An Integrated Diagnostic Process for Automotive Systems

Krishna Pattipati[1], Anuradha Kodali[1], Jianhui Luo[3], Kihoon Choi[1], Satnam Singh[1], Chaitanya Sankavaram[1], Suvasri Mandal[1], William Donat[1], Setu Madhavi Namburu[2], Shunsuke Chigusa[2], and Liu Qiao[2]

[1] University of Connecticut, Storrs, CT 06268, USA, krishna@engr.uconn.edu
[2] Toyota Technical Center USA, 1555 Woodridge Rd., Ann Arbor, MI 48105, USA
[3] Qualtech Systems, Inc., Putnam Park, Suite 603, 100 Great Meadow Road, Wethersfield, CT 06109, USA

1 Introduction

The increased complexity and integration of vehicle systems has resulted in greater difficulty in the identification of malfunction phenomena, especially those related to cross-subsystem failure propagation and thus made system monitoring an inevitable component of future vehicles. Consequently, a continuous monitoring and early warning capability that detects, isolates and estimates size or severity of faults (viz., fault detection and diagnosis), and that relates detected degradations in vehicles to accurate remaining life-time predictions (viz., prognosis) is required to minimize downtime, improve resource management via condition-based maintenance, and minimize operational costs.

The recent advances in sensor technology, remote communication and computational capabilities, and standardized hardware/software interfaces are creating a dramatic shift in the way the health of vehicle systems is monitored and managed. The availability of data (sensor, command, activity and error code logs) collected during nominal and faulty conditions, coupled with intelligent health management techniques, ensure continuous vehicle operation by recognizing anomalies in vehicle behavior, isolating their root causes, and assisting vehicle operators and maintenance personnel in executing appropriate remedial actions to remove the effects of abnormal behavior. There is also an increased trend towards online real-time diagnostic algorithms embedded in the Electronic Control Units (ECUs), with the diagnostic troubleshooting codes (DTCs) that are more elaborate in reducing cross-subsystem fault ambiguities. With the advancements in remote support, the maintenance technician can use an intelligent scanner with optimized and adaptive state-dependent test procedures (e.g., test procedures generated by test sequencing software, e.g., [47]) instead of pre-computed static paper-based decision trees, and detailed maintenance logs ("cases") with diagnostic tests performed, their outcomes, test setups, test times and repair actions can be recorded automatically for adaptive diagnostic knowledge management. If the technician can not isolate the root cause, the history of sensor data and symptoms are transmitted to a technical support center for further refined diagnosis.

The automotive industry has adopted quantitative simulation as a vital tool for a variety of functions, including algorithm design for ECUs, rapid prototyping, programming for hardware-in-the-loop simulations (HILS), production code generation, and process management documentation. Accordingly, fault detection and diagnosis (FDD) and prognosis have mainly evolved upon three major paradigms, viz., model-based, data-driven and knowledge-based approaches.

The model-based approach uses a mathematical representation of the system. 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. Most applications of model-based diagnostic approach have been on systems with a relatively small number of inputs, outputs, and states. 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.

K. Pattipati et al.: *An Integrated Diagnostic Process for Automotive Systems*, Studies in Computational Intelligence (SCI) **132**, 191–218 (2008)
www.springerlink.com
© Springer-Verlag Berlin Heidelberg 2008

A data-driven approach to FDD is preferred when system models are not available, but instead system monitoring data is available. This situation arises frequently when subsystem vendors seek to protect their intellectual property by not providing internal system details to the system or vehicle integrators. 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 FDD. Neural network and statistical classification methods are illustrative of data-driven techniques. Significant amount of data is needed from monitored variables under nominal and faulty scenarios for data-driven analysis.

The knowledge-based approach uses qualitative models for process monitoring and troubleshooting. The approach is especially well-suited for systems for which detailed mathematical models are not available. Most knowledge-based techniques are based on casual analysis, expert systems, and/or ad hoc rules. Because of the qualitative nature of these models, knowledge-based approaches have been applied to many complex systems. Graphical models such as Petri nets, multi-signal flow graphs and Bayesian networks are applied for diagnostic knowledge representation and inference in automotive systems [34]. Bayesian Networks subsume the deterministic fault diagnosis models embodied in the Petri net and multi-signal models. However, multi-signal models are preferred because they can be applied to large-scale systems with thousands of failure sources and tests, and can include failure probabilities and unreliable tests as part of the inference process in a way which is computationally more efficient than Bayesian networks. Model based, data-driven and knowledge-based approaches provide the "sand box" that test designers can use to experiment with, and systematically select relevant models or combinations thereof to satisfy the requirements on diagnostic accuracy, computational speed, memory, on-line versus off-line diagnosis, and so on. Ironically, no single technique alone can serve as the diagnostic approach for complex automotive applications. Thus, an integrated diagnostic process [41] that naturally employs data-driven techniques, graph-based dependency models and mathematical/physical models is necessary for fault diagnosis, thereby enabling efficient maintenance of these systems.

Integrated diagnostics represents a structured, systems engineering approach and the concomitant information-based architecture for maximizing the economic and functional performance of a system by integrating the individual diagnostic elements of design for testability, on-board diagnostics, automatic testing, manual troubleshooting, training, maintenance aiding, technical information, and adaptation/learning [4, 29]. This process, illustrated in Fig. 1, is employed during all stages of a system life cycle, viz., concept, design, development, production, operations, and training. From a design perspective, it has been well-established that a system must be engineered simultaneously with three design goals in mind: performance, ease of maintenance, and reliability [12]. To maximize its impact, these design goals must be considered at all stages of the design: concept to design of subsystems to system integration. Ease of maintenance and reliability are improved by performing testability and reliability analyses at the design stage.

The integrated diagnostic process we advocate contains six major steps: *model, sense, develop and update test procedures, infer, adaptive learning*, and *predict*.

(A) Step 1: Model

In this step, models to understand fault-to-error characteristics of system components are developed. This is achieved by a hybrid modeling technique, which combines mathematical models (simulation models), monitored data and graphical cause-effect model (e.g., diagnostic matrix (D-matrix) [34]) in the failure space, through an understanding of the failure modes and their effects, physical/behavioral models, and statistical and machine learning techniques based on actual failure progression data (e.g., field failure data). The testability analysis tool (e.g., TEAMS [47]) computes percent fault detection and isolation measures, identifies redundant tests and ambiguity groups, and generates updated Failure Modes Effects and Criticality Analysis (FMECA) report [13], and the diagnostic tree [11]. The onboard diagnostic data can also be downloaded to a remote diagnostic server (such as TEAMS-RDS [47]) for interactive diagnosis (by driving interactive electronic technical manuals), diagnostic/maintenance data management, logging and trending. The process can also be integrated with the supply-chain management systems and logistics databases for enterprise-wide vehicle health and asset management.

(B) Step 2: Sense

The sensor suite is typically designed for vehicle control and performance. In this step, the efficacies of these sensors are systematically evaluated and quantified to ensure that adequate diagnosis and prognosis

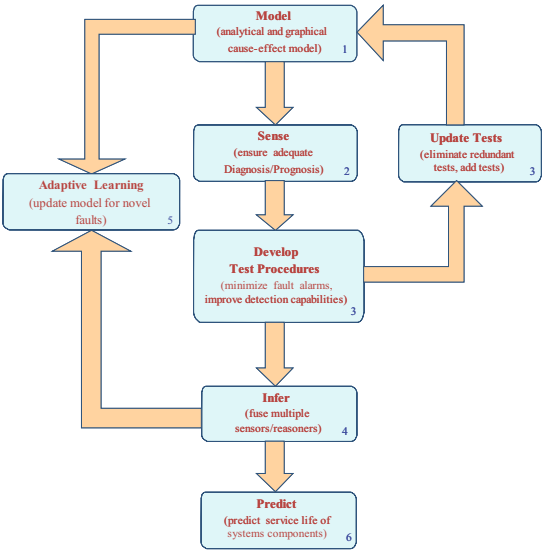

Fig. 1. Integrated diagnostic process

are achievable. If the existing sensors are not adequate for diagnosis/prognosis, use of additional sensors and/or analytical redundancy must be considered without impacting vehicle control and performance. Diagnostic analysis by analysis tools (such as TEAMS [47]) can be used to compare and evaluate alternative sensor placement schemes.

(C) Step 3: Develop and Update Test Procedures
Smart test procedures that detect failures, or onsets thereof, have to be developed. These procedures have to be carefully tuned to minimize false alarms, while improving their detection capability (power of the test and detection delays). The procedures should have the capability to detect trends and degradation, and assess the severity of a failure for early warning.

(D) Step 4: Adaptive Learning
If the observed fault signature does not correspond to faults reflected in the graphical dependency model derived from fault simulation, system identification techniques are invoked to identify new cause-effect relationships to update the model.

(E) Step 5: Infer
An integrated on-board and off-board reasoning system capable of fusing results from multiple sensors/reasoners and driver (or "driver model") to evaluate the health of the vehicle needs to be applied. This reasoning engine and the test procedures have to be compact enough so that they can be embedded in the ECU and/or a diagnostic maintenance computer for real-time maintenance. If on-board diagnostic data is downloaded to a repair station, remote diagnostics is used to provide assistance to repair personnel in rapidly identifying replaceable component(s).

(F) Step 6: Predict (Prognostics)
Algorithms for computing the remaining useful life (RUL) of vehicle components that interface with onboard usage monitoring systems, parts management and supply chain management databases are needed. Model-based prognostic techniques based on singular perturbation methods of control theory, coupled with an interacting multiple model (IMM) estimator [1], provide a systematic method to predict the RUL of system components.

This development process provides a general framework for diagnostic design and implementation for automotive applications. The applications of this process are system specific, and one need not go through all the steps for every system. In this chapter, we focus on fault diagnosis of automotive systems using model-based and data-driven approaches. The above integrated diagnostic process has been successfully applied to automotive diagnosis, including an engine's air intake subsystem (AIS) [35] using model-based techniques and an anti-lock braking system (ABS) [36] using both model-based and data-driven techniques. Data-driven techniques are employed for fault diagnosis on automobile engine data [9, 10, 38]. The prognostic process is employed to predict the remaining life of an automotive suspension system [37].

2 Model-Based Diagnostic Approach

2.1 Model-Based Diagnostic Techniques

A key assumption of quantitative model-based techniques is that a mathematical model is available to describe the system. Although this approach is complex and needs more computing power, several advantages make it very attractive. The mathematical models are used to estimate the needed variables for analytical (software) redundancy. With the mathematical model, a properly designed detection and diagnostic scheme can be not only robust to unknown system disturbances and noise, but also can estimate the fault size at an early stage. The major techniques for quantitative model-based diagnostic design include parameter estimation, observer-based design and/or parity relations [43, 54].

Parity (Residual) Equations

Parity relations are rearranged forms of the input-output or state-space models of the system [26]. The essential characteristic of this approach is to check for consistency of the inputs and outputs. Under normal operating conditions, the magnitudes of residuals or the values of parity relations are small. To enhance residual-based fault isolation, directional, diagonal and structured residual design schemes are proposed [22]. In the directional residual scheme, the response to each fault is confined to a straight line in the residual space. Directional residuals support fault isolation, if the response directions are independent. In the diagonal scheme, each element of the residual vector responds to only one fault. Diagonal residuals are ideal for the isolation of multiple faults, but they can only handle m faults, where m equals the number of outputs [21]. Structured residuals are designed to respond to different subsets of faults and are insensitive to others not in each subset. Parity equations require less computational effort, but do not provide as much insight into the process as parameter estimation schemes.

Parameter Identification Approach

The parameter estimation-based method [24, 25] not only detects and isolates a fault, but also may estimate its size. A key requirement of this method is that the mathematical model should be identified and validated so that it expresses the physical laws of the system as accurately as possible. If the nominal parameters are not known precisely, they need to be estimated from observed data. Two different parameter identification approaches exist for this purpose.

Equation Error Method. The parameter estimation approach not only detects and isolates a fault, but also estimate its size, thereby providing FDD as a one-shot process. Equation error methods use the fact that faults in dynamic systems are reflected in the physical parameters, such as the friction, mass, inertia, resistance and so on. Isermann [25] has presented a five-step parameter estimation method for general systems.

(1) Obtain a nominal model of the system relating the measured input and output variables:

$$\underline{y}(t) = f\{\underline{u}(t), \underline{\theta}_0\} \tag{1}$$

(2) Determine the relationship function $\underline{g}$ between the model parameters $\underline{\theta}$, where underscore notation of the parameters represents a vector, and the physical system coefficients $\underline{p}$:

$$\underline{\theta} = \underline{g}(\underline{p}) \tag{2}$$

(3) Identify the model parameter vector $\underline{\theta}$ from the measured input and output variables

$$\underline{U}^N = \{\underline{u}(k) : 0 \le k \le N\} \text{ and } \underline{Y}^N = \{\underline{y}(k) : 0 \le k \le N\} \tag{3}$$

(4) Calculate the system coefficients (parameters): $\underline{p} = g^{-1}(\underline{\theta})$ and deviations from nominal coefficients, $\underline{p}_0 = \underline{g}^{-1}(\underline{\theta}_0)$, viz., $\Delta\underline{p} = \underline{p} - \underline{p}_0$

(5) Diagnose faults by using the relationship between system faults (e.g., short-circuit, open-circuit, performance degradations) and deviations in the coefficients $\Delta\underline{p}$.

Output Error (Prediction-Error) Method. For a multiple input-multiple output (MIMO) system, suppose we have collected a batch of data from the system:

$$\underline{Z}^N = [\underline{u}(1), \underline{y}(1), \underline{u}(2), \underline{y}(2), \ldots, \underline{u}(N), \underline{y}(N)] \tag{4}$$

Let the output error provided by a certain model parameterized by $\underline{\theta}$ be given by

$$\underline{e}(k, \underline{\theta}) = \underline{y}(k) - \hat{\underline{y}}(k|\underline{\theta}) \tag{5}$$

Let the output-error sequence in (5) be filtered through a stable filter L and let the filtered output be denoted by $\underline{e}_F(k, \underline{\theta})$. The estimate $\hat{\underline{\theta}}_N$ is then computed by solving the following optimization problem:

$$\hat{\underline{\theta}}_N = \arg\min_{\underline{\theta}} V_N(\underline{\theta}, \underline{Z}^N) \tag{6}$$

where

$$V_N(\underline{\theta}, \underline{Z}^N) = \frac{1}{N}\sum_{k=1} \underline{e}_F^T(k, \underline{\theta})\Sigma^{-1}\underline{e}_F(k, \underline{\theta}) \tag{7}$$

Here Σ is the covariance of error vector. The effect of filter L is akin to frequency weighting [32]. For example, a low-pass filter can suppress high-frequency disturbances. The minimization of (7) is carried out iteratively. The estimated covariance matrix and the updated parameter estimates at iteration i are

$$\begin{aligned} \hat{\Sigma}_N^{(i)} &= \tfrac{1}{N-1}\sum_{k=1}^{N} \underline{e}_F(k, \underline{\theta}_N^{(i)})\underline{e}_F^T(k, \underline{\theta}_N^{(i)}) \\ \hat{\underline{\theta}}_N^{(i+1)} &= \arg\min_{\underline{\theta}} \tfrac{1}{N}\sum_{k=1}^{N} \underline{e}_F^T(k, \underline{\theta})[\hat{\Sigma}_N^{(i)}]^{-1}\underline{e}_F(k, \underline{\theta}) \end{aligned} \tag{8}$$

We can also derive a recursive version for the output-error method. In general, the function $V_N(\underline{\theta}, \underline{Z}^N)$ cannot be minimized by analytical methods; the solution is obtained numerically. The computational effort of this method is substantially higher than the equation error method, and, consequently, on-line real-time implementation may not be achievable.

Observers

The basic idea here is to estimate the states of the system from measured variables. The output estimation error is therefore used as a residual to detect and, possibly, isolate faults. Some examples of the observers are Luenberger observer [52], Kalman filters and Interacting Multiple Models [1], output observers [43, 54], nonlinear observers [20, 53], to name a few.

In order to introduce the structure of a (generalized) observer, consider a discrete-time, time-invariant, linear dynamic model for the process under consideration in state-space form as follows.

$$\begin{aligned} &\underline{x}(t+1) = A\underline{x}(t) + B\underline{u}(t) \\ &\underline{y}(t) = C\underline{x}(t) \\ &\text{where } \underline{u}(t) \in \Re^r,\ \underline{x}(t) \in \Re^n \text{ and } \underline{y}(t) \in \Re^m \end{aligned} \tag{9}$$

Assuming that the system matrices A, B and C are known, an observer is used to reconstruct the system variables based on the measured inputs and outputs $\underline{u}(t)$ and $\underline{y}(t)$:

$$\begin{aligned} \hat{\underline{x}}(t+1) &= A\hat{\underline{x}}(t) + B\underline{u}(t) + H\underline{r}(t) \\ \underline{r}(t) &= \underline{y}(t) - C\hat{\underline{x}}(t) \end{aligned} \tag{10}$$

For the state estimation error $\underline{e}_x(t)$, it follows from (10) that

$$\begin{aligned} \underline{e}_x(t) &= \underline{x}(t) - \hat{\underline{x}}(t) \\ \underline{e}_x(t+1) &= (A - HC)\underline{e}_x(t) \end{aligned} \tag{11}$$

The state estimation error $\underline{e}_x(t)$, and the residual $\underline{r}(t) = \mathrm{C}\underline{e}_x(t)$ vanish asymptotically

$$\lim_{t\to\infty} \underline{e}_x(t) = \underline{0} \tag{12}$$

if the observer is stable; this can be achieved by proper design of the observer feedback gain matrix H (provided that the system is detectable). If the process is subjected to parametric faults, such as changes in parameters in $\{A,\ B\}$, the process behavior becomes

$$\begin{aligned} \underline{x}(t+1) &= (A + \Delta A)\underline{x}(t) + (B + \Delta B)\underline{u}(t) \\ \underline{y}(t) &= C\underline{x}(t) \end{aligned} \tag{13}$$

Then, the state error $\underline{e}_x(t)$, and the residual $\underline{r}(t)$ are given by

$$\begin{aligned} \underline{e}_x(t+1) &= (A - HC)\underline{e}_x(t) + \Delta A\underline{x}(t) + \Delta B\underline{u}(t) \\ \underline{r}(t) &= C\underline{e}_x(t) \end{aligned} \tag{14}$$

In this case, the changes in residuals depend on the parameter changes, as well as input and state variable changes. The faults are detected and isolated by designing statistical tests on the residuals.

2.2 Application of Model-Based Diagnostics to an Air-Intake System

Experimental Set-Up: HILS Development Platform

The hardware for the development platform consists of a custom-built ComputeR Aided Multi-Analysis System (CRAMAS) and two Rapid Prototype ECUs (Rtypes) [19]. The CRAMAS (Fig. 2) is a real-time simulator that enables designers to evaluate the functionality and reliability of their control algorithms installed in ECUs for vehicle sub-systems under simulated conditions, as if they were actually mounted on an automobile. The Rtype is an ECU emulator for experimental research on power train control that achieves extremely high-speed processing and high compatibility with the production ECU [23]. Besides emulating the commercial ECU software, experimental control designs can be carried out in the Rtype host PC using the MATLAB/Simulink environment and compiled through the Real-Time Workshop. Typical model-based techniques include digital filter design to suppress the noise, abrupt change detection techniques (such as the generalized likelihood ratio test (GLRT), cumulative sum (CUSUM), sequential probability ratio test (SPRT)), recursive least squares (RLS) estimation, and output error (nonlinear) estimation for parametric faults, extended Kalman filter (EKF) for parameter and state estimation, Luenberger observer, and the diagnostic inference algorithms (e.g., TEAMS-RT) [2, 45, 47]. This toolset facilitates validation of model-based diagnostic algorithms.

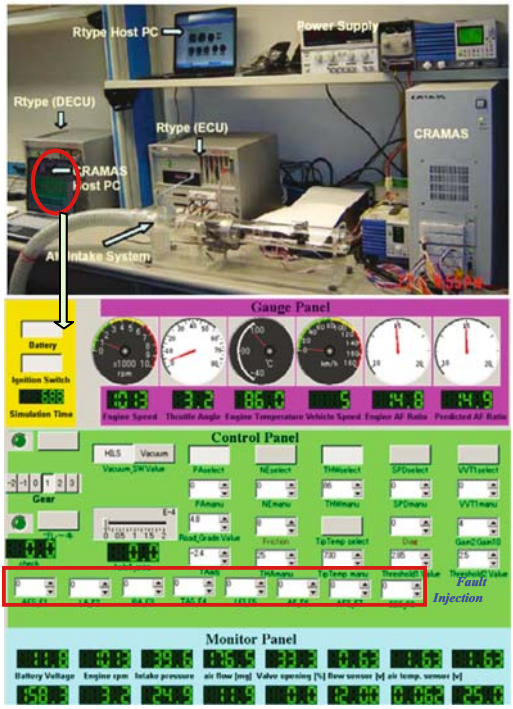

Fig. 2. CRAMAS® engine simulation platform and operation GUI

Combining the Rtype with the CRAMAS, and a HIL Simulator, designers can experiment with different diagnostic techniques, and/or verify their own test designs/diagnostic inference algorithms, execute simulations, and verify HILS operations. After rough calibration is confirmed, the two Rtypes can also be installed in an actual vehicle, and test drives can be carried out [23]. As a result, it is possible to create high-quality diagnostic algorithms at the initial design stage, thereby significantly shortening the development period ("time-to-market").

The diagnostic experiment employs a prototype air intake subsystem (AIS) as the hardware system in our HILS. The function of AIS is to filter the air, measure the intake air flow, and control the amount of air entering the engine. The reasons for selecting the AIS are its portability and its reasonably accurate physical model. Figure 3 shows the photograph of our prototype AIS. It consists of a polyvinyl chloride pipe, an air flow sensor, an electronic throttle, and a vacuum pump. It functionally resembles the real AIS for the engine.

The model consists of five primary subsystems: air dynamics, fuel dynamics, torque generation, rotational dynamics, and the exhaust system. We used a mean value model, which captures dynamics on a time-scale spanning over several combustion cycles (without considering in-cycle effects). In the following, we elaborate on the sub-system models. The details of the subsystems and SIMULINK model of air-intake system are available in [35]. Nine faults are considered for this experiment. The air flow sensor fault (F1) is injected by adding 6% of the original sensor measurement. Two physical faults, a leak in the manifold (F2) and a dirty air filter (F3), can be manually injected in the prototype AIS. The leakage fault is injected by adjusting the hole size in the pipe, while the dirty air filter fault is injected by blocking the opening of the pipe located at the right hand side of Fig. 3. The throttle angle sensor fault (F4) is injected by adding 10% of

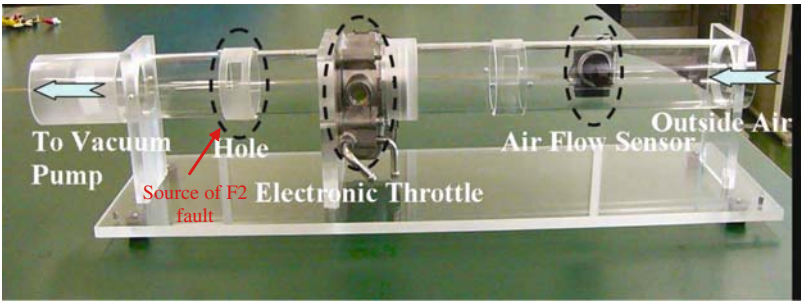

Fig. 3. Photograph of air-intake system

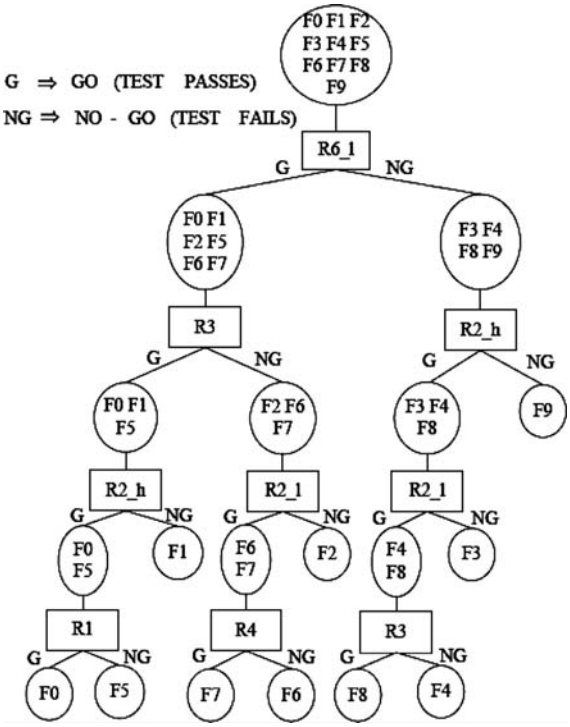

Fig. 4. Test sequence generation for the engine system

the original sensor measurement. Throttle actuator fault (F5) is injected by adding a pulse to the output of the throttle controller [40]. The pulse lasts for a duration of 3 s and the pulse amplitude is 20% of the nominal control signal amplitude. The other faults are modeled using a realistic engine model in CRAMAS, and are injected via the GUI in CRAMAS host PC. Faults F6–F9 injected through the CRAMAS include:

Table 1. Fault list of engine system

F1[a]	Air flow sensor fault (+6%)
F2[a]	Leakage in AIS (2 cm × 1 cm)
F3[a]	Blockage of air filter
F4[a]	Throttle angle sensor fault (+10%)
F5[a]	Throttle actuator fault (+20%)
F6[b]	Less fuel injection (−10%)
F7[b]	Added engine friction (+10%)
F8[b]	AF sensor fault (−10%)
F9[b]	Engine speed sensor fault (−10%)

[a] Real faults in the AIS
[b] Simulated faults in the CRAMAS engine model

Table 2. Diagnostic matrix of the engine system

Fault\test	R1	R2_h	R2_l	R3	R4	R5	R6_h	R6_l
F0	0	0	0	0	0	0	0	0
F1	0	1	0	0	0	0	0	0
F2	0	0	1	1	0	0	1	0
F3	0	0	1	1	0	0	0	1
F4	0	0	0	1	0	0	0	1
F5	1	0	0	0	0	0	0	0
F6	0	0	0	1	1	1	1	0
F7	0	0	0	1	0	0	0	0
F8	0	0	0	0	0	0	0	1
F9	0	1	0	1	0	0	0	1

less fuel injection (−10%), added engine friction (+10%), air/fuel sensor fault (−10%), and the engine speed sensor fault (−10%). Here all the % changes are deviations from the nominal values. All the injected faults are listed in Table 1.

Residuals for the Engine Model

The identified subsystem reference models are used to set up six residuals based on the dynamics of the diagnostic model as follows. The first residual $R1$ is based on the difference between the throttle angle sensor reading from the AIS and the predicted throttle angle. The second residual $R2$ is based on the difference between air mass flow sensor reading from the AIS and the predicted air flow past the throttle. The third residual $R3$ is based on the difference between the engine speed from CRAMAS, and the predicted engine speed. The fourth residual $R4$ is based on the difference between the turbine speed from CRAMAS and predicted turbine speed. The fifth residual $R5$ is based on the difference between the vehicle speed from CRAMAS, and the predicted vehicle speed. The sixth residual $R6$ is obtained as the difference between the air/fuel ratio from CRAMAS, and the predicted air/fuel ratio.

Experimental Results

Table 2 shows the diagnostic matrix (D-matrix), which summarizes the test designs for all faults considered in the engine. Each row represents a fault state and columns represent tests. The D-matrix $D = \{d_{ij}\}$ provides detection information, where d_{ij} is 1 if test j detects a fault state i. Here, F0 represents "System OK" status, with all the tests passing. Since there are no identical rows in this table, all the faults can be uniquely isolated. For *R2*, there are two tests: *R2_l* and *R2_h*, which correspond to the threshold tests for

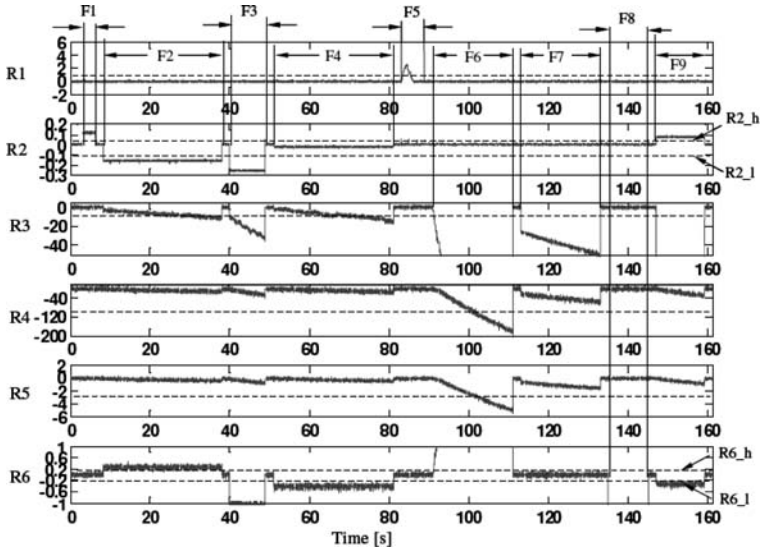

Fig. 5. Behaviors of the residuals with different faults injected at different times

low (negative) and high (positive) levels. There are also two tests *R6_l* and *R6_h* for *R6* corresponding to low and high levels of the threshold. The other tests have the same name as the residual name (e.g., *R1* is the test for residual *R1*). The goal of using minimum expected cost tests to isolate the faults is achieved by constructing an AND/OR diagnostic tree via an optimal test sequencing algorithm [42, 48, 49]; the tree is shown in Fig. 4, where an AND node represents a test and an OR node denotes the ambiguity group. In this tree, the branch which goes to the left/right below the test means the test passed (G)/failed (NG). It can be seen that tests *R5* and *R6_h* are not shown in the tree, which means that these tests are redundant. This is consistent with the D-matrix, since *R4* and *R5* have identical columns. One feature of the diagnostic tree is that it shows the set of Go-path tests (*R6_l, R3, R2_h* and *R1*) that can respond to any fault. The Go-path tests can be obtained by putting all the tests on the left branches of the tree, which lead to the "system OK" status. Any residual exceeding the threshold(s) will result in a test outcome of 1 (failed). The residuals for throttle position, air mass flow, engine speed, turbine speed, vehicle speed, and air/fuel ratio show the expected behavior. In the fault-free case, the residuals are almost zero. The results of real-time fault diagnosis are exemplified in Fig. 5 with different faults injected at different times for a particular operating condition of the engine. The noise level of residuals is very low and changes are abrupt under all faulty scenarios. Therefore, simple threshold tests would achieve 100% fault detection and 0% false alarms. Although more advanced tests (such as generalized likelihood ratio test) can be adopted using the fault diagnosis toolset, they provided the same results in this experiment. The diagnostic scheme under other operating conditions (different throttle angles, etc.) was also tested. We found that under nominal conditions, the residuals have the same behavior (near zero).

However, under faulty conditions, the deviation of residuals will change as the operating condition changes. Therefore, to obtain the best performance (minimal false alarm and maximum fault detection) of this diagnosis scheme under different operating conditions, a reasonable practice is to use a lookup table for the thresholds on residual tests according to different operating conditions. During the experiment, we found that, for two faults, the engine operating conditions are unstable for large-size faults. The first fault is the air flow sensor fault (F1); the maximum size of the fault is +6%. The second fault is the leakage in air intake manifold; the maximum size of this fault is 4 cm × 1 cm. These findings are beneficial feedback

to control engineers in understanding the limits of their design. If the diagnostic matrix does change under different operating conditions, we can use multi-mode diagnosis techniques to handle this situation [51].

2.3 Model-Based Prognostics

Conventional maintenance strategies, such as corrective and preventive maintenance, are not adequate to fulfill the needs of expensive and high availability transportation and industrial systems. A new strategy based on forecasting of system degradation through a prognostic process is required. The recent advances in model-based design technology facilitate the integration of model-based diagnosis and prognosis of systems, leading to condition-based maintenance to potentially increase the availability of systems. The advantage of model-based prognostics is that, in many situations, the changes in feature vector are closely related to model parameters [7]. Therefore, it can also establish a functional mapping between the drifting parameters and the selected prognostic features. Moreover, as understanding of the system degradation improves, the model can be adapted to increase its accuracy and to address subtle performance problems.

Prognostic methods use residuals as features. The premise is that the residuals are large in the presence of malfunctions, and small in the presence of normal disturbances, noise and modeling errors.

Statistical techniques are used to define the thresholds on residuals to detect the presence of faults.

The model-based prognostic process is illustrated in Fig. 6. This process consists of six steps for predicting the remaining life of a system. These are briefly described below.

(A) Step 1: Identify System Model

The degradation model of a system is considered to be of the following singular perturbation form:

$$\begin{aligned} \dot{\underline{x}} &= \underline{f}(\underline{x}, \lambda(\theta), \underline{u}) \\ \dot{\theta} &= \varepsilon\ g(\underline{x}, \theta) \\ \underline{y} &= C\underline{x} + D\underline{u} + \underline{v} \end{aligned} \tag{15}$$

Here $\underline{x} \in R^n$ is the set of state variables associated with the fast dynamic behavior of the system; θ is a scalar, slow dynamic variable related to system damage (degradation); $\underline{u} \in \mathrm{R}^r$ is the input vector; the drifting parameter λ is a function of θ; the rate constant $0 < \varepsilon << 1$ defines the time-scale separation between the fast system state dynamics and the slow drift due to damage [6]; $\underline{y} \in R^m$ is the output vector and $\underline{v} \in R^m$ is the measurement noise. Since ε is very small, (15) can be considered as a two-time scale system with a slowly drifting parameter.

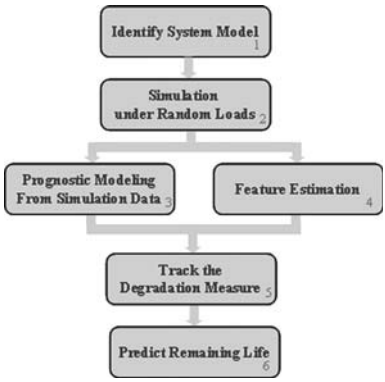

Fig. 6. Model-based prognostic process

The modified prognostic model, after scaling the damage variable $\theta(\theta_0 \leq \theta \leq \theta_M)$ to a normalized degradation measure $\xi(\xi_0 \leq \xi \leq 1)$ [58] and relating the degradation measure from load cycle $(i-1)$ to load cycle i is:

$$\begin{aligned} \dot{\underline{x}} &= \underline{f}(\underline{x}, \lambda(\xi), \underline{u}) \\ \xi_i &= \eta\phi_1(\xi_{i-1})\phi_2(\rho_i) + \xi_{i-1} \\ \underline{y} &= C\underline{x} + D\underline{u} + \underline{v} \end{aligned} \tag{16}$$

Here ρ_i is the random load parameter during cycle i (e.g., stress/strain amplitude). The function $\lambda(\xi)$, which maps the degradation measure to a system parameter, is often assumed to be a polynomial [58]:

$$\lambda(\xi) = \sum_{k=0}^{K} \alpha_k \xi^k \tag{17}$$

where K is the total number of stress levels.

(B) Step 2: Simulation Under Random Loads

The prognostic model is generally nonlinear. Consequently, the evolution of system dynamics (including fast and slow-time) is typically obtained through Monte-Carlo simulations. Since the parameter p is a stochastic process in many applications [57], simulation of (16) for ξ requires the update of degradation parameter ξ for every cycle based on the random load parameter p in that cycle, where p can be represented as a function of $\underline{x}$. The cycle number n_i and randomly realized load parameter $\left\{p_i^j\right\}_{j=1}^{n_i}$ can be obtained through cycle counting method, viz., the rainflow cycle. This method catches both slow and rapid variations of load by forming cycles that pair high maxima with low minima, even if they are separated by intermediate extremes [28].

Consequently, the updated degradation measure is obtained as

$$\xi_i = \eta\phi_1(\xi_{i-1}) \sum_{j=1}^{n_i} \phi_2(p_i^j) + \xi_{i-1} \tag{18}$$

The initial degradation measure ξ_0 is assumed to be a Gaussian random variable $N(\mu, \sigma^2)$, where μ and σ represent the mean and the standard deviation, respectively.

(C) Step 3: Prognostic Modeling

Prognostic modeling is concerned with the dynamics of the degradation measure. Consider a system excited under L different random load conditions as being in modes $1, 2, \ldots, L$. Assume that M Monte-Carlo simulations are performed for each random load condition. Then the L models can be constructed, one for each mode. The dynamic evolution of degradation measure under mode m is given by:

$$\xi_m(k+1) = \beta_m(\xi_m(k)) + v_m(k) \qquad k = 0, 1, \ldots \tag{19}$$

where m is the mode number, β_m is a function of previous state $\xi_m(k)$ and $v_m(k)$ is a zero mean white Gaussian noise with variance $\hat{Q}_m(k)$. The functional form of β_m is obtained from historical/simulated data. The state prediction (function β_m) is obtained in IMM [37] for mode m via:

$$\hat{\xi}_m(k+1/k) = \beta_m(\hat{\xi}_m(k/k)) \qquad k = 0, 1, \ldots. \tag{20}$$

Here $\hat{\xi}_m(k/k)$ is the updated (corrected) estimate of ξ_m at time k and $\hat{\xi}_m(k+1/k)$ denotes propagated (predicted) estimate of ξ_m at time $k+1$ based on the measurements up to and including time k.

(D) Step 4: Feature Parameter Estimation

Since the hidden variable ξ is unobserved, it needs to be estimated from the input/output data $\{\underline{u}(t), \underline{y}(t)\}_{t=0}^{T}$. One way to estimate ξ is to use the update equation for ξ in (16), where $\phi_2(\rho_i)$ is a function of measurement $\underline{y}$. Since the initial value of ξ_0 is not known, it will produce biased estimates.

Another method is based on estimation of the drifting parameters λ of the fast time process in (16). Two parameter estimation techniques, equation error method and output error method, can be

employed to estimate λ from a time history of measurements $\left\{\underline{u}(t), \underline{y}(t)\right\}_{t=0}^{T}$ [33]. Here, the equation error method is employed to estimate λ.

In the equation-error method, the governing equation for estimating λ is the residual equation. The residual equation $r(\underline{u}, \underline{y}, \lambda)$ is the rearranged form of the input-output or state-space model of the system. Suppose N data points of the fast-time process are acquired in an intermediate time interval $[t, t+\alpha T]$, then the optimal parameter estimate is given by:

$$\lambda^* = \arg\min_{\lambda^*} \sum_{i=1}^{N} \|r(u_i, y_i, \lambda)\|^2 \tag{21}$$

Based on the internal structure of the residual equation, two optimization algorithms – linear least squares and nonlinear least squares – can be implemented. If prior knowledge on the range of λ is available, the problem can be solved via constrained optimization. In any case, the measurement equation is constructed as:

$$z(k) = \lambda(\xi(k)) + \vartheta(k) \qquad k = 0, 1, 2, \ldots \tag{22}$$

where $z(k) = \lambda^*,\ \lambda(\xi)$ is typically a polynomial function as in (17) and $\vartheta(k)$ is a zero mean Gaussian noise with variance $\hat{S}(k)$. The variance is obtained as a by-product of the parameter estimation method.

(E) Step 5: Track the Degradation Measure

To track the degradation measure, an interacting multiple model (IMM) estimator [1, 46] is implemented for online estimation of the damage variable. For a system with L operational modes, there will be L models in the IMM, one for each mode. Each mode will have its own dynamic equation and the measurement equation is of the form in (22).

(F) Step 6: Predict the Remaining Life

The remaining life depends on the current damage state $\xi(k)$, as well as the future usage of the system. If the future operation of a system is known a priori, the remaining life can be estimated using the knowledge of future usage. Typically, one can consider three types of prior knowledge on future usage.

(1) Deterministic Operational Sequence: In this case, the system is assumed to be operated according to a known sequence of mode changes and mode durations. Define a sequence $S = \{m_i,\ T_{si},\ T_{ei}\}_{i=1}^{Q}$, where T_{si} and T_{ei} represent the start time and the end time under mode m_i, such that $T_{s1} = 0$, $T_{ei} = T_{si+1}$ and T_{sQ} is the time at which $\xi = 1$. Suppose M Monte-Carlo simulations are performed for this operational sequence. Then, the M remaining life estimates can be obtained based on $\hat{\xi}(k/k)$, the updated damage estimate at time instant k. The mean remaining life estimate and its variance are obtained from these M estimates.

(2) Probabilistic Operational Sequences: In this case, the system is assumed to operate under J operational sequences $S_j = \{m_i^j,\ T_{si}^j,\ T_{ei}^j\}_{j=1}^{J}$, where S_j is assumed to occur with a known probability p_j. If $\hat{r}_j(k)$ is the estimate of residual life based on sequence S_j, then the remaining life estimate $\hat{r}(k)$, and its variance $P(k)$, are given by:

$$\begin{aligned} \hat{r}(k) &= \sum_{j=1}^{L} p_j(k)\hat{r}_j(k) \\ P(k) &= \sum_{j=1}^{L} p_j(k)\left\{P_j(k) + [\hat{r}_j(k) - \hat{r}(k)]^2\right\} \end{aligned} \tag{23}$$

(3) On-Line Sequence Estimation: This method estimates the operational sequence based on measured data via IMM mode probabilities. Here, the future operation of the system is assumed to follow the observed history and the dynamics of mode changes. For the i^{th} Monte-Carlo run in mode m, the time to failure until $\xi = 1$ can be calculated as [61]:

$$t_m^i(end) = (\psi_m^i)^{-1} \tag{24}$$

The remaining life estimate from i^{th} Monte-Carlo run for mode m is:

$$\hat{r}_m^i(k) = t_m^i(end) - t_m^i(k) \tag{25}$$

Then, the remaining life estimate and its variance for mode m are:

$$\begin{aligned} \hat{r}_m(k) &= \tfrac{1}{M} \sum_{i=1}^{M} \hat{r}_m^i(k) \\ P_m(k) &= \tfrac{1}{M-1} \sum_{i=1}^{M} [\hat{r}_m(k) - \hat{r}_m^i(k)]^2 \end{aligned} \tag{26}$$

The above calculation can be performed off-line based on simulated (or historical) data. To reflect the operational history, the mode probabilities from IMM can be used to estimate the remaining life.

2.4 Prognostics of Suspension System

To demonstrate the prognostic algorithms, a simulation study is conducted on an automotive suspension system [35]. The demonstration to estimate the degradation measure will follow the prognostic process discussed above for a half-car two degree of freedom model [64]. Singular perturbation methods of control theory, coupled with dynamic state estimation techniques, are employed. An IMM filter is implemented to estimate the degradation measure. The time-averaged mode probabilities are used to predict the remaining life.

The details of the system model are given in [35]. The results of 100 Monte-Carlo simulations for the system under three different road conditions viz., very good, fair, and severe road conditions are presented in Fig. 7. Compared to the severe road condition, the increases in the life times for the fair and very good roads are about 35 and 80%, respectively. If we assume a 10% calendar time usage of the automobile (2.4 h a day), the expected life of suspension system will be 4.5, 6 and 8 years, respectively, for the three road conditions. Since the suspension system has three random road conditions, the number of modes in the degradation model is 3. IMM may be viewed as a software sensor. It tracks the road condition very well based on noisy data. For IMM implementation, the following transition matrix is used in this scenario:

$$\Phi = \begin{bmatrix} 0.9 & 0.05 & 0.05 \\ 0.05 & 0.9 & 0.05 \\ 0.05 & 0.05 & 0.9 \end{bmatrix}$$

where $\phi_{ij} = P$ (mode j in effect at time $k+1|$ mode i in effect at time k). The system mode changes are expected to be as follows. Mode 1 is from 0 to 70×10^5 s, Mode 2 is from 70×10^5 s to 140×110^5 s, Mode 3 is from 140×110^5 s to t_{end}, where t_{end} is the time at which degradation measure $\xi = 1$. Figure 8 shows the plot of mode probabilities of the IMM. Figure 9 presents the estimate of remaining life (solid bold line)

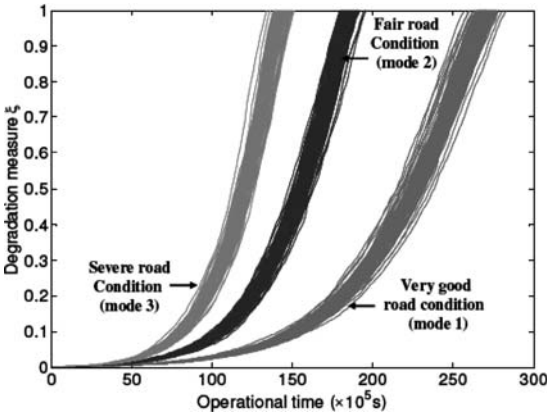

Fig. 7. 100 Monte-Carlo simulations for three random loads

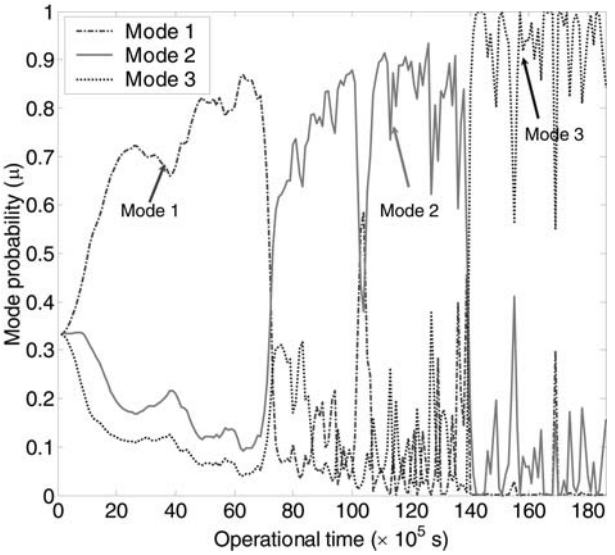

Fig. 8. Mode probabilities of IMM (Mode 1: good, Mode 2: fair, Mode 3: severe road condition)

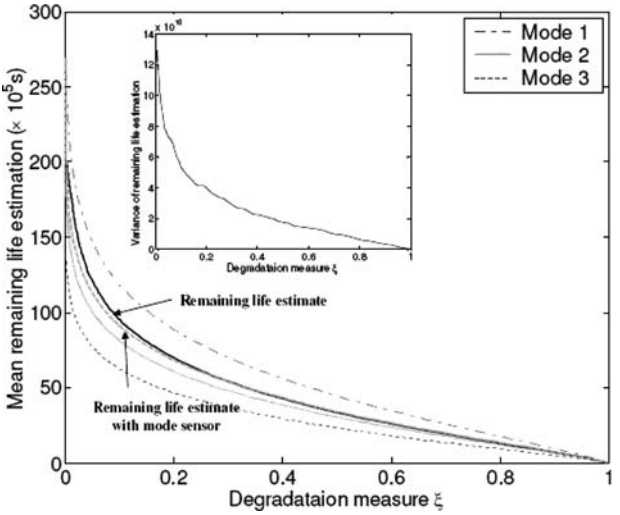

Fig. 9. Estimation of remaining life for a typical simulation run

and its variance for a single run of the scenario considered using the IMM mode probabilities. We can see that the remaining life estimate moves at first to the estimate in between Modes 1 and 2, then gradually approaches the estimate for Mode 2, which is what one would expect. The dashed bold line represents the remaining life estimate assuming that the road surface condition can be measured accurately via a sensor (e.g., an infrared sensor).

In this case, the mode is known, and we can evaluate the accuracy of the remaining life estimate. The dashed bold line represents the remaining life estimate with the mode sensor. In Fig. 9, we can see the IMM produces remaining life estimate close to the estimate of the mode sensor. The difference between these two estimates is relatively high (about 6%) at the beginning ($\xi < 0.1$), and they become virtually identical as degradation measure ξ increases.

3 Data-Driven Diagnostic Approach

Data-driven FDD techniques seek to categorize the input–output data into normal or faulty classes based on training information. Efficient data reduction techniques are employed to handle high-dimensional and/or large volumes of data. Typically, one experiments with a number of classification algorithms and fusion architectures, along with the data reduction techniques, to achieve high diagnostic accuracy. The fusion architectures minimize variability in diagnostic accuracy, and ensure better collective reliability and efficiency when compared with a single classifier.

3.1 Data-Driven Techniques

Data Preprocessing and Reduction

The sensor data obtained from a system is typically noisy and often incomplete. The data may be continuous or discrete (categorical). A linear trend, a signal mean, or noise in the raw data, outliers and drift can cause errors in the FDD analysis. Hence, it is important to preprocess the data. Data preprocessing involves filtering the data to isolate the signal components, de-trending, removing drift and outliers, smart fill in of missing values, pre-filtering and auto-scaling to adjust scale differences among variables to obtain normalized data (typically zero mean and unit variance), to name a few. In addition, traditional methods of data collection and storage capabilities become untenable mainly because of the increase in the number of variables associated with each observation ("dimension of the data") [16]. Data reduction, an intelligent preprocessing technique, synthesizes a smaller number of features to overcome the "curse of dimensionality." One of the issues with high-dimensional datasets (caused by multiple modes of system operation and sensor data over time) is that all the measurements are not salient for understanding the essential phenomena of interest. The salient features extracted using the data reduction techniques enable real-time implementation of data-driven diagnostic algorithms via compact memory footprint, improved computational efficiency and generally enhanced diagnostic accuracy.

In the data reduction process, the entire data is projected onto a low-dimensional space, and the reduced space often gives information about the important structure of the high-dimensional data space. Feature extraction for data reduction involves signal processing methods, such as wavelets, fast Fourier transforms (FFT) and statistical techniques to extract relevant information for diagnosing faults. Statistical data reduction techniques, such as multi-way partial least squares (MPLS) and multi-way principal component analysis (MPCA), are among the widely investigated techniques. The MPCA is used to reduce the dimensionality of data, and produces a representation that preserves the correlation structures among the monitored variables. The PCA is optimal in terms of capturing the variation in data [8]. The MPLS is another dimensionality reduction technique that considers both pattern (independent data) and class (response) information. MPLS technique is widely used for its ability to enhance classification accuracy on high-dimensional datasets, and its computational efficiency [5]. The reduced data can be processed with classifiers for categorizing the various fault classes.

Multi-Way Partial Least Squares (MPLS). In an MPLS technique, the dimensionality of the input and output spaces are transformed to find latent variables, which are most highly correlated with the output, i.e., those that not only explain the variation in the input tensor $X \in R^{I\times J\times K}$, but which are most predictive of output matrix $Y \in R^{I\times M}$. The input tensor X (data samples × sensors × time steps) is decomposed into one set of score vectors (latent variables) $\{\underline{t}_f \in R^I\}_{f=1}^L$, and two sets of weight vectors $\{\underline{w}_f \in R^J\}_{f=1}^L$ and $\{\underline{v}_f \in R^K\}_{f=1}^L$ in the second and third dimensions, respectively [50]. The Y matrix is decomposed into score vectors $\underline{t}$ and loading vectors $\underline{q}$. Formally,

$$\begin{aligned} x_{ijk} &= \sum_{f=1}^{L} t_{if} w_{jf} v_{kf} + e_{ijk};\ 1 \le i \le I,\ 1 \le j \le J,\ 1 \le k \le K \\ y_{im} &= \sum_{f=1}^{L} t_{if} q_{mf} + u_{im};\ 1 \le i \le I,\ 1 \le m \le M \end{aligned} \tag{27}$$

where L is the number of factors, J is number of sensor readings, K is time variations, and $\underline{e}_{ijk}$ and $\underline{u}_{ijk}$ are residuals. The problem of finding $\underline{t}$, $\underline{w}$, $\underline{v}$, and $\underline{q}$ is accomplished by nonlinear iterative partial least squares (NIPALS) algorithm [63]. This reduced space (score matrix) will be applied to the classifiers, discussed in the following section, for fault isolation.

Classifiers

Data-driven techniques for fault diagnosis have a close relationship with pattern recognition, wherein one seeks to categorize the input–output data into normal or one of several fault classes. Many classification algorithms use supervised learning to develop a discriminant function. This function is used to determine the support for a given category or class, and assigns it to one of a set of discrete classes. Once a classifier is constructed from the training data, it is used to classify test patterns. The pattern classifiers used extensively for automobile fault diagnosis are discussed below.

Support Vector Machine (SVM). Support vector machine transforms the data to a higher dimensional feature space, and finds an optimal hyperplane that maximizes the margin between two classes via quadratic programming [3, 15]. There are two distinct advantages of using the SVM for classification. One is that the features are often associated with the physical meaning of data, so that it is easy to interpret. The second advantage is that it requires only a small amount of training data. A kernel function, typically a radial basis function, is used for feature extraction. An optimal hyperplane is found in the feature space to separate the two classes. In the multi-class case, a hyperplane separating each pair of faults (classes) is found, and the final classification decision is made based on a majority vote among the binary classifiers.

Probabilistic Neural Network (PNN). The probabilistic neural network is a supervised method to estimate the probability distribution function of each class. In the recall mode, these functions are used to estimate the likelihood of an input vector being part of a learned category, or class. The learned patterns can also be weighted, with the a priori probability, called the relative frequency, of each category and misclassification costs to determine the most likely class for a given input vector. If the relative frequency of the categories is unknown, then all the categories can be assumed to be equally likely and the determination of category is solely based on the closeness of the input vector to the distribution function of a class. The memory requirements of PNN are substantial. Consequently, data reduction methods are essential in real-world applications.

K-Nearest Neighbor (KNN). The k-nearest neighbor classifier is a simple non-parametric method for classification. Despite the simplicity of the algorithm, it performs very well, and is an important benchmark method [15]. The KNN classifier requires a metric d and a positive integer k. A new input vector $\underline{x}_{new}$ is classified using a subset of k–feature vectors that are closest to $\underline{x}_{new}$ with respect to the given metric d. The new input vector $\underline{x}_{new}$ is then assigned to the class that appears most frequently within the k–subset. Ties can be broken by choosing an odd number for k (e.g., 1, 3, 5). Mathematically, this can be viewed as computing a posteriori class probabilities $P(c_i|\underline{x}_{new})$ as,

$$P\left(c_i|\underline{x}_{new}\right) = \frac{k_i}{k} p\left(c_i\right) \tag{28}$$

where k_i is the number of vectors belonging to class c_i within the subset of k nearest vectors. A new input vector $\underline{x}_{new}$ is assigned to the class c_i with the highest posterior probability $P(c_i|\underline{x}_{new})$. The KNN classifier needs to store all previously observed cases, and thus data reduction methods should be employed for computational efficiency.

Principal Component Analysis (PCA). Principal component analysis transforms correlated variables into a smaller number of uncorrelated variables, called principal components. PCA calculates the covariance matrix of the training data and the corresponding eigenvalues and eigenvectors. The eigenvalues are then sorted, and the vectors (called scores) with the highest values are selected to represent the data in a reduced space. The number of principal components is determined by cross-validation [27]. The score vectors from different principal components are the coordinates of the original data sample in the reduced space. A classification of a new test pattern (data sample) is made by obtaining its predicted scores and residuals. If the test pattern is similar to a specific class in the trained classifier, the scores will be located near the origin of the reduced space, and the residual should be small. The distance of test pattern from the origin of the reduced space can be measured by Hotelling statistic [39].

Linear Discriminant Analysis (LD) and Quadratic Discriminant Analysis (QD). Discriminant functions can be related to the class-conditional density functions through Bayes' theorem [44]. The decision rule for minimizing the probability of misclassification may be cast in terms of discriminant functions. Linear discriminant function can be written as

$$g_i\left(\underline{x}\right) = \underline{w}_i^T \underline{x} + w_{i0} \tag{29}$$

where $\underline{w}_i$ and w_{i0} are the weight vector and bias for the ith class, respectively. Decision boundaries corresponding to linear discriminant functions are hyper planes. Quadratic discriminant function can be obtained by adding terms corresponding to the covariance matrix with $c(c+1)/2$ coefficients to produce more complicated separating surfaces [15]. The separating surfaces can be hyperquadratic, hyperspheric, hyperellipsoid, hyperhyperboloid, etc.

Classifier Fusion Techniques

Fusion techniques combine classifier outputs, viz., single class labels or decisions, confidence (or probability) estimates, or class rankings, for higher performance and more reliable diagnostic decisions than a single classifier alone. The accuracy of each classifier in a fusion ensemble is not the same. Thus, the classifier's priority or weight needs to be optimized as part of the fusion architecture for improved performance.

Many fusion techniques are explored in the area of fault diagnosis [10, 14]. Some of these are discussed below:

Fusion of Classifier Output Labels. If a classifier's final decision on a test pattern is a single class label, an ensemble of R classifiers provides R discrete output labels. The following algorithms use output labels from each classifier and combine them into a final fused decision.

(1) Majority Voting: The simplest type of fusion, majority (plurality) voting counts votes for each class from the classifiers. The class with the most votes is declared the winner. If a tie exists for the most votes, either it can be broken arbitrarily or a "tie class label" can be assigned. This type of fusion does not require any training or optimized architecture.

(2) Weighted Voting: In weighted voting, a weight calculated during training of the fusion architecture is used to calculate the overall score of each class. The higher the classifier accuracy, the more weight that classifier is given. A score is constructed for each class by using a sum of weighted votes for each class c_i [14]. The class with the highest score is declared the winner.

(3) Naïve Bayes: Classifiers often have very different performance across classes. The confusion matrix of a classifier (derived from training data) contains this information. The entries in a confusion matrix $cm^i_{k,s}$ represent the number of times true class c_k is labeled class c_s by the classifier D_i. Support for each class k on pattern x is developed as [31]:

$$\mu_k\left(x\right) \propto \frac{1}{N_k^{L-1}} \left\{ \prod_{i=1}^{L} cm^i_{k,s_i} \right\} \tag{30}$$

Here, N_k is the number of training samples from class c_k, L is the number of classifiers. The class with the highest support is declared as the winner.

Fusion of Classifier Output Ranks. The output of classifiers can be a ranking of the preferences over the C possible output classes. Several techniques operating on this type of output are discussed below.

(1) Borda Count: The ranked votes from each classifier are assigned weights according to their rank. The class ranked first is given a weight of C, the second a weight of $(C-1)$ and so on until a weight of 1 is assigned for the class ranked last. The score for each class is computed as the sum of the class weights from each classifier and the winner is the class with the highest total weight [31].

(2) Ranked Pairs: Ranked Pairs is a voting technique where each voter participates by listing his/her preference of the candidates from the most to the least preferred. In a ranked pair election, the majority preference is sought as opposed to the majority vote or the highest weighted score. That is, we combine the outputs of classifiers to maximize the mutual preference among the classifiers. This approach assumes that voters have a tendency to pick the correct winner [31]. This type of fusion, as in majority voting, does not require any training. If a crisp label is required as a final output, the first position in the ranked vector RV is provided as the final decision.

Fusion of Classifier Posterior Probabilities. The output of a classifier can be an array of confidence estimates or posterior probability estimates. These estimates represent the belief that the pattern belongs to each of the classes. The techniques in this section operate on the values in this array to produce a final fusion label.

(1) Bayesian Fusion: Class-specific Bayesian approach to classifier fusion exploits the fact that different classifiers can be good at classifying different fault classes. The most-likely class is chosen given the test pattern and the training data using the total probability theorem. The posterior probabilities of the test pattern along with the associated posterior probabilities of class c_i from each of the R classifiers obtained during training are used to select the class with the highest posterior probability [10].

(2) Joint Optimization of Fusion Center and of Individual Classifiers: In this technique, the fusion center must decide on the correct class based on its own data and the evidence from the R classifiers. A major result of distributed detection theory (e.g., [44, 59, 60]) is that the decision rules of the individual classifiers and the fusion center are *coupled*. The decisions of individual classifiers are denoted by $\{u_k\}_{k=1}^{L}$ while the decision of fusion center by u_0: The classification rule of kth classifier is $u_k = \gamma_k(\underline{x}) \in \{1, 2, \ldots, C\}$ and that of the fusion center is $u_0 = \gamma_0(u_1, u_2, \ldots, u_L) \in \{1, 2, \ldots, C\}$. Let $J(u_0, c_j)$ be the cost of decision u_0 by the committee of classifiers when the true class is c_j. The joint committee strategy of the fusion center along with the classifiers is formulated to minimize the expected cost $E\{J(u_0, c_j)\}$. For computational efficiency, an assumption is made to correlate each classifier only with the best classifier during training to avoid the computation of exponentially increasing entries with the number of classifiers in the joint probability [10]. The decision rule can be written as

$$\gamma_k : u_k = \arg \min_{d_k \in \{1,2,\cdots,C\}} \sum_{j=1}^{C} \sum_{u_0=1}^{C} P_k\left(c_j|x\right) \hat{J}\left(u_0, c_j\right) \tag{31}$$

where

$$\begin{aligned} \hat{J}\left(u_0, c_j\right) &= \sum_{u_0=1}^{C} P\left(u_0|x, u_k = d_k, c_j\right) J\left(u_0, c_j\right) \\ &\approx \sum_{u_0=1}^{C} P\left(u_0|u_k = d_k, c_j\right) J\left(u_0, c_j\right). \end{aligned} \tag{32}$$

Dependence Tree Architectures. We can combine classifiers using a variety of fusion architectures to enhance the diagnostic accuracy [44, 59, 60]. The class-dependent fusion architectures are developed based on the diagnostic accuracies of individual classifiers on the training data for each class. The classifiers are arranged as a dependence tree to maximize the sum of mutual information between all pairs of classifiers [31].

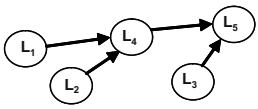

Fig. 10. Generic decision tree architecture

For illustrative purposes, consider Fig. 10, where five classifiers are arranged in the form of a tree. Suppose that the classifiers provide class labels $\{L_j\}_{j=1}^{5}$. Then, the support for class c_i is given by:

$$P(\{L_j\}_{j=1}^{5}|c_i) = P(L_5|c_i)P(L_5|L_4, c_i)P(L_5|L_3, c_i)P(L_4|L_1, c_i)P(L_4|L_2, c_i) \tag{33}$$

Here, the term $P(L_5|c_i)$ denotes the probability of label L_5 given the true class c_i from the confusion matrix of classifier 5. The double entries of the form $P(L_k|L_j, c_i)$ represent the output labels of classifiers k and j in the coincidence matrix developed from classifiers k and j on class c_i during training. The final decision corresponds to the class with the highest probability in (33).

Adaptive Boosting (AdaBoost). AdaBoost [18], short for adaptive boosting, uses the same training set randomly and repeatedly to create an ensemble of classifiers for fusion. This algorithm allows adding weak learners, whose goal is to find a weak hypothesis with small pseudo-loss[1], until a desired low level of training error is achieved. To avoid more complex requirement on the performance of the weak hypothesis, pseudo loss is chosen in place of the prediction error. The pseudo-loss is minimized when correct labels y_i are assigned the value 1, and incorrect labels are assigned the value 0, and it is also calculated with respect to a distribution over all pairs of patterns and incorrect labels. By controlling the distribution, the weak learners can focus on the incorrect labels, thereby hopefully improving the overall performance.

Error-Correcting Output Codes (ECOC). Error-correcting output codes (ECOC) can be used to solve multi-class problems by separating the classes into dichotomies and solving the concomitant binary classification problems, one for each column of the ECOC matrix. The dichotomies are chosen using the principles of orthogonality to ensure maximum separation of rows and columns to enhance the error-correcting properties of the code matrix and to minimize correlated errors of the ensemble, respectively. The maximum number of dichotomies for C classes is $2^{C-1} - 1$; however, it is common to use much less than this maximum as in robust design [44]. Each dichotomy is assigned to a binary classifier, which will decide if a pattern belongs to the 0 or 1 group. Three approaches to fuse the dichotomous decisions are discussed below:

(1) Hamming Distance: Using Hamming distance, we compute the number of positions which are different between the row representing a class in the ECOC matrix and the output of the classifier bank. The class which has the minimum distance is declared as the output.

(2) Weighted Voting: Each classifier j detects class i with a different probability. As the multi-class problem is converted into dichotomous classes using ECOC, the weights of each classifier can be expressed in terms of the probability of detection (Pd_j) and the probability of false alarm (Pf_j). These parameters are learned as part of fusion architecture during training. The weighted voting follows the optimum voting rules for binary classifiers [44].

(3) Dynamic fusion: Dynamic fusion architecture, combining ECOC and dynamic inference algorithm for factorial hidden Markov models, accounts for temporal correlations of binary time series data [30, 55]. The fusion process involves three steps: the first step transforms the multi-class problem into dichotomies using error correcting output codes (ECOC) and thus solving the concomitant binary classification problems; the second step fuses the outcomes of multiple binary classifiers over time using a sliding-window dynamic fusion method. The dynamic fusion problem is formulated as a maximum *a posteriori* decision problem of inferring the fault sequence based on uncertain binary outcomes of multiple classifiers over time. The resulting problem is solved via a primal-dual optimization framework [56]. The third step optimizes the fusion parameters using a genetic algorithm. The dynamic fusion process is shown in Fig. 11. The probability of detection Pd_j and false alarm probability Pf_j of each classifier are employed as fusion parameters or the classifier weights; these probabilities are jointly optimized with the dynamic fusion in the fusion architecture, instead of optimizing the parameters of each classifier separately.

[1] True loss is non-differentiable and difficult to optimize [3].

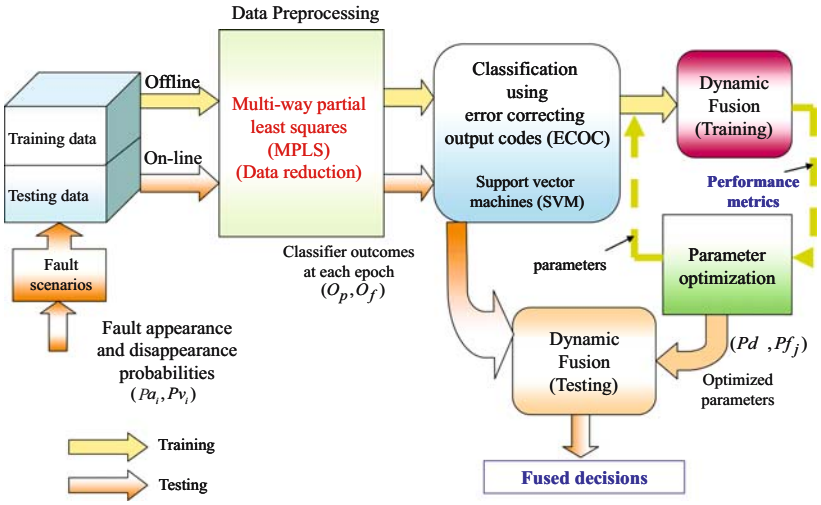

Fig. 11. Overview of the dynamic fusion architecture

This technique allows tradeoff between the size of the sliding window (diagnostic decision delay) and improved accuracy by exploiting the temporal correlations in the data; it is suitable for an on-board application [30]. A special feature of the proposed dynamic fusion architecture is the ability to handle multiple and intermittent faults occurring over time. In addition, the ECOC-based dynamic fusion architecture is an ideal framework to investigate heterogeneous classifier combinations that employ data-driven (e.g., support vector machines, probabilistic neural networks), knowledge-based (e.g., TEAMS-RT [47]), and model-based classifiers (e.g., parity relation-based or observer-based) for the columns of the ECOC matrix.

Fault Severity Estimation

Fault severity estimation is performed by regression techniques, such as the partial least squares (PLS), SVM regression (SVMR), and principal component regression (PCR) in a manner similar to their classification counterparts. After a fault is isolated, we train with the training patterns from the isolated class using the associated severity levels as the targets (Y), i.e., we train the fault severity estimator for each class. Pre-classified test patterns are presented to the corresponding estimator, and the estimated severity levels are obtained [9].

3.2 Application of Data-Driven Techniques

We consider the CRAMAS® engine data considered earlier, but now from a data-driven viewpoint[2]. A 5×2 cross-validation[3] is used to assess the classification performance of various data-driven techniques.

The diagnostic results, measured in terms of classification errors, with $\pm$ representing standard deviations over 5×2 cross validation experiments, are shown in Table 3. We achieved not only smaller fault isolation

[2] The throttle actuator fault F5 is not considered in the data driven approach. HILS data was available only for the remaining eight faults.

[3] A special case of cross validation where the data is divided into two halves, one for training and other for testing. Next time the sets are reversed. This process is repeated for 5 times for a total of 10 training and test sets.

Table 3. Data driven classification and fusion results on CRAMAS® engine data

CRAMAS®	Method	Classification error ± Std. Dev. in %					
		SVM	KNN($k = 1$)	PNN	PCA	LD	QD
Raw data (25.6 MB)	Individual classification	8.8 ± 2.5	12.9 ± 2.2	14.8 ± 2.1	22.5 ± 2.3	N/A	N/A
Reduced data via MPLS (12.8 KB)	Individual classification	8.2 ± 2.5	12.8 ± 2.1	14.1 ± 2.1	21.1 ± 3.7	33.1 ± 3.2	16.3 ± 2.3
	Tandem (serial) fusion	15.87 ± 2.49					
	Fusion center (parallel)	14.81 ± 3.46					
	Majority voting	12.06 ± 1.89					
	Naïve Bayes	11.81 ± 1.96					
	ECOC fusion with hamming distance	9.0 ± 2.85					
	Adaboost	7.625 ± 2.14					
	Bayesian	6.25 ± 2.29					
	fusion Joint optimization with majority voting	5.87 ± 2.04					
	Dynamic fusion	4.5 ± 1.6					

error, but also significant data reduction (25.6 MB → 12.8 KB for the size of training and testing data). The proposed approaches are mainly evaluated on the reduced data. The Bayesian and dynamic fusion outperformed majority voting, naïve Bayes techniques and serial and parallel fusion approaches. We are able to further improve classification performance of joint optimization by applying majority voting after getting decisions from the joint optimization algorithm. Posterior probabilities from PNN, KNN ($k = 3$), and PCA are fed into the joint optimization algorithm, and then SVM and KNN ($k = 1$) are used for majority voting with decisions from the joint optimization algorithm. Majority voting alone provided poor isolation results, which means that the joint optimization approach is definitely a contributor to the increased accuracy. We believe that this is because the joint optimization of fusion center and individual classifiers increases the diversity of the classifier outputs, which is a vital requirement for reducing the diagnostic errors.

For the dynamic fusion approach, we employ SVM as the base classifier for all the columns of the ECOC matrix. This approach achieves low isolation errors as compared to single classifier results. We experimented with two different approaches for Pd and Pf in dynamic fusion process. The first approach used Pd and Pf learned from the training data, while coarse optimization is applied to learn Pd and Pf, and the optimal parameters are $Pd = 0.5 \sim 0.6$ and $Pf = 0 \sim 0.02$. We found that the dynamic fusion approach involving the parameter optimization reduces diagnostic errors to about 4.5%. Dynamic fusion with parameter optimization is superior to all other approaches considered in this analysis.

The severity estimation results for raw data and reduced data are shown in Table 4. For training and testing, we randomly selected 60% for training (24 levels for each class) and 40% for testing (16 levels for each class). Relative errors in % are averaged for 16 severity levels in Table 4. We have applied three different estimators, PLS, SVMR, and PCR. Large errors with the raw data can be attributed to ill-conditioning of the parameter estimation problem due to collinearity of data when compared to the reduced data. It is evident that faults 1, 3, and 6 provided poor estimation performance on raw data due to difficulties in estimating low severity levels. However, significant performance improvement can be observed when the estimators are applied to the reduced data. PLS is slightly better than SVMR and PCR in terms of severity estimation performance and provides good estimation results for high severity levels, although estimating low severity levels remains a problem. In all cases, SVMR and PCR are comparable to the PLS in terms of fault severity estimation performance. It is also observed that our techniques perform better on the reduced dataset in terms of severity estimation accuracy.

In addition to individual classifiers, such as the SVM, PNN, KNN, and PCA for fault isolation, posterior probabilities from these classifiers can be fused by the novel Bayesian fusion, joint optimization of fusion and

Table 4. Comparison of severity estimation performance on raw and reduced data

Fault	Average error, 100% × (true severity level – its estimate)/true level					
	PLS		SVMR		PCR	
	Raw (%)	Reduced (%)	Raw (%)	Reduced (%)	Raw (%)	Reduced (%)
Air flow sensor fault (F1)	−66.88	+4.02	−9.21	−6.14	+23.13	+1.06
Leakage in air intake system (F2)	−10.11	+0.76	−0.20	−0.72	−11.22	+0.75
Blockage of air filter (F3)	−75.55	+6.42	+1.37	+0.75	−44.20	+6.38
Throttle angle sensor fault (F4)	+0.63	−1.28	−1.19	+1.31	+5.51	−0.35
Less fuel injection (F6)	−73.42	−30.92	+8.04	+6.77	−51.36	−28.60
Added engine friction (F7)	+23.38	+1.43	+4.84	+6.97	+27.20	+1.73
Air/fuel sensor fault (F8)	+36.32	+0.40	−2.01	−2.90	−26.28	−0.16
Engine speed sensor fault (F9)	−7.14	+10.46	−25.19	−26.23	−1.55	−3.08
Overall % of error	−21.60	−1.09	−2.94	−2.52	−9.85	−2.78

individual classifiers, and dynamic fusion approaches. Our results confirm that fusing individual classifiers can increase the diagnostic performance substantially and that fusion reduces variability in diagnostic classifier performance. In addition, regression techniques such as the PLS, SVMR and PCR estimate the severity of the isolated faults very well when the data is transformed into a low-dimensional space to reduce noise effects.

4 Hybrid Model-Based and Data-Driven Diagnosis

Due to the very diverse nature of faults and modeling uncertainty, no single approach is perfect on all problems (no-free-lunch theorem). Consequently, a hybrid approach that combines model-based and data-driven techniques may be necessary to obtain the required diagnostic performance in complex automotive applications. Here, we present an application involving fault diagnosis in an anti-lock braking system (ABS) [36], where we integrated model and data-driven diagnostic schemes. Specifically, we combined parity equations, nonlinear observer, and SVM to diagnose faults in an ABS. This integrated approach is necessary since neither model-based nor data-driven strategy could adequately solve the entire ABS diagnosis problem, i.e., isolate faults with sufficient accuracy.

4.1 Application of Hybrid Diagnosis Process

We consider longitudinal braking with no steering, and neglect the effects of pitch and roll. The model considers the wheel speed and vehicle speed as measured variables, and the force applied to the brake pedal as the input. The wheel speed is directly measured and vehicle speed can be calculated by integrating the measured acceleration signals, as in [62]. Further details of the model are found in [36]. One commonly occurring sensor fault and four parametric faults are considered for diagnosis in the ABS system. In the case of a wheel speed sensor fault, the sensor systematically misses the detection of teeth in the wheel due to incorrect wheel speed sensor gap caused by loose wheel bearings or worn parts. In order to model the wheel speed sensor fault (F1), we consider two fault severity cases: greater than 0 but less than 5% reduction in the nominal wheel speed (F1.1), and greater than 5% reduction in the nominal wheel speed (F1.2). The four parametric faults (F2–F5) are changes in radius of the wheel (R_w), torque gain (K_f), rotating inertia of the wheel (I_w) and the time constant of the Master Cylinder (τ_m). Fault F2 is the tire pressure fault, F3 and F5 correspond to cylinder faults, while F4 is related to vehicle body. Faults corresponding to more than 2% decrease in R_w are considered. We distinguish among two R_w faults: greater than 2% but less than 20% (F2.1) decrease in R_w, and greater than 20% decrease in R_w (F2.2). The severities or sizes for K_f and I_w faults considered are as follows: $\pm 2, \pm 3, \ldots, \pm 10\%$. The size for τ_m fault corresponds to a more than 15% increase in the time constant. Table 5 shows the list of considered faults. The minimum fault magnitude is

selected such that changes in the residual signals can not be detected if we choose fault magnitude less than this minimum. The measurement variables for vehicle and wheel speed are corrupted by the zero mean white noise with variances of 0.004 each. The process noise variables are also white with variance of 0.5% of the mean square values of the corresponding states (which corresponds to a signal-to-noise ratio of +23 dB).

A small amount of process noise is added based on the fact that these states are driven by disturbances from combustion processes in the engine (un-modeled dynamics of wheel and vehicle speeds), and non-linear effects in the ABS actuator (for brake torque and oil pressure).

Figure 12 shows the block diagram of our proposed FDD scheme for the ABS. The parity equations and GLRT test (G_P_1) are used to detect severe Rw($\geq$20%) and wheel speed sensor ($\geq$5%) faults. Then, a nonlinear observer [17, 36] is used to generate two additional residuals. The GLRTs based on these two residuals (G_O_1 and G_O_2) and their time dependent GLRT test ($G\_O\_T_1$ and $G\_O\_T_2$) are used to isolate the τ_m fault, less severe (small) Rw and sensor faults. They are also used to detect K_f and I_w faults. Finally, we use the SVM to isolate the K_f and I_w faults. After training, a total of 35 patterns are misclassified in the test data, which results in an error rate of 4.7%. We designed two tests S_K_f and S_I_w using the SVM, which assigns $S\_K_f = 1$ when the data is classified as the K_f fault or assigns $S\_I_w = 1$ when the data is classified as the I_w fault. The diagnostic matrix of the ABS system is shown in Table 6. With the subset of tests, all the faults considered here can be detected. Subsequently, a parameter estimation technique is used

Table 5. Simulated fault list of ABS system

F1.1	Sensor fault (<5% decrease)
F1.2	Sensor fault ($\geq$5% decrease)
F2.1	R_w fault (<20% decrease)
F2.2	R_w fault ($\geq$20% decrease)
F3	K_f fault ($\pm 2\% \sim \pm 10\%$)
F4	I_w fault ($\pm 2\% \sim \pm 10\%$)
F5	τ_m fault ($\geq$15% increase)

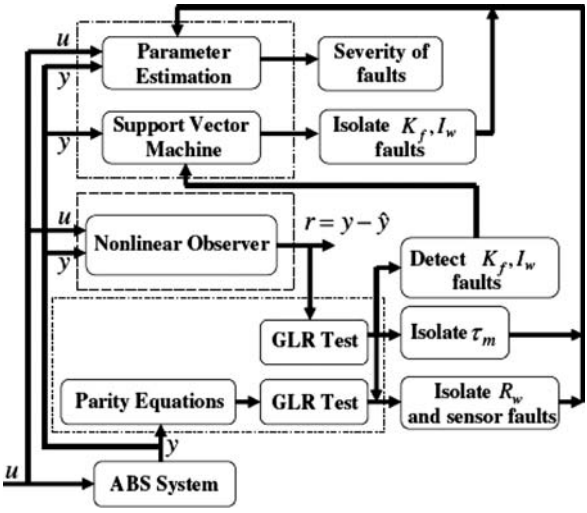

Fig. 12. FDD Scheme for ABS

Table 6. Diagnostic matrix for ABS test design

Fault\Test	G_P_1	G_O_1	G_O_2	$G\_O\_T_1$	G_O_T2	S_Kf	S_Iw
F0	0	0	0	0	0	0	0
F1.1	0	1	0	0	0	0	0
F1.2	0	0	1	1	0	0	1
F2.1	0	0	1	1	0	0	0
F2.2	0	0	0	1	0	0	0
F3	1	0	0	0	0	0	0
F4	0	0	0	1	1	1	1
F5	0	0	0	1	0	0	0

Table 7. Mean relative errors and normalized standard deviations in parameter estimation

		Block Estimation				Subset Parameter Estimation
		R_w	K_f	I_w	τ_m	
K_f	*err*	3.2	5.0	6.0	25.0	1.05
	std	1.2	3.5	6.8	22.2	0.12
I_w	*err*	2.0	4.5	4.0	19.0	0.52
	std	1.6	4.8	7.2	39.3	0.35
τ_m	*err*	3.5	7.8	10.3	27.5	2.0
	std	2.4	5.2	5.6	46.5	0.80
R_w	*err*	0.39	0.33	2.98	279.33	0.004
	std	0.25	0.12	1.48	33.4	0.014

$err = \frac{\text{mean relative error}}{\text{“true” value}} \times 100\%$

$std = \frac{\text{standard deviation of estimated parameters}}{\text{“true” value}} \times 100\%$

after fault isolation to estimate the severity of the fault. After parametric faults are isolated, an output error method is used to estimate the severity of isolated faults. In the ABS, the nonlinear output error parameter estimation method produces biased estimates when all the parameters are estimated as a block. Therefore, the subset parameter estimation techniques are well suited for our application. The subset of parameters to be estimated is chosen by the detection and isolation of the parametric fault using the GLRT and SVM. When a parametric fault is isolated, this parameter is estimated via the nonlinear output error method. Table 7 compares the accuracies of parameter estimation averaged over 20 runs via the two methods: estimating all the parameters versus reduced (one-at-a-time) parameter estimation after fault detection and isolation. The parameters *err* and *std* shows the mean relative errors and standard deviations of the estimated parameters, respectively, normalized by their “true” values (in %).

From Table 7, it is evident that subset parameter estimation provides much more precise estimates than the method which estimates all four parameters as a block. This is especially significant with single parameter faults.

5 Summary and Future Research

This chapter addressed an integrated diagnostic development process for automotive systems. This process can be employed during all stages of a system life cycle, viz., concept, design, development, production, operations, and training of technicians to ensure ease of maintenance and high reliability of vehicle systems by performing testability and reliability analyses at the design stage. The diagnostic design process employs both model-based and data-driven diagnostic techniques. The test designers can experiment with a combination of these techniques that are appropriate for a given system, and trade-off several performance

evaluation criteria: detection speed, detection and isolation accuracy, computational efficiency, on-line/off-line implementation, repair strategies, time-based versus preventive versus condition-based maintenance of vehicle components, and so on. The use of condition-based maintenance, on-line system health monitoring and smart diagnostics and reconfiguration/self-healing/repair strategies will help minimize downtime, improve resource management, and minimize operational costs. The integrated diagnostics process promises a major economic impact, especially when implemented effectively across an enterprise.

In addition to extensive applications of the integrated diagnostics process to real-world systems, there are a number of research areas that deserve further attention. These include: dynamic tracking of the evolution of degraded system states (the so-called "gray-scale diagnosis"), developing rigorous analytical framework for combining model-based and data-driven approaches for adaptive knowledge bases, adaptive inference, agent-based architectures for distributed diagnostics and prognostics, use of diagnostic information for reconfigurable control, and linking the integrated diagnostic process to supply chain management processes for effective parts management.

References

1. Bar-Shalom Y, Li XR, and Kirubarajan T (2001) *Estimation with Applications to Tracking and Navigation*. Wiley, New York, 2001
2. Basseville M and Nikiforov IV (1993) *Detection of Abrupt Changes*. Prentice-Hall, New Jersey
3. Bishop CM (2006) *Pattern Recognition and Machine Learning*. Springer, Berlin Heidelberg New York
4. Bohr J (1998) Open systems approach – integrated diagnostics demonstration program. *NDIA Systems Engineering and Supportability Conference and Workshop*, http://www.dtic.mil/ndia/support/bohr.pdf
5. Bro R (1996) Multiway Calibration. Multilinear PLS. *Journal of Chemometrics* 10:47–61
6. Chelidze D (2002) Multimode damage tracking and failure prognosis in electro mechanical system. *SPIE Conference Proceedings*, pp 1–12
7. Chelidze D, Cusumano JP, and Chatterjee A (2002) Dynamical systems approach to damage evolution tracking, part I: The experimental method. *Journal of Vibration and Acoustics* 124:250–257
8. Chen J, Liu K (2002) On-line batch process monitoring using dynamic PCA and dynamic PLS models. *Chemical Engineering Science* 57:63–75
9. Choi K, Luo J, Pattipati K, Namburu M, Qiao L, and Chigusa S (2006) Data reduction techniques for intelligent fault diagnosis in automotive systems. *Proceedings of IEEE AUTOTESTCON*, Anaheim, CA, pp 66–72
10. Choi K, Singh S, Kodali A, Pattipati K, Namburu M, Chigusa S, and Qiao L (2007) A novel Bayesian approach to classifier fusion for fault diagnosis in automotive systems. *Proceedings of IEEE AUTOTESTCON*, Baltimore, MD, pp 260–269
11. Deb S, Pattipati K, Raghavan V, Shakeri M, and Shrestha R (1995) Multi-signal flow graphs: A novel approach for system testability analysis and fault diagnosis. *IEEE Aerospace and Electronics Magazine*, pp 14–25
12. Deb S, Pattipati K, and Shrestha R (1997) QSI's Integrate diagnostics toolset. *Proceedings of the IEEE AUTOTESTCON*, Anaheim, CA, pp 408–421
13. Deb S, Ghoshal S, Mathur A, and Pattipati K (1998) Multi-signal modeling for diagnosis, FMECA and reliability. *IEEE Systems, Man, and Cybernetics Conference*, San Diego, CA
14. Donat W (2007) *Data Visualization, Data Reduction, and Classifier Output Fusion for Intelligent Fault Detection and Diagnosis*. M.S. Thesis, University of Connecticut
15. Duda RO, Hart PE, and Stork DG (2001) *Pattern Classification* (2nd edn.). Wiley, New York
16. Fodor K, *A survey of dimension reduction techniques*. Available: http://www.llnl.gov/CASC/sapphire/pubs/148494.pdf
17. Frank PM (1994) On-line fault detection in uncertain nonlinear systems using diagnostic observers: a survey. *International Journal of System Science* 25:2129–2154
18. Freund Y and Schapire RE (1996) Experiments with a new boosting algorithm. Machine Learning: *Proceedings of the Thirteenth International Conference*
19. Fukazawa M (2001) Development of PC-based HIL simulator CRAMAS 2001, *FUJITSU TEN Technical Journal* 19:12–21
20. Garcia EA and Frank P (1997) Deterministic nonlinear observer based approaches to fault diagnosis: a survey. *Control Engineering Practice* 5:663–670
21. Gertler J (1995) Fault detection and isolation using parity relations. *Control Engineering Practice* 5:1385–1392

22. Gertler J and Monajmey R (1995) Generating directional residuals with dynamic parity relations. *Automatica* 33:627–635
23. Higuchi T, Kanou K, Imada S, Kimura S, and Tarumoto T (2003) Development of rapid prototype ECU for power train control. *FUJITSU TEN Technical Journal* 20:41–46
24. Isermann R (1984) Process fault detection based on modeling and estimation methods: a survey. *Automatica* 20:387–404
25. Isermann R (1993) Fault diagnosis of machines via parameter estimation and knowledge processing-tutorial paper. *Automatica* 29:815–835
26. Isermann R (1997) Supervision, fault-detection and fault-diagnosis methods – an introduction. *Control Engineering Practice* 5:639–652
27. Jackson JE (1991) *A User's Guide to Principal Components*. Wiley, New York
28. Johannesson (1998) Rainflow cycles for switching processes with Markov structure. *Probability in the Engineering and Informational Sciences* 12:143–175
29. Keiner W (1990) A navy approach to integrated diagnostics. *Proceedings of the IEEE AUTOTESTCON*, pp 443–450
30. Kodali A, Donat W, Singh S, Choi K, and Pattipati K (2008) Dynamic fusion and parameter optimization of multiple classifier systems. *Proceedings of GT 2008*, Turbo Expo 2008, Berlin, Germany
31. Kuncheva LI (2004) *Combining Pattern Classifiers*, Wiley, New York
32. Ljung L (1987) *System Identification: Theory for the User*, Prentice-Hall, New Jersey
33. Luo J, Tu F, Azam M, Pattipati K, Qiao L, and Kawamoto M (2003) Intelligent model-based diagnostics for vehicle health management. *Proceedings of SPIE Conference*, Orlando, pp 13–26
34. Luo J, Tu H, Pattipati K, Qiao L, and Chigusa S (2006) Graphical models for diagnostic knowledge representation and inference. *IEEE Instrument and Measurement Magazine* 9:45–52
35. Luo J, Pattipati K, Qiao L, and Chigusa S (2007) An integrated diagnostic development process for automotive engine control systems. *IEEE Transactions on Systems, Man, and Cybernetics: Part C – Applications and Reviews* 37:1163–1173
36. Luo J, Namburu M, Pattipati K, Qiao L, and Chigusa S (2008) Integrated model-based and data-driven diagnosis of automotive anti-lock braking systems. *IEEE System, Man, and Cybernetics – Part A: Systems and Humans*
37. Luo J, Pattipati K, Qiao L and Chigusa S (2008) Model-based prognostic techniques applied to a suspension system. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews*
38. Namburu M (2006) *Model-Based and Data-Driven Techniques and Their Application to Fault Detection and Diagnosis in Engineering Systems and Information Retrieval*. M.S. Thesis, University of Connecticut
39. Nomikos P (1996) Detection and diagnosis of abnormal batch operations based on multi-way principal component analysis. *ISA Transactions* 35:259–266
40. Nyberg M and Nielsen L (1997) Model based diagnosis for the air intake system of the SI-engine. SAE Transactions. *Journal of Commercial Vehicles* 106:9–20
41. Pattipati K (2003) Combinatorial optimization algorithms for fault diagnosis in complex systems. *International Workshop on IT-Enabled Manufacturing, Logistics and Supply Chain Management*, Bangalore, India
42. Pattipati K and Alexandridis M (1990) Application of heuristic search and information theory to sequential fault diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics – Part A* 20:872–887
43. Patton RJ, Frank PM, and Clark RN (2000) *Issues of Fault Diagnosis for Dynamic Systems*, Springer, Berlin Heidelberg New York London
44. Pete A, Pattipati K, and Kleinman DL (1994) Optimization of detection networks with generalized event structures. *IEEE Transactions on Automatic Control* 1702–1707
45. Phadke MS (1989) *Quality Engineering Using Robust Design*. Prentice Hall New Jersey
46. Phelps E and Willett P (2002) Useful lifetime tracking via the IMM. *SPIE Conference Proceedings*, pp 145–156, 2002
47. *QSI website*, http://www.teamsqsi.com
48. Raghavan V, Shakeri M, and Pattipati K (1999) Test sequencing algorithms with unreliable tests. *IEEE Transactions on Systems, Man, and Cybernetics – Part A* 29:347–357
49. Raghavan V, Shakeri M, and Pattipati K (1999) Optimal and near-optimal test sequencing algorithms with realistic test models. *IEEE Transactions on Systems, Man, and Cybernetics – Part A* 29:11–26
50. Rasmus B (1996) Multiway calibration. Multilinear PLS. *Journal of Chemometrics* 10:259–266
51. Ruan S, Tu F, Pattipati K, and Patterson-Hine A (2004) On a multimode test sequencing problem. *IEEE Transactions on Systems, Man and Cybernetics – Part B* 34:1490–1499

52. Schroder D (2000) *Intelligent Observer and Control Design for Nonlinear Systems*, Springer, Berlin Heidelberg New York
53. Shakeri M (1998) *Advances in System Fault Modeling and Diagnosis*. Ph.D. Thesis, University of Connecticut
54. Simani S, Fantuzzi C, and Patton RJ (2003) *Model-Based Fault Diagnosis in Dynamic Systems Using Identification Techniques*. Springer, Berlin Heidelberg New York London
55. Singh S, Choi K, Kodali A, Pattipati K, Namburu M, Chigusa S, and Qiao L (2007) Dynamic classifier fusion in automotive systems. *IEEE SMC Conference*, Montreal, Canada
56. Singh S, Kodali A, Choi K, Pattipati K, Namburu M, Chigusa S, Prokhorov DV, and Qiao L (2008) Dynamic multiple fault diagnosis: mathematical formulations and solution techniques. *accepted for IEEE Trans. on SMC – Part A*. To appear
57. Sobczyk K and Spencer B (1993) *Random Fatigue: From Data to Theory*. Academic, San Diego
58. Sobczyk K and Trebicki J (2000) Stochastic dynamics with fatigue induced stiffness degradation. *Probabilistic Engineering Mechanics* 15:91–99
59. Tang ZB, Pattipati K, and Kleinman DL (1991) Optimization of detection networks: Part I – tandem structures. *IEEE Transactions on Systems, Man, and Cybernetics: Special Issue on Distributed Sensor Networks* 21:1045–1059
60. Tang ZB, Pattipati K, and Kleinman DL (1992) A Distributed M-ary hypothesis testing problem with correlated observations. *IEEE Transactions on Automatic Control*, 196:32 pp 1042–1046
61. Terry B and Lee S (1995) What is the prognosis on your maintenance program. *Engineering and Mining Journal*, 196:32
62. Unsal C and Kachroo P (1999) Sliding mode measurement feedback control for antilock braking system. *IEEE Transactions on Control Systems Technology* 7:271–281
63. Wold S, Geladi P, Esbensen K, and Ohman J (1987) *Principal Component Analysis Chemometrics and Intelligent Laboratory System* 2:37–52
64. Yoshimura T, Nakaminami K, Kurimoto M, and Hino J (1999) Active suspension of passenger cars using linear and fuzzy logic controls. *Control Engineering Practice* 41:41–47