 1|x_i(k-1) = 0)$ and $\Pr(x_i(k) = 0|x_i(k-1) = 1)$ respectively. Also, $P = (Pd_{ij}, Pf_{ij})$ represent probability of detection and probability of false alarm associated with each test outcome j and fault class i.

This DMFD problem can be formulated as one of maximizing the *a posteriori* probability:

$$\hat{X}^K = \arg\max_{X^K} \Pr(X^K/O^K) \tag{29}$$

where $X^K = [\underline{x}(k)]_{k=1}^{K}$, $O^K = [\underline{o}(k)]_{k=1}^{K}$, K = total number of epochs and $\underline{o}(k) = [o_1(k), o_2(k), \ldots, o_n(k)]$.

The solution is a primal-dual optimization framework that employs Lagrangian relaxation method for decomposing the original DMFD problem into parallel decoupled subproblems, one for each fault. Each subproblem corresponds to finding the optimal fault-state sequence, which is solved using a binary Viterbi decoding algorithm. The subproblems are coordinated by updating the Lagrange multipliers using a subgradient method.

The inputs to the algorithm are test outcomes $\underline{o}(k)$ at each sampling time k, their reliabilities, and a fault diagnostic matrix (D-matrix, D) that represents the dependency relations between failure modes and tests. The test outcomes could be statistical test decisions derived from sensor data. These test outcomes, together with the fault-dependency matrix, are processed through a primal-dual optimization method that exploits temporal correlations of test outcomes over time for inference. A detailed description of DMFD algorithms may be found in our previous work (see formulation 1 in [28]).

IV. SIMULATION-BASED FAULT INJECTION EXPERIMENTS, RESULTS AND DISCUSSION

The RBS model described in Section II is simulated under nominal and faulty conditions with driver profiles: *Profile*1 and *Profile*2 shown in Figures 11 and 12. The driver profiles (drive cycles) are collected from a Chevy 2007 model year vehicle and act as inputs to the driver block of the RBS model. The list of faults that are considered for fault injection experiments in the RBS is provided in Table 2. The parametric and sensor fault scenarios are simulated as a deviation of $100\Delta\%$

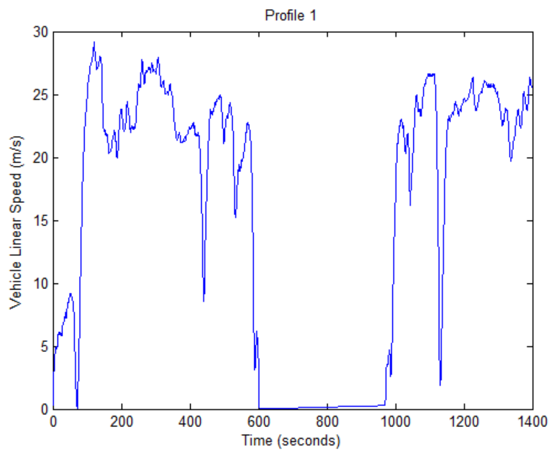


FIGURE 11. Driver profile 1.

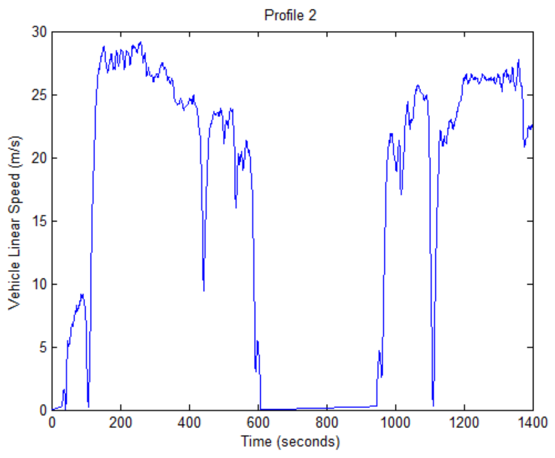


FIGURE 12. Driver profile 2.

TABLE 2. List of faults.

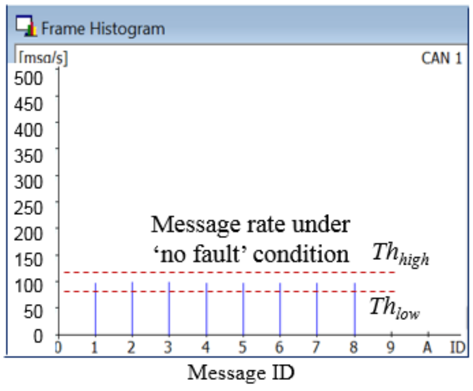
Fault: Fault Description
F1: Nominal; **F2:** Battery Current Sensor Fault; **F3:** Battery Temperature Sensor Fault; **F4:** Engine Speed Sensor Fault; **F5:** Motor1 Current Sensor Fault; **F6:** Motor1 Speed Sensor Fault; **F7:** Vehicle Speed Sensor Fault; **F8:** Wheel Inertia Fault; **F9:** Engine Message Loss Fault; **F10:** Burst Loss of Engine Message; **F11:** Burst Loss of Messages from PTC; **F12:** Too Many Messages From Battery; **F13:** Battery Initial SOC Fault; **F14:** Wheel Radius Fault; **F15:** Engine Message Faulty Data; **F16:** Motor1 Message Faulty Data; **F17:** Wheel Message Faulty Data

from their nominal values using the following equation,

$$X_{faulty} = X_{nominal}(1 + \Delta). \tag{30}$$

In (30), X_{faulty} is the parameter value under faulty condition, $X_{nominal}$ is the nominal parameter value and Δ is the fractional change in the parameter value (fault severity). Here, Δ is chosen to be 0.1 in all the scenarios. The sensor faults are simulated as additive biases on the sensor signals. Since the sensor measurements are used by the supervisory controller for system-level control, faults in sensors could lead to wrong feedback information being sent to the powertrain controller.

The network faults are simulated through Controller Area Network Access Programming Language (CAPL) environment in the Vector CANoe software. M*essage loss fault* is simulated with 12% loss rate, i.e., 12% of the messages are dropped in an interval of 1 second. The *burst loss faults* are simulated in such a way that several messages are lost in a time frame whereas in *too many messages* fault ('babbling idiot' fault), an ECU sends repeated messages onto the bus. In addition, the faulty data message faults are simulated as though the ECUs transmit an outdated message (i.e., faulty data value with respect to current time instant). The histograms show the message rate for the nominal scenario (Fig. 13). Here, Th_{low} and Th_{high} are the low and high thresholds for the message rate within which the message communication is considered to be normal. There are 25 monitored signals, including speed, torque, current and temperature as well as signals from the controllers (see Table 3 for the list of monitored signals). The fault simulation data is arranged in the form of a tensor as $\underline{X} \in R^{I\times J\times K}$ where I, J and K are the nominal and fault cases, measured (monitored) signals, and time samples, respectively.



Message ID	Source (S) – Destination (D) Node (ECU)
1, 5	PTC (S) – Motor1 Control Unit (D)
2	Battery Control Unit (S) - PTC (D)
3, 8	PTC (S) – Motor2 Control Unit (D)
4	PTC(S) - Mechanical Brake Control Unit (D)
6, 7	PTC (S) – Engine ECU (D)

FIGURE 13. Histogram of CAN bus message rate.

TABLE 3. List of monitored signals.

Monitored Signal: Signal Description
S1: Battery SOC; **S2:** Motor2 Torque Demand; **S3:** Wheel Torque Demand; **S4:** Motor1 Torque Demand; **S5:** Engine Torque Demand; **S6:** Battery Temperature; **S7:** Battery Current; **S8:** Driver Torque Demand; **S9:** Motor1 Command; **S10:** Gearbox Speed; **S11:** Wheel Input Speed; **S12:** Wheel Output Speed; **S13:** Wheel Torque; **S14:** Vehicle Linear Speed; **S15:** Motor1 Speed; **S16:** Motor1 Current; **S17:** Clutch Input Speed; **S18:** Engine Command; **S19:** Motor2 Command; **S20:** Motor2 Speed; **S21:** Motor2 Current; **S22:** Engine Speed; **S23:** Clutch Output Speed; **S24:** Mechanical Accessory Torque; **S25:** Wheel Command

The simulations are conducted for 1400 seconds (length of the driver profile) at a sampling time of 0.1 seconds. Hence, there are a total of 14001 data points for each nominal and faulty case. Consequently, the dimensionality of total data collected from the fault simulations is $17 \times 25 \times 14001$ for each profile (17 – nominal and faulty cases, 25 – monitored signals for each fault case, and 14001 – data points). A Gaussian measurement noise with random seed is added to the signals with a variance of 0.6% of the squares of the magnitude of the signals (this corresponds to a signal-to-noise (SNR) ratio of 22.2dB).

The fault detection, data reduction and fault classification results for different diagnostic methodologies are briefly discussed in the following subsections.

A. FAULT DETECTION

The faults in RBS system are detected via a test on sum of squared residuals. First, the cumulative sums of the squared residuals for the 25 monitored variables within a window of size L are collected. Here L is chosen to be 50 samples (5 second duration at 0.1 second sampling interval). A threshold is defined and the cumulative sum of the squared residual is compared against this threshold to determine whether the system is in a faulty state or nominal state. Fig. 14 shows the cumulative sum of the squared residuals for a battery current sensor fault between 851 and 900 samples, i.e., 85.1–90 seconds (viz., fifty samples in a 5 second interval). Here, x-axis represents the sample number and y-axis represents the cumulative sum of squared residuals for the 25 monitored signals. A suitable threshold on the residuals detects the presence of fault(s), for instance, a threshold of 500 on signal S7 indicates the presence of a fault at $t = 89.4$ seconds. In this work, test on residuals is employed only for fault detection.

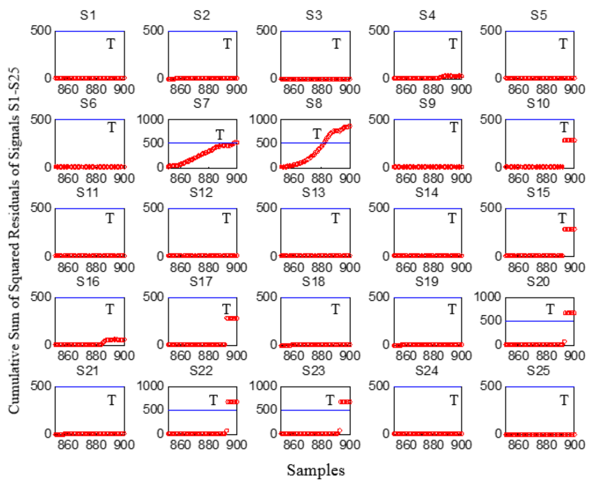
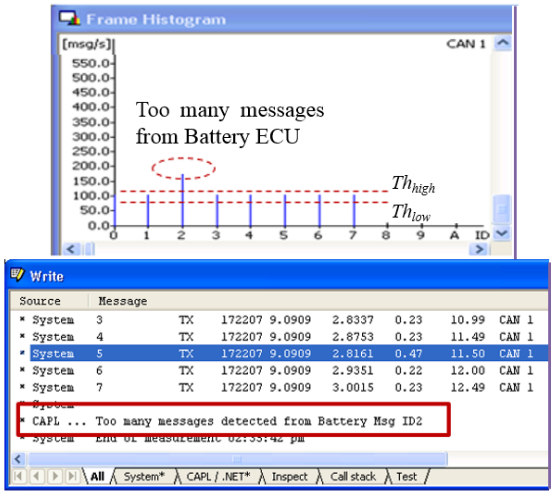


FIGURE 14. Cumulative sum of squares of residuals of signals S1-S25 for battery current sensor fault (F2).

Once a fault is detected, data reduction techniques and classification techniques are employed to isolate the fault.

Fig. 15 shows the message rate when the too many message fault is injected into the system. As described in Section III-B,



```
Source    Message
* System   3          TX     172207 9.0909   2.8337   0.23     10.99  CAN 1
* System   4          TX     172207 9.0909   2.8753   0.23     11.49  CAN 1
* System   5          TX     172207 9.0909   2.8161   0.47     11.50  CAN 1
* System   6          TX     172207 9.0909   2.9351   0.22     12.00  CAN 1
* System   7          TX     172207 9.0909   3.0015   0.23     12.49  CAN 1
* System
* CAPL ... Too many messages detected from Battery Msg ID2
* System   End of measurement 02:33:42 pm
```

FIGURE 15. Message rate for too many messages fault and its fault detection.

when the message rate is above a predefined nominal threshold, the too many messages fault is detected. The variables are defined to store the detection results and the fault detection message is displayed through CAPL environment as shown in Fig. 15.

B. DATA REDUCTION AND FAULT CLASSIFICATION RESULTS

5×2 cross validation is employed in which a 2-fold cross validation is carried out on a randomized dataset and the process is repeated for 5 times. That is, initially, the randomized dataset is split into two equally sized subsets $\{\Omega_1, \Omega_2\}$ and the classifier is trained and tested twice: first, Ω_1 is used to train the classifier and Ω_2 to test the classifier; then, the roles of the subsets are reversed. This procedure is repeated five times and the results of ten tests are averaged to obtain the final fault classification results [9], [10]. The final classification performance is evaluated using the metric, classification accuracy (CA), defined as the average percentage of faults that are correctly classified by the algorithm. The following subsections discuss the performance of the various classification methodologies 1 through 4 depicted in Fig. 16.

1) METHOD 1

In this method, data reduction technique MPLS is first employed on the residuals to reduce the high dimensional data into fewer score components (717.6 MB $\rightarrow$ 12.4 KB). The data is reduced to 4 score vectors and the reduced data is used for training fault classification algorithms, SVM and KNN. Fig. 16a shows the pictorial representation of the fault isolation process. The trained SVM model employs a radial basis function (RBF) kernel with a standard deviation (kernel parameter, σ) of 0.001 and box constraint (soft margin parameter, C) of 100, whereas the KNN algorithm employs 3-nearest neighbor rule for classification. The total number

TABLE 4. Classification results.

Method	Classification Algorithm	Classification accuracy ± standard deviation (%)	Average testing time per pattern (seconds)
1	SVM	**95.7 ± 0.6**	0.0000598
	KNN	94.7 ± 0.9	0.000353
2	SVM	91.3 ± 0.7	**0.0000588**
	KNN	90.1 ± 1.0	0.000294
3	Hamming distance/ Euclidean distance	90.5 ± 1.2	0.00143
4	Inference Algorithm	93.8 ± 0.6	0.00189

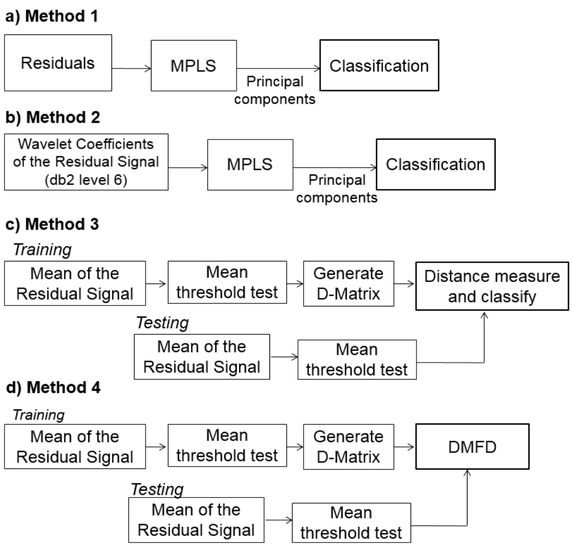


FIGURE 16. Fault classification process methods 1-4.

of patterns available is 204 (34 patterns from Profiles 1 and 2 and 170 patterns by adding noise to the measured signals). This data is divided into two equal subsets for training and validating the classification algorithms, i.e., 50% is used for training and the remaining 50% of the data is used for validating the trained classification algorithms via the 5×2 cross validation process. Table 4 summarizes the classification results for Method 1 using the two classifiers. The correct classification rate with SVM is 96% and with KNN it is 95%. The average testing time per test pattern is also provided in Table 4.

2) METHOD 2

In this method, Daubechies 2 (db2) wavelet techniques are employed to extract detailed coefficients (level 6) for each pattern of the residual matrix. These coefficients are then reduced to 4 score vectors via MPLS and the reduced data is used for training the classification algorithms as shown in Fig. 16b. Here, the data is reduced from 12.36 MB to 8.2 KB. Similar to Method 1, the data (204 patterns) is divided into training and testing subsets for cross validation. The classification accuracy with SVM model (parameters – C: 100 and σ: 0.0001) is 91% whereas with KNN ($k = 3$) classifier, it is 90% (see Table 4).

3) METHOD 3

In Method 3, a simple mean threshold test on the residuals is used as a detection technique in the training phase to generate the diagnostic matrix (i.e., a D-matrix) for all the faults. A snapshot of the D-matrix with 25 tests, 16 faults, and 1 nominal is provided in Appendix B. In the testing phase, these tests are applied to generate binary codes. Euclidean distance criterion (and Hamming distance) is then used to measure the distance between the code generated from the test pattern and each entry of the diagnostic matrix. Subsequently, the test pattern is classified as the fault that has the nearest distance with the entry in the diagnostic matrix (Refer Fig. 16c for the process). The classification accuracy using a 5×2 cross validation process with the Euclidean distance measure criterion is 90.5% (see Table 4).

4) METHOD 4

Here, similar to Method 3, threshold tests are applied to generate the test outcomes. The test outcomes are input to the DMFD inference algorithms in [28] and briefly described in Section III-C3 to infer the component failures (see Fig. 16d for the process). Using the D-matrix in Appendix B, the DMFD algorithm inferred the faults with an average isolation accuracy of 93.8% (see Table 4).

The diagnosis/classification results from Methods 1-4 indicate that all of the 16 faults can be isolated with at least 90% accuracy. The algorithms are implemented in MATLAB and the experiments are performed on a Pentium 2.80 GHz processor. The average time required for the simulations can be reduced by approximately a factor of ten using the C programming language. Although the data obtained in this paper to validate the data-driven diagnostic process is collected from a simulation model built in MATLAB/Simulink, it is equivalent to sensor signals obtained over time from a vehicle through telematics services in real time. It should be noted that the fault classification problem can be solved sequentially with a smaller window size [28]. The sample of faults and signals considered in this paper are to test the viability of the presented approach. It is necessary to explore more realistic scenarios, such as different driving conditions, road scenarios etc., and validate the proposed approach. However, preliminary experimental results are promising. The systematic FDD process presented in this paper featuring fault detection, multivariate statistical data reduction and fault classification is a generic approach for fault analysis and can be easily applied to any real-world engineering system. Indeed, simpler variants of this process have been applied to a number of engineering systems [39], [40], [43].

V. CONCLUSION

In this paper, a systematic data-driven fault detection and diagnosis approach is presented for fault diagnosis in components, control software, and communication network associated with an automotive regenerative braking system. The RBS simulation model is developed using the PSAT software. Various fault injection experiments are conducted through the co-simulation of MATLAB and Vector CANoe. We employed wavelet-based feature extraction and MPLS-based data reduction to extract salient fault discriminating information, and subsequently, different diagnosis algorithms such as pattern recognition techniques, Euclidean distance criteria, and inference-based algorithms are used for classifying faults in the RBS simulation model. The approach presented here demonstrated good diagnostic accuracy and can be used for fault analysis in any vehicle system. It has the potential for real-time implementation in automotive and aerospace systems due to the significant reduction in the data size without compromising fault classification accuracy at a reduced computational load/time. In the future, we plan to investigate the robustness of the FDD scheme with varying driver profiles. In addition, we plan to predict the degradation of RBS parameters and estimate the remaining useful life of components.

APPENDIX

Appendix A. Component specifications of series-parallel drivetrain configuration.

COMPONENT	SPECIFICATIONS
Motor1	Permanent Magnet (PM) Electric Motor with Continuous Power = 25KW Peak Power = 50 KW
Motor2	UQM Power-phase PM motor Continuous Power = 55KW Peak Power = 100 KW
Engine	Gasoline Engine 1.497 liter 57 KW
Energy Storage	SAFT Li-ion Battery Capacity = 6 Ah Number of cells = 75
Gearbox	2 Gear Manual Transmission Gear ratios = 1.86 and 1
Wheel Axle	2 Wheel drive
Vehicle	Vehicle body mass = 800 Kg Frontal area = 1.8 m^2 Drag coefficient = 0.38

Appendix B. Snapshot of diagnostic matrix (F1: *nominal*, F2-F17: *faults*, S1-S25: *monitored signals*).

	S1	S2	S3	S4	S5	S6	S7	S8	...	S21	S22	S23	S24	S25
F1	0	0	0	0	0	0	0	0		0	0	0	0	0
F2	0	1	1	0	0	0	1	1		0	0	0	0	0
F3	0	0	0	0	1	1	0	0		1	1	0	0	0
F4	0	0	0	0	1	1	0	0	...	0	0	1	0	0
F5	0	0	0	0	0	0	0	1		1	1	1	0	0
F6	0	0	0	0	1	1	1	0		1	1	1	0	0
…				…					…				…	
F12	0	0	0	1	1	1	0	0		0	0	1	0	0
F13	0	0	0	0	1	1	1	0		0	0	0	0	0
F14	0	0	0	1	0	1	1	0	...	1	1	0	0	0
F15	0	0	1	0	1	1	1	1		0	0	0	1	0
F16	1	1	1	0	1	0	1	1		0	0	0	0	0
F17	0	0	0	1	1	1	1	0	...	0	0	0	0	1

ACKNOWLEDGMENT

The authors would like to thank GM R&D and NSF for their support of this work. Any opinions expressed in this paper are solely those of the authors and do not represent those of the sponsor.

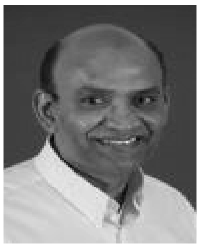
REFERENCES

[1] J. K. Ahn, K. H. Jung, D. H. Kim, H. B. Jin, H. S. Kim, and S. H. Hwang, "Analysis of a regenerative braking system for hybrid electric vehicles using an electro-mechanical brake," *Int. J. Autom. Technol.*, vol. 10, no. 2, pp. 229–234, 2009.
[2] S. R. Cikanek and K. E. Bailey, "Regenerative braking system for a hybrid electric vehicle," in *Proc. Amer. Control Conf.*, Anchorage, AK, USA, May 2002, pp. 3129–3134.
[3] K. R. Pattipati *et al.*, "An integrated diagnostic process for automotive systems," in *Computational Intelligence in Automotive Applications* (Studies in Computational Intelligence), vol. 132, D. Prokhorov, Ed. Berlin, Germany: Springer-Verlag, 2008, pp. 191–218.
[4] G. Ciccarella, M. Dalla Mora, and A. Germani, "A Luenberger-like observer for nonlinear systems," *Int. J. Control*, vol. 57, no. 3, pp. 537–556, 1993.
[5] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation With Applications to Tracking and Navigation*. New York, NY, USA: Wiley, 2001.
[6] J. Gertler, "Fault detection and isolation using parity relations," *Control Eng. Pract.*, vol. 5, no. 5, pp. 653–661, 1997.
[7] C. Sankavaram, A. Kodali, D. F. M. Ayala, K. Pattipati, S. Singh, and P. Bandyopadhyay, "Event-driven data mining techniques for automotive fault diagnosis," in *Proc. 21st Int. Workshop Principles Diagnosis*, Portland, OR, USA, Oct. 2010, pp. 1–8.
[8] C. Sankavaram *et al.*, "Model-based and data-driven prognosis of automotive and electronic systems," in *Proc. 5th Annu. IEEE Conf. Autom. Sci. Eng.*, Bangalore, India, Aug. 2009, pp. 96–101.
[9] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York, NY, USA: Wiley, 2001.
[10] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006.
[11] J. Luo, H. Tu, K. Pattipati, L. Qiao, and S. Chigusa, "Diagnosis knowledge representation and inference," *IEEE Instrum. Meas. Mag.*, vol. 9, no. 4, pp. 45–52, Aug. 2006.
[12] X. Huang and J. Wang, "Model predictive regenerative braking control for lightweight electric vehicles with in-wheel motors," *J. Autom. Eng.*, vol. 226, no. 9, pp. 1220–1232, Apr. 2012.
[13] Y. Gao and M. Ehsani, "Design and control methodology of plug-in HEVs," *IEEE Trans. Ind. Electron.*, vol. 57, no. 2, pp. 633–640, Feb. 2010.

[14] J. Liu and H. Peng, "Modeling and control of a power-split hybrid vehicle," *IEEE Trans. Control Syst. Technol.*, vol. 16, no. 6, pp. 1242–1251, Nov. 2008.
[15] R. Ghorbani, E. Bibeau, P. Zanetel, and A. Karlis, "Modeling and simulation of a series parallel hybrid electric vehicle using REVS," in *Proc. Rec. Amer. Control Conf.*, Jul. 2007, pp. 4413–4418.
[16] H. A. Borhan and A. Vahidi, "Model predictive control of a power-split hybrid electric vehicle with combined battery and ultracapacitor energy storage," in *Proc. Rec. Amer. Control Conf.*, 2010, pp. 5031–5036.
[17] Y. Song and B. Wang, "Analysis and experimental verification of a fault-tolerant HEV powertrain," *IEEE Trans. Power Electron.*, vol. 28, no. 12, pp. 5854–5864, Dec. 2013.
[18] L. Parsa and H. A. Toliyat, "Fault-tolerant interior-permanent-magnet machines for hybrid electric vehicle applications," *IEEE Trans. Veh. Technol.*, vol. 56, no. 4, pp. 1546–1552, Jul. 2007.
[19] B. Akin, S. B. Ozturk, and H. A. Toliyat, "On-board fault diagnosis of HEV induction motor drive at start-up and during idle mode," in *Proc. IEEE Veh. Power Propuls. Conf.*, Sep. 2007, pp. 140–147.
[20] K. Rothenhagen and F. W. Fuchs, "Current sensor fault detection, isolation, and reconfiguration for doubly fed induction generators," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 4239–4245, Oct. 2009.
[21] R. Jayabalan and B. Fahimi, "Monitoring and fault diagnosis of multiconverter systems in hybrid electric vehicles," *IEEE Trans. Veh. Technol.*, vol. 55, no. 5, pp. 1475–1484, Sep. 2006.
[22] R. Merzouki, M. A. Djeziri, and B. Ould-Bouamama, "Intelligent monitoring of electric vehicle," in *Proc. IEEE/ASME Int. Conf. AIM*, Jul. 2009, pp. 797–804.
[23] S. Grammatico, A. Balluchi, and E. Cosoli, "A series-parallel hybrid electric powertrain for industrial vehicles," in *Proc. IEEE Veh. Power Propuls. Conf. (VPPC)*, Sep. 2010, pp. 1–6.
[24] M. Ehsani, Y. Gao, and A. Emad, *Modern Electric, Hybrid Electric, and Fuel Cell Vehicles: Fundamentals, Theory, and Design*. Boca Raton, FL, USA: CRC Press, 2010.
[25] Argonne Nat. Lab. *Powertrain System Analysis Toolkit*. [Online]. Available: http://www.transportation.anl.gov/modeling_simulation/PSAT/index.html, accessed Jul. 12, 2014.
[26] *Development of Distributed Systems and ECU Test—Datasheet for CANoe*. [Online]. Available: http://www.vector.com, accessed Jul. 12, 2014.
[27] K. Chaaban and P. Leserf, "Simulation of a steer-by-wire system using flexray-based ECU network," in *Proc. Int. Conf. Adv. Comput. Tools Eng. Appl.*, Jul. 2009, pp. 21–26.
[28] S. Singh *et al.*, "Dynamic multiple fault diagnosis: Mathematical formulations and solution techniques," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 1, pp. 160–176, Jan. 2009.
[29] [Online]. Available: http://www.epa.gov/nvfel/testing/dynamometer.htm, accessed Jul. 12, 2014.
[30] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 121–167, 1998.
[31] M. Basseville and A. Benveniste, *Detection of Abrupt Changes in Signals and Dynamical Systems*. Berlin, Germany: Springer-Verlag, 1986.
[32] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 1994.
[33] Z. Ghahramani and M. I. Jordan, "Factorial hidden Markov models," in *Machine Learning*. Boston, MA, USA: Kluwer, 1997.
[34] I. K. Fodor. *A Survey of Dimension Reduction Techniques*. [Online]. Available: http://www.llnl.gov/CASC/sapphire/pubs/148494.pdf
[35] R. E. Bellman, *Adaptive Control Processes*. Princeton, NJ, USA: Princeton Univ. Press, 1961.
[36] M. Azam, K. Pattipati, J. Allanach, S. Poll, and A. Patterson-Hine, "In-flight fault detection and isolation in aircraft flight control systems," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Mar. 2005, pp. 3555–3565.
[37] M. S. Azam, F. Tu, K. R. Pattipati, and R. Karanam, "A dependency model-based approach for identifying and evaluating power quality problems," *IEEE Trans. Power Del.*, vol. 19, no. 3, pp. 1154–1166, Jul. 2004.
[38] J. A. Crossman, H. Guo, Y. L. Murphy, and J. Cardillo, "Automotive signal fault diagnostics—Part I: Signal fault analysis, signal segmentation, feature extraction and quasi-optimal feature selection," *IEEE Trans. Veh. Technol.*, vol. 52, no. 4, pp. 1063–1075, Jul. 2003.
[39] K. Choi, J. Luo, K. R. Pattipati, S. M. Namburu, L. Qiao, and S. Chigusa, "Data reduction techniques for intelligent fault diagnosis in automotive systems," in *Proc. IEEE Autotestcon*, Anaheim, CA, USA, Sep. 2006, pp. 66–72.
[40] K. Choi, S. M. Namburu, M. S. Azam, J. Luo, K. R. Pattipati, and A. Patterson-Hine, "Fault diagnosis in HVAC chillers," *IEEE Instrum. Meas. Mag.*, vol. 8, no. 3, pp. 24–32, Aug. 2005.
[41] P. Nomikos, "Detection and diagnosis of abnormal batch operations based on multi-way principal component analysis," *ISA Trans.*, vol. 35, no. 3, pp. 259–266, 1996.
[42] J. Chen and R. J. Patton, *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Norwell, MA, USA: Kulwer, 1999.
[43] C. Sankavaram, B. Pattipati, K. Pattipati, Y. Zhang, M. Howell, and M. Salman, "Data-driven fault diagnosis in a hybrid electric vehicle regenerative braking system," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Mar. 2012, pp. 1–11.

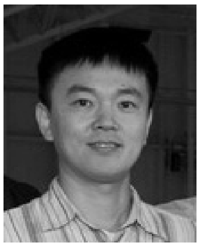
CHAITANYA SANKAVARAM (M'11) has been a Researcher with the Vehicle Systems Research Laboratory, General Motors Global Research and Development Center, Warren, MI, USA, since 2013. Her work involves research and development of diagnostics, prognostics, and health management solutions for automotive systems. She was a Project Engineer with Wipro Technologies, Bangalore, India. She received the M.S. degree from the University of Connecticut, Storrs, CT, USA, where she is currently pursuing the Ph.D. degree in electrical and computer engineering. She received the B.Tech. degree in electrical and electronics engineering from Sri Venkateswara University, Tirupathi, India, in 2005. Her research interests include fault diagnosis and prognosis, machine learning, data mining, pattern recognition, reliability analysis, and optimization theory.

BHARATH PATTIPATI (M'12) received the B.E. degree in electrical and electronics engineering from the M. S. Ramaiah Institute of Technology, Bangalore, India, in 2005, and the master's degree in electrical and computer engineering from the University of Connecticut, Storrs, CT, USA, in 2009, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. His current research interests include vehicle health management, fault diagnosis, prognosis and condition-based maintenance, battery management systems, and application of systems, optimization, estimation, and control techniques to complex large-scale systems, neural networks, and pattern recognition.



KRISHNA R. PATTIPATI (S'77–M'80–SM'91–F'95) received the B.Tech. (Hons.) degree in electrical engineering from IIT Kharagpur, Kharagpur, India, in 1975, and the M.S. and Ph.D. degrees in systems engineering from the University of Connecticut (UCONN), Storrs, CT, USA, in 1977 and 1980, respectively. He was with Alphatech, Inc., Burlington, MA, USA, from 1980 to 1986. He has been with the Department of Electrical and Computer Engineering, UCONN, where he is currently the UTC Professor of Systems Engineering and serves as the Interim Director of the UTC Institute for Advanced Systems Engineering. His current research activities are in the areas of agile planning, diagnosis and prognosis techniques for cyber-physical systems, multiobject tracking, and combinatorial optimization. A common theme among these applications is that they are characterized by a great deal of uncertainty, complexity, and computational intractability. He is the Co-Founder of Qualtech Systems, Inc., East Hartford, CT, USA, a firm specializing in advanced integrated diagnostics software tools (TEAMS, TEAMS-RT, TEAMS-RDS, TEAMATE), and serves on the board of Aptima, Inc.

Dr. Pattipati was selected by the IEEE Systems, Man, and Cybernetics (SMC) Society as the Outstanding Young Engineer in 1984, and received the Centennial Key to the Future Award. He served as the Editor-in-Chief of the IEEE Transactions on Systems, Man, and Cybernetics–Part B from 1998 to 2001, the Vice President for Technical Activities of the IEEE SMC Society from 1998 to 1999, and the Vice President for Conferences and Meetings of the IEEE SMC Society from 2000 to 2001. He was a co-recipient of the 1999 Andrew P. Sage Award for the Best SMC Transactions Paper, the 2000 Barry Carlton Award for the Best AES Transactions Paper, the 2002 and 2008 NASA Space Act Awards for *A Comprehensive Toolset for Model-Based Health Monitoring and Diagnosis*, and *Real-Time Update of Fault-Test Dependencies of Dynamic Systems: A Comprehensive Toolset for Model-Based Health Monitoring and Diagnostics*, and the 2003 AAUP Research Excellence Award from UCONN. He also received the Best Technical Paper Awards at the 1985, 1990, 1994, 2002, 2004, 2005, and 2011 IEEE AUTOTEST Conferences, and the 1997 and 2004 Command and Control Conferences. He is an elected fellow of the Connecticut Academy of Science and Engineering.



YILU ZHANG (M'02–SM'08) received the B.S. and M.S. degrees in electrical engineering from Zhejiang University, Hangzhou, China, in 1994 and 1997, respectively, and the Ph.D. degree in computer science from Michigan State University, East Lansing, MI, USA, in 2002. He joined the General Motors Global Research and Development Center, Warren, MI, USA, in 2002, as a Senior Researcher, where he currently manages the Vehicle Health Management Group. His research interests include statistical pattern recognition, machine learning, signal processing, and their applications, including integrated vehicle health management and human–machine interactions. He served as an Associate Editor of the *International Journal of Humanoid Robotics* from 2003 to 2007, the Publication Chair of the IEEE 8th International Conference on Development and Learning in 2009, and the Chair of the Battery Management System Workshop in conjunction with the PHM Society Annual Conference in 2011. He has authored over 50 refereed technical papers. He holds 20 U.S. patents and over 20 pending patent applications, of which most are in the area of vehicle diagnosis and prognosis. He was a recipient of the Boss Kettering Award (2008 and 2010), the highest technology award in General Motors, for his contribution in vehicle diagnostics technologies.

MARK HOWELL (M'02–SM'14) received the B.Sc. degree in cybernetics from the University of Reading, Reading, U.K., in 1987, and the M.Sc. and Ph.D. degrees in control engineering from the University of Sheffield, Sheffield, U.K., in 1995. From 1995 to 2003, he was a Post-Doctoral Research Fellow of Automotive Engineering at Loughborough University, Loughborough, U.K. He was a Senior Researcher with General Motors, Detroit, MI, USA, from 2003 to 2014, where he was involved in developing control, prognostic, and vehicle health management solutions for the automotive industry. He is currently a Control Systems Research Engineer with Trimble Navigation, Christchurch, New Zealand, developing autonomous machine control systems. He has authored over 30 papers and holds 15 U.S. patents.