

Moamar Sayed-Mouchaweh *Editor*

Fault Diagnosis of Hybrid Dynamic and Complex Systems



Springer

Fault Diagnosis of Hybrid Dynamic and Complex Systems

Moamar Sayed-Mouchaweh
Editor

Fault Diagnosis of Hybrid Dynamic and Complex Systems

 Springer

Editor

Moamar Sayed-Mouchaweh
Institute Mines-Telecom Lille Douai
Douai, France

ISBN 978-3-319-74013-3 ISBN 978-3-319-74014-0 (eBook)
<https://doi.org/10.1007/978-3-319-74014-0>

Library of Congress Control Number: 2018933452

© Springer International Publishing AG, part of Springer Nature 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG part of Springer Nature.

The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Online fault diagnosis is crucial to ensure safe operation of complex dynamic systems in spite of faults affecting the system behaviors. Consequences of the occurrence of faults can be severe and result in human casualties, environmentally harmful emissions, high repair costs, and economic losses caused by unexpected stops in production lines. Therefore, early detection and isolation of faults is the key to maintaining system performance, ensuring system safety, and increasing system life. The majority of real systems are hybrid dynamic systems (HDS). In HDS, the dynamical behaviors evolve continuously with time according to the discrete mode (configuration) in which the system is. Consequently, model-based diagnostic approaches must take into account both discrete and continuous dynamics as well as the interactions between them in order to perform correct fault diagnosis. In addition, in HDS, two types of faults may occur: parametric and discrete faults. Parametric faults occur as abnormal changes in the value of parameters describing the continuous dynamics, while discrete faults are defined as unexpected, abnormal, changes in the system discrete mode.

A key challenge of fault diagnosis of HDS is related to the state estimation and tracking because of the cohabitation of continuous and discrete dynamics. Therefore, the fault diagnosis requires distinguishing between healthy and faulty states during mode changes for all hybrid trajectories generated by the system. However, tracking all the possible trajectories of a hybrid system is computationally intractable, in particular in the presence of faults. This is due to multiple reasons. Firstly, faults cause unknown changes in the system model. Thus, it becomes challenging to differentiate the change in behavior due to a fault from change in behavior caused by a normal mode transition. Secondly, pre-enumerating all the operation modes of a system is computationally intractable, in particular in the presence of faults. Indeed computing the reachable set of states of HDS is an undecidable problem due to the infinite state space of continuous systems.

Another challenge is related to the robustness of fault diagnosis and its time processing to issue the decision (fault detection and isolation). Indeed, the diagnosis engine must be able to manage out of order alarms and handle uncertainties and issue the diagnosis decision fast enough in order to give ample time to

human operators of supervision to implement corrective and maintenance actions. Finally, the diagnosis engine (inference) must scale well to large systems with multiple discrete modes. Indeed, a global model representing both the discrete and continuous dynamics can be too huge to be physically constructed for systems with large number of discrete modes.

This edited Springer book presents recent and advanced approaches and techniques that address the complex problem of fault diagnosis of hybrid dynamic and complex systems using different model-based and data-driven approaches in different application domains (inductor motors, chemical process formed by tanks, reactors and valves, ignition engine, sewer networks, mobile robots, planetary rover prototype etc.). These approaches cover the different aspects of performing single/multiple online/offline parametric/discrete abrupt/tear and wear fault diagnosis in incremental/nonincremental manner, using different modeling tools (hybrid automata, hybrid Petri nets, hybrid bond graphs, extended Kalman filter etc.) for different classes of hybrid dynamic and complex systems.

Finally, the editor is very grateful to all authors and reviewers for their very valuable contribution allowing setting another corner stone in the research and publication history of fault diagnosis of hybrid dynamic and complex systems. I would like also to acknowledge Mrs. Mary E. James for establishing the contract with Springer and supporting the editor in any organizational aspects. I hope that this volume will be a useful basis for further fruitful investigations and fresh ideas for researcher and engineers as well as a motivation and inspiration for newcomers to address the problems related to this very important and promising field of research.

Douai, France

Moamar Sayed-Mouchaweh

Contents

1	Prologue	1
	Moamar Sayed-Mouchaweh	
2	Motor Fault Detection and Diagnosis Based on a Meta-cognitive Random Vector Functional Link Network	15
	Choiru Za'in, Mahardhika Pratama, Mukesh Prasad, Deepak Puthal, Chee Peng Lim, and Manjeevan Seera	
3	Optimal Adaptive Threshold and Mode Fault Detection for Model-Based Fault Diagnosis of Hybrid Dynamical Systems	45
	Om Prakash, A. K. Samantaray, and R. Bhattacharyya	
4	Diagnosing Hybrid Dynamical Systems Using Max-Plus Algebraic Methods	79
	Gregory Provan	
5	Monitoring of Hybrid Dynamic Systems: Application to Chemical Process	101
	Nelly Olivier-Maget and Gilles Hêtreux	
6	Hybrid Bond-Graph Possible Conflicts for Hybrid Systems Fault Diagnosis	123
	Carlos J. Alonso-González, Belarmino Pulido, and Anibal Bregon	
7	Hybrid System Model Based Fault Diagnosis of Automotive Engines	153
	E. P. Nadeer, S. Mukhopadhyay, and A. Patra	
8	Diagnosis of Hybrid Systems Using Structural Model Decomposition	179
	Matthew J. Daigle, Anibal Bregon, and Indranil Roychoudhury	

9	Diagnosis of Hybrid Systems Using Hybrid Particle Petri Nets: Theory and Application on a Planetary Rover	209
	Quentin Gaudel, Elodie Chanthery, Pauline Ribot, and Matthew J. Daigle	
10	Diagnosis of Hybrid Dynamic Systems Based on the Behavior Automaton Abstraction	243
	Ramon Sarrate, Vicenç Puig, and Louise Travé-Massuyès	
	Index	279

Chapter 1

Prologue



Moamar Sayed-Mouchaweh

1.1 Hybrid Dynamic Systems: Definition, Classes, and Modeling Tools

Most of the real systems, e.g., vehicles, planes, power electronic devices, manufacturing systems, are hybrid dynamic systems (HDS) [1] in which the discrete and continuous dynamics cohabit. The discrete dynamics is described by discrete state variables while the continuous dynamics is described by continuous state variables. HDS exhibit different continuous dynamic behavior depending on the current operation mode q as follows:

$$\dot{X} = A_{(q)} X + B_{(q)} u$$

where X is the state vector and u is the input vector. In the case of linear systems, $A_{(q)}$ and $B_{(q)}$ are constant matrices of appropriate dimensions.

There are different classes of HDS, e.g., autonomous switching systems [2], discretely controlled switching systems [1], piecewise affine systems [3], discretely controlled jumping systems [4]. Many complex systems are embedded in the sense that they consist of a physical plant with a discrete controller. Therefore, the system has several discrete changes between different configuration modes through the actions of the controller exercised on the system plant (e.g., actuators). This kind of HDS is called discretely controlled continuous or switching systems (DCCS) [4]. Piecewise affine systems [3] are another important class of HDS where complex nonlinearities are substituted by a sequence of simpler piecewise linear behaviors.

M. Sayed-Mouchaweh (✉)
Institute Mines-Telecom Lille Douai, Douai, France
e-mail: moamar.sayed-mouchaweh@mines-douai.fr

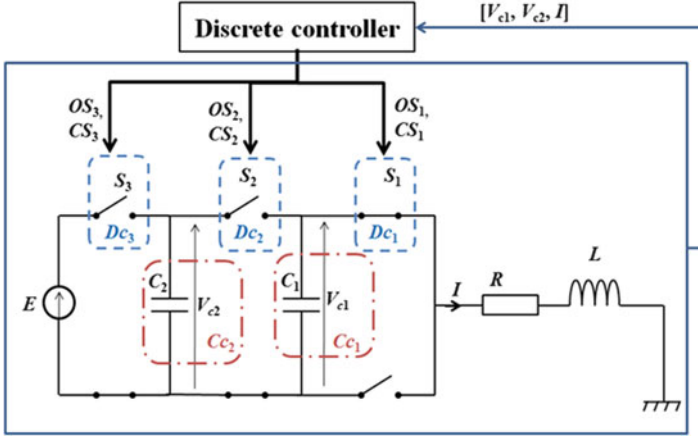


Fig. 1.1 Three-cell power converter as discretely controlled continuous system (DCCS) where capacitors C_1 and C_2 represent the continuous components (Cc) and switches S_1 , S_2 , and S_3 the discrete components (Dc)

The three-cellular power converters [5], depicted in Fig. 1.1, present an example of DCCS. The continuous dynamics of the system is described by state vector $X = [V_{C1} \ V_{C2} \ I]^T$, where V_{C1} and V_{C2} represent, respectively, the floating voltage of capacitors C_1 and C_2 and I represents the load current flowing from source E towards the load (R, L) through three elementary switching cells $S_j, j \in \{1, 2, 3\}$. The latter represent the system discrete dynamics. Each discrete switch S_j has two discrete states: S_j opened or S_j closed. The control of this system has two main tasks: (1) balancing the voltages between the switches and (2) regulating the load current to a desired value. To accomplish that, the controller changes the switches' states from opened to closed or from closed to opened by applying discrete commands “ CS_j ” or “ OS_j ” to each discrete switch $S_j, j \in \{1, 2, 3\}$ (see Fig. 1.1) where CS_j refers to “close switch S_j ” and OS_j to “open switch S_j .” Thus, the considered example is a DCCS.

There are three major modeling tools widely used in the literature to model HDS. These tools are hybrid Petri nets [6], hybrid bond graphs [7], and hybrid automata [8].

Hybrid Petri nets (HPN) model HDS by combining discrete and continuous parts. HPN is formally defined by the tuple:

$$\text{HPN} = \{P, T, h, \text{Pre}, \text{Post}\}$$

where $P = P_d \cup P_c$ is a finite, not empty, set of places partitioned into a set of discrete places P_d , represented as circles, and a set of continuous places P_c , represented as double circles. $T = T_d \cup T_c$ is a finite, not empty, set of discrete transitions T_d and a set of continuous transitions T_c represented as double boxes. $h: P \cap T \rightarrow \{D, C\}$, called “hybrid function,” indicates for every node whether it is a discrete node (D) or a continuous node (C). $\text{Pre}: P_c \times T \rightarrow \mathbb{R}^+$ or $\text{Pre}: P_d \times T \rightarrow \mathbb{N}$

is a function that defines an arc from a place to a transition. $\text{Post} : P_i x T_j \rightarrow \mathbb{R}^+$ or $\text{Post} : P_d x T \rightarrow \mathbb{N}$ is a function that defines an arc from a transition to a place.

Hybrid bond graph is a graphical description of physical dynamic systems with discontinuities. The latter represent the transitions between discrete modes. Similar to a regular bond graph, it is an energy-based technique. It is directed graphs defined by a set of summits and a set of edges. Summits represent components. The latter are: (1) passive components which transform energy into potential energy (C -components), inertia energy (L -components), and dissipated energy (T -components), (2) active components that can be source of effort or source pf flow. The edges, called bonds (drawn as half arrows), represent ideal energy connections between the components. The components interconnected by the edges construct the model of the global system. This model is represented by 1 junction for components having a common flow, 0 junction for common effort and transformers and gyrators to connect different kinds of energy. In order to take into account the information during the transitions between discrete modes, hybrid bond graph is extended by adding controlled junctions (CJs). The latter allow considering the local changes in individual component modes due to discrete transitions. The CJs may be switched ON (activated) or OFF (deactivated). An activated CJ behaves like a conventional bond graph junction. Deactivated CJs turn inactive the entire incident junction and hence do not influence any part of the system.

Hybrid automata are a mathematical model for HDS, which combines, in a single formalism, transitions for capturing discrete change with differential equations for capturing continuous dynamics. A hybrid automaton is a finite state machine with a finite set of continuous variables whose values are described by a set of ordinary differential equations. A hybrid automaton is defined by the tuple:

$$G = (Q, \Sigma, X, \text{flux}, \text{Init}, \delta)$$

where: Q is the set of states, Σ is the set of discrete events, X is a finite set of continuous variables describing the continuous dynamics of the system, $\text{flux} : Q \times X \rightarrow \mathbb{R}^n$ is a function characterizing the continuous dynamics of X in each state q of Q , $\text{Init} = (q \in Q, X(q), \text{flux}(q))$ is the set of initial conditions and $\delta : Q \times \Sigma \rightarrow Q$ is the state transition function. A transition $\delta(q, e) = q^+$ corresponds to a change from state q to state q^+ after the occurrence of discrete event $e \in \Sigma$.

1.2 Fault Diagnosis of Hybrid Dynamic Systems: Problem Formulation, Methods, and Challenges

A fault can be defined as a non-permitted deviation of at least one characteristic property of a system or one of its components from its normal or intended behavior. Fault diagnosis is the operation of detecting faults and determining possible candidates that explain their occurrence. Online fault diagnosis is crucial

to ensure safe operation of complex dynamic systems in spite of faults affecting the system behaviors. Consequences of the occurrence of faults can be severe and result in human casualties, environmentally harmful emissions, high repair costs, and economical losses caused by unexpected stops in production lines. Therefore, early detection and isolation of faults is the key to maintaining system performance, ensuring system safety, and increasing system life.

Faults may manifest in different parts of the system, namely, the actuators (loss of engine power, leakage in a cylinder, etc.), the system (e.g., leakage in the tank), the sensors (e.g., reduction of the displayed value relative to the true value, or the presence of a skew or increased noise preventing proper reading), and the controller (i.e., the controller does not respond properly to its inputs sensor reading). Faults can be abrupt (e.g., the failed-on or failed-off of the pump and the stuck opened or stuck closed of the valve), intermittent or gradual (degradation of a component). Faults also may occur in a single or a multiple scenario. In the former, one fault candidate explains the observations (is responsible for the fault behavior). In the latter, several fault candidates are responsible for the fault behavior.

In HDS, faults can occur as a change in the nominal values of parameters characterizing the continuous dynamics, and are called parametric faults. Faults can also occur in the form of abnormal or unpredicted mode-changing behavior and are called discrete faults. Therefore, two types of faults should be considered for HDS depending on the dynamics that is affected by faults (parametric or discrete). Discrete faults are related to faults in actuators and usually exhibit great discontinuities in system behavior, whilst parametric faults are related to tear and wear and introduce faults with much slower dynamics. For parametric faults, after the fault detection and isolation (determining the fault candidate), a fault identification phase is required in order to estimate the amplitude (e.g., the section of leakage of a tank) of the fault, its time of occurrence, its importance, etc.

For the example of three-cellular converters, eight faults can be considered for the diagnosis [4] as it is depicted in Fig. 1.2. Parametric faults (abnormal deviation of the nominal value of capacitors) are principally due to the effect of aging or pollution. The discrete faults (switch stuck-on or stuck-off) are more frequent and their consequences are more destructive. For instance, in open-circuit (stuck-off)

<i>Fault types</i>	<i>Fault labels</i>	<i>Fault event - Fault description</i>
Discrete faults	F_1	$f_{s1so} - S_1$ stuck opened
		$f_{s1sc} - S_1$ stuck closed
	F_2	$f_{s2so} - S_2$ stuck opened
		$f_{s2sc} - S_2$ stuck closed
	F_3	$f_{s3so} - S_3$ stuck opened
		$f_{s3sc} - S_3$ stuck closed
Parametric faults	F_4	f_{C_1} - Abnormal change in the nominal values of C_1 due to C_1 ageing
	F_5	f_{C_2} - Abnormal change in the nominal values of C_2 due to C_2 ageing

Fig. 1.2 Faults for the diagnosis of three-cell converters

failure, the system operates in degraded performance. However, unstable load may lead to further damage on the system. Therefore, the fault diagnosis of these faults is necessary to ensure the system safety and quality.

The fault diagnosis task [9, 10] is generally performed by reasoning over differences between desired or expected behavior, defined by a model, and observed behavior provided by sensors. This task can be performed offline or online. Offline diagnosis assumes that the system is not operating in normal conditions but it is in a test bed, i.e., ready to be tested for possible prior failures. The test is based on inputs, e.g. commands, and outputs, e.g. sensors readings, in order to observe a difference between the resulting signals with the ones obtained in normal conditions. In online diagnosis, the system is assumed to be operational and the diagnostic module is designed in order to continuously monitor the system behavior, isolate and identify failures. Within these methods, we can distinguish between active diagnosis that uses both inputs and outputs, and passive diagnosis that uses only system outputs. The diagnosis can also be non-incremental (i.e., the diagnosis inference engine is built offline) or incremental (the diagnosis inference engine is built online in response to the observation).

There are numerous methods in the literature that are used to perform fault diagnosis in HDS. They can be divided into internal, or model-based, and external, or data-driven, methods. The internal methods (see Fig. 1.3) use a mathematical or/and structural model to represent the relationships between measurable variables by exploiting the physical knowledge or/and experimental data about the system dynamics. They can be categorized into residual-based and set-membership [11] approaches. In residual-based approaches, the response of the mathematical model is compared to the observed values of variables in order to generate indicators used as a basis for the fault diagnosis. Generally, the model is used to estimate the system state, its output, or its parameters. The difference between the system and the model responses is monitored on the basis of residual generation. Then, the

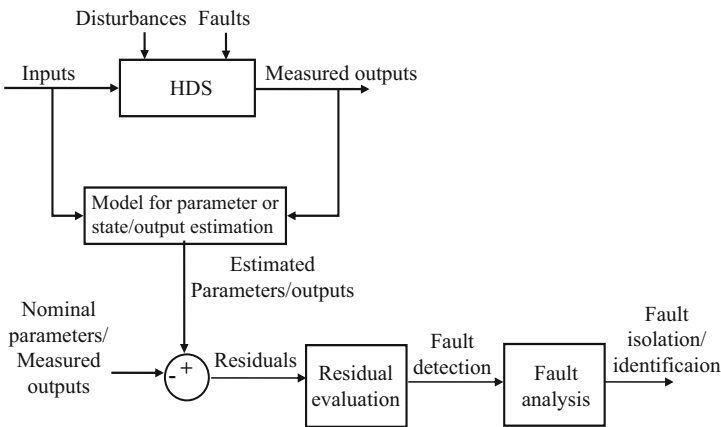


Fig. 1.3 Internal methods for fault diagnosis

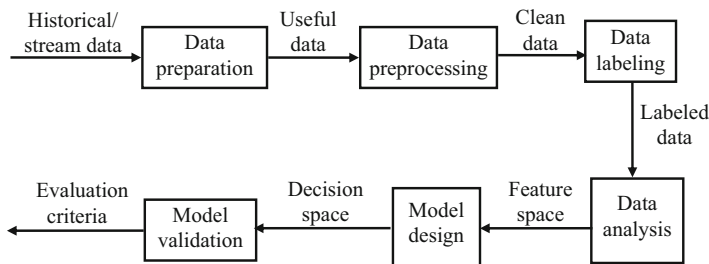


Fig. 1.4 External methods for fault diagnosis

trend analysis of this difference can be used to detect changing characteristics of the system resulting from a fault occurrence. Set-membership based fault diagnosis techniques are used for the detection of some specific faults. Generally, they discard models that are not compatible with observed data, in contrast to the residual-based approaches which identify the most likely model.

The external methods [3, 12–15] (see Fig. 1.4) consider the system as a black box, in other words, they do not need any mathematical model to describe the system dynamical behaviors. They use exclusively a set of measurements or/and heuristic knowledge about system dynamics to build a mapping from the measurement space into a decision space. They include expert systems and machine learning and data mining techniques.

A key challenge of fault diagnosis of HDS is related to the state estimation and tracking because of the cohabitation of continuous and discrete dynamics. Therefore, the fault diagnosis requires distinguishing between healthy and faulty states during mode changes for all hybrid trajectories generated by the system. However, tracking all the possible trajectories of a hybrid system is computationally intractable, in particular in the presence of faults. This is due to multiple reasons. Firstly, faults cause unknown changes in the system model. Thus, it becomes challenging to differentiate the change in behavior due to a fault from change in behavior caused by a normal mode transition. Secondly, pre-enumerating all the operation modes of a system is computationally intractable, in particular in the presence of faults. Indeed computing the reachable set of states of HDS is an undecidable problem due to the infinite state space of continuous systems.

Another challenge is related to the robustness of fault diagnosis and its time processing to issue the decision (fault detection and isolation). Indeed, the diagnosis engine must be able to manage out of order alarms and handle uncertainties and issue the diagnosis decision enough fast in order to give ample time to human operators of supervision to implement corrective and maintenance actions.

Finally, the diagnosis engine (inference) must scale well to large systems with multiple discrete modes. Indeed, a global model representing both the discrete and continuous dynamics can be too huge to be physically constructed for systems with large number of discrete modes.

1.3 Contents of the Book

This edited Springer book presents recent and advanced approaches and techniques that address the complex problem of fault diagnosis of hybrid dynamic and complex systems using different model-based and data-driven approaches in different application domains (inductor motors, chemical process formed by tanks, reactors, and valves, ignition engine, sewer networks, mobile robots, planetary rover prototype, etc.). These approaches cover the different aspects of performing single/multiple online/offline parametric/discrete abrupt/tear and wear fault diagnosis in incremental/non-incremental manner, using different modeling tools (hybrid automata, hybrid Petri nets, hybrid bond graphs, extended Kalman filter, etc.) for different classes of hybrid dynamic and complex systems.

1.3.1 Chapter 2

This chapter proposes a model built by learning from historical data (available sensor data) set in order to perform the fault diagnosis of induction motor. The considered faults are represented by five classes (normal, broken rotor bars, unbalanced voltages, stator winding faults, and eccentricity problems). The proposed approach exploits an evolving type-2 random vector functional link network (eT2RVFLN) since it allows dealing with four issues common in industrial processes: temporal system behavior, uncertainties in the data streams, a changing learning environment, and a large number of features. The proposed approach is split into three phases: (1) the what-to-learn phase allows to select a relevant subset of training samples, (2) the how-to-learn phase prunes and generates nodes in the hidden layer of the network, applies feature selection and parameter learning, and lastly (3) the when-to-learn phase assigns samples to the reserved sample set which are not considered for learning right away. The authors evaluated the method for diagnosis of induction motors and compared the approach to five other classifiers on two data sets (one with added noises and one without added noise). The obtained results show that the proposed approach outperforms the other methods in regard to their classification rate, learning time, and number of samples used during training.

1.3.2 Chapter 3

This chapter presents an approach to perform the fault detection and isolation of discrete and parametric faults when they may occur together in a single fault scenario. The proposed approach is based on the use of a hybrid bond graph that accounts for parameter uncertainties (multiplicative) and measurement noises (additive). The generated residuals from the hybrid bond graph are therefore robust against these

uncertainties and noises and are sensitive for both discrete and parametric faults. Indeed, these residuals take into account not only the magnitude of the deviation from the system's healthy conditions but also the direction (increase/decrease) of this deviation. The goal is to improve the fault isolation in particular when the behavior (fault signature) of parametric faults is similar to the discrete faults. In this case, the fault isolation is much complicated in comparison to continuous dynamic systems because the discrete mode fault may occur besides or together with a parametric fault in hybrid dynamic systems. Therefore, when a fault is detected, the cause of the residual inconsistency must be determined as a discrete mode fault or due to a parametric fault. In the former, a reconfiguration/shutdown procedure may be launched in response to the fault occurrence; while in the latter, parameter estimation is required to estimate the magnitude of the suspected parameters with known discrete mode information in order to calculate its severity. Then this information is provided to the decision-making system for fault accommodation. The discrimination between parametric faults and discrete mode faults is based on the magnitude of the residuals. The generated residuals are bounded within some time varying thresholds and then are used to generate fault signatures for different parametric or discrete mode faults. The proposed approach is applied for the fault diagnosis of discrete and parametric faults of a hybrid two-tank system using numerical simulation. The advantage of this method is its capacity to distinguish between discrete fault (e.g., valve stuck-on/valve stuck-off) from a parametric fault (valve partially stuck-on/valve partially stuck-off) based on the magnitudes of robust thresholds against parameter uncertainties and measurements noises. However, its computation complexity grows exponentially for large-scale systems with multiple discrete modes.

1.3.3 Chapter 4

This chapter presents an approach based on the use of max-plus algebra to perform the parametric and discrete faults of hybrid dynamic systems, in particular switching linear systems without concurrency. The latter are equivalent to continuous piecewise affine (PWA) systems where the state space is partitioned into a finite number of polyhedral regions. Each one of the latter is associated with different affine dynamics that the system switches between. The advantage of using max-plus algebra is its capacity to transform inference (that are nonlinear in a conventional algebra) on system timed dynamics to be linear in the max-plus algebra. The max function models the synchronization between events: an event occurs once all processes it depends on have finished. The $+$ function models the process times: the moment a process finishes must equal the sum of starting time and the time the process takes to finish. The presented approach considers the switching behaviors as stochastic in order to capture the stochastic nature of the faults. The proposed approach is based on three steps. The first step is based on the use of an observer in order to generate residuals that are used to detect fault occurrences. The second step looks to estimate the most likely discrete modes in order to calculate the correspond-

ing set of residuals. The third step aims at computing the most likely fault based on the optimization of a cost function and a set of residuals. The diagnosis inference is based on the time properties of the system: an event occurs too early, too late or does not occur at all. These properties are modeled using a timed event graph which is a subclass of time Petri nets. The proposed approach is illustrated and tested using an example of chemical process formed by two tanks, two reactors and seven valves. The evolution of the solvent level in the different tanks and reactors represents the continuous dynamics, while opening and closing the different valves represent the discrete dynamics (switching) of the system. The parametric faults are simulated as leakages in the tanks or reactors or a partial stuck-on/stuck-off of the valves. The discrete faults are represented by stuck-on/stuck-off faults. The main advantage of the proposed approach is the linear computational complexity of its diagnosis inference. However, its main drawback is its need to have information about each discrete and parametric fault behavior. This may impact the scalability of the proposed approach in the case of large-scale systems with multiple discrete modes.

1.3.4 Chapter 5

This chapter proposes a model-based approach in order to perform the simple and multiple fault diagnosis of hybrid dynamic systems. The proposed approach is based on three steps: residual generation, residual assessment, and fault localization. The residual generation is achieved offline and online. In order to generate the residuals offline, a hybrid dynamic simulator, called PrODHyS, is used to generate the reference model. The latter represents the normal, fault-free, system behavior. Then, faults are injected into the reference model in order to simulate the resulting fault behaviors. Then, residuals sensitive to each of these simple and multiple faults are generated and normalized in order to obtain the theoretical (simulated) fault signatures. The residuals are generated online by comparing the state of the reference model with the estimated one. The latter is obtained by using the extended Kalman filter allowing to improve the robustness of the monitoring against noises and uncertainties and therefore to avoid false alarms. The online generated residuals are normalized in order to obtain the instantaneous fault signature. Then, the distance between the latter and the offline fault signatures is calculated using an improved version of Manhattan distance. This distance provides an indication about the fault occurrence and its amplitude. The proposed approach was applied to perform simple and multiple faults detection and isolation using a complex chemical system composed by interconnected and shared resources (reactors/valves), in which a continuous treatment is carried out. The main advantage of the proposed approach is its capacity to diagnose simple and multiple faults as well as degradations in the system performance (quality). However, it needs to build a reference model about the system dynamics as well as to simulate the fault behaviors. This increases the computational complexity of the proposed approach for large-scale systems with multiple discrete models.

1.3.5 Chapter 6

This chapter presents a model-based diagnosis framework in order to diagnose both parametric and discrete faults of a class of hybrid dynamic systems (HDS). This class of HDS has continuous behavior controlled by discrete events. In those systems, the main source of hybrid behavior is discrete actuators, like valves or switches in fluid or electrical systems. The used model is hybrid bond graphs (HBGs) allowing providing a graphical description of the system's dynamics (links between its different variables). The diagnosis is performed based on the verification of the consistency (or conflicts) of a set of hypotheses represented by algebraic or differential equations. This verification allows generating fault hypotheses (residuals) from observed measurement deviations. Then, fault signatures are derived from residuals in order to isolate the fault source. This approach assumes that a discrete fault exhibits great discontinuities in system behavior, while parametric faults are related to tear and wear and introduce faults with much slower dynamics. Two examples are used to illustrate and evaluate the proposed approach. The first example is a simple electric circuit with two discrete switches, two batteries in parallel and a load of a resistance and a capacity. The second example is a hybrid four-tank system using on/off valves. The discrete faults are stuck-on/stuck-off of the actuators (switches/valves) and the parametric faults are related to the abnormal decrease/increase of the nominal value of resistance/capacity/fluid due to, as an example, the aging or pollution effect. The main advantage of the proposed approach is that the complete enumeration of the system operation modes (configurations) is not necessary and it can model nonlinear behavior. However, it requires the knowledge about the global model in order to build the algebraic or differential equations between the system variables in each configuration. This may be a handicap for the diagnosis of large-scale hybrid dynamic systems.

1.3.6 Chapter 7

This chapter proposes a model-based approach for online fault diagnosis for a spark ignition automotive engine. The forward and backward motions of fluids, the different strokes of the piston cycle, fuel injection, ignition and combustion, etc., the transient dynamics of the automotive engine is best modeled as a hybrid system with nonlinear continuous dynamics in its various discrete modes. The proposed fault diagnosis scheme starts with a continuous state estimation stage from inputs and measurements using a single extended Kalman filter (EKF) estimator. This is followed by a residual prediction stage in order to predict a residual vector corresponding to each fault hypothesized. Then, the hypothesis testing stage generates fault detection functions for each fault using the predicted and actual residuals. Each detection function is compared to its respective thresholds, which are experimentally decided based on simulations, followed by the isolation stage

that aims at isolating the fault source using predicate logic and knowledge of the process, under the assumption that at most a single fault could occur. Once the fault is isolated, the parameters (e.g., magnitude) of the isolated fault are identified using a joint estimation in the nominal EKF itself, or dual estimation, or some other separate estimators like particle filters. This approach is applied for the diagnosis of parametric faults of ignition engine such as a leak in the manifolds, injector block, cylinder valve wear, and sensor failures. The advantage of this approach is its computational efficiency allowing it to be used for on-board fault diagnosis implementations. However, it requires an additional computational computing in order to refine the set of isolated fault candidates. This may handicap its capacity to perform a precise fault diagnosis online.

1.3.7 Chapter 8

The chapter proposes a model-based, qualitative fault diagnosis framework for hybrid dynamic systems which can diagnose both parametric and discrete faults, and can handle observation delays. This approach is based on a structural model decomposition in order to decompose the model into independent submodels. The input and output variables are assigned locally to each submodel. Then, residuals are generated according to each submodel as well as the different switching modes. The residuals are transformed into qualitative 0 (no change), $-$ (decrease), and $+$ (increase) changes for the magnitude and slope in the residual. Once a residual is detected to deviate in a statistically significant manner from zero, symbols (0, $-$, $+$) are generated for that residual, and fed into the fault isolation module. Since local submodels are used, the residual generator will only contain the local modes of that submodel which is less than the number of system modes. Thus, fewer modes must be searched and the diagnosis efficiency is improved. The proposed approach is applied to two examples. The first example is an electric circuit that includes a voltage source, two capacitors, two inductors, two resistors and two switches, connected through a set of serial and parallel connections. Sensors measure the current or voltage in different locations. Each switch can be in one of two modes: on and off. Thus, this circuit can be represented as a hybrid system, with four system-level modes. The second example is the Advanced Diagnostics and Prognostics Testbed (ADAPT). It is an electrical power distribution system developed at NASA Ames Research Center and is used as a case study to demonstrate that the approach can correctly isolate faults in hybrid systems even if the system transitions among different modes and presents observation delays during the isolation process. It comprises a battery, circuit breakers, relays, AC to DC inverter, DC load and AC load. The structural model decomposition has the advantage to avoid the mode pre-enumeration problem and to facilitate the reusability of component models and their maintenance. In addition, it decreases the complexity of the hybrid system diagnosis problem since each submodel is dependent on only a subset of the system faults as well as a limited number of modes. However, the proposed approach is applied for

single fault scenarios and considers that all mode change events are observable. In addition, it assumes the feasibility of finding a structure decomposition allowing obtaining independent submodels.

1.3.8 Chapter 9

This chapter proposes a model-based scheme to perform the online fault diagnosis of hybrid dynamic systems. The proposed scheme is based on two phases: offline and online. The offline phase comprises two steps. The first step aims at building the model of the system using the hybrid particle Petri net (HPPN) framework. The latter is built either from a multimode description of the system or directly from expert knowledge. This model comprises the health modes of the hybrid system (nominal, degraded, and failure modes). They are represented by combinations of discrete states, continuous dynamics and degradation dynamics. Transitions aim at modeling the changes of health modes from one health mode (one for each type) to another health mode. The second offline step is the generation of the diagnoser based on the HPPN model. In the online phase, the built diagnoser is used to perform the diagnosis using the system consecutive observations (inputs and outputs). The proposed diagnosis scheme is evaluated and illustrated using two examples. The first example is a mobile robot with a motor commanded by on/off command. The second example is the K11 planetary rover prototype. The K11 is a testbed developed by NASA Ames Research Center to be used for diagnostics and prognostics purposes. It is powered by 24 2:2 Ah lithium-ion single cell batteries. The battery charge depletion, the motor overheating, the failed motor temperature sensors are examples of the faults diagnosed by the proposed diagnosis scheme. The advantage of the proposed diagnosis scheme is its capacity to take into account the uncertainty in the system representation. However, the model is built intuitively. This may handicap the flexibility of the proposed scheme for the diagnosis of complex hybrid systems in particular the large-scale ones with multiple discrete modes.

1.3.9 Chapter 10

This chapter focuses on the use of the hybrid automaton framework to develop a method for diagnosing both structural and non-structural faults in hybrid dynamic systems. Diagnosis is directly performed by interpreting the events and measurements issued by the physical system with respect to the hybrid automaton model. The discrete event part of the hybrid automaton constrains the possible transitions among modes and is referred to as the underlying discrete event system (DES). Then, the residuals capturing the consistency of the continuous dynamics are defined for each discrete mode. The abstraction of these residuals generates events, referred to as signature events, and is used to enrich the underlying DES. This enrichment

leads to obtain the so-called behavior automaton from which a diagnoser can be built. The latter can operate in a non-incremental and an incremental manner. In the non-incremental form, the diagnoser is built using the global model whereas in the incremental form only the useful parts of the diagnoser are built, developing the branches that are needed to explain the occurrence of incoming events. The proposed approach is evaluated using a representative part of the Barcelona sewer networks. The latter present several elements exhibiting numerous operating modes depending on the sewer flows. The used part of the sewer networks comprises nine virtual tanks, one real tank, three redirection gates, one retention gate, one four rain gauges to measure the rain intensity, and ten limimeters to measure the sewer level. The control gates are commanded by a controller where actions are open or close gate depending on the flow in the sewer. The structural faults are the stuck-on and stuck-off faults while the non-structural faults are the faults in sensors. The advantage of the proposed approach is its capacity to be used in incremental manner. This allows to obtain a significant gain in terms of memory storage compared to building offline the full diagnoser. However, the proposed approach is not adapted to scale with large-scale systems with multiple discrete modes.

References

1. Van Der Schaft, A. J., & Schumacher, J. M. (2000). *An introduction to hybrid dynamical systems* (Vol. 251). London: Springer.
2. Branicky, M. S., Borkar, V. S., & Mitter, S. K. (1998). A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1), 31–45.
3. Rodrigues, L., & Boyd, S. (2005). Piecewise-affine state feedback for piecewise-affine slab systems using convex optimization. *Systems & Control Letters*, 54(9), 835–853.
4. Louajri, H., & Sayed-Mouchaweh, M. (2014). Decentralized diagnosis and diagnosability of a class of hybrid dynamic systems. In *11th International conference on informatics in control, automation and robotics (ICINCO)* (Vol. 2).
5. Shahbazi, M., Jamshidpour, E., Poure, P., Saadate, S., & Zolghadri, M. R. (2013). Open-and short-circuit switch fault diagnosis for nonisolated dc–dc converters using field programmable gate array. *IEEE Transactions on Industrial Electronics*, 60(9), 4136–4146.
6. David, R., & Alla, H. (2010). *Discrete, continuous, and hybrid Petri nets*. Berlin Heidelberg: Springer.
7. Wang, D., Arogeti, S., Zhang, J. B., & Low, C. B. (2008). Monitoring ability analysis and qualitative fault diagnosis using hybrid bond graph. *IFAC Proceedings Volumes*, 41(2), 10516–10521.
8. Henzinger, T. A. (2000). The theory of hybrid automata. In *Verification of digital and hybrid systems* (pp. 265–292). Berlin Heidelberg: Springer.
9. Sayed-Mouchaweh, M., & Lughofer, E. (2015). Decentralized fault diagnosis approach without a global model for fault diagnosis of discrete event systems. *International Journal of Control*, 88(11), 2228–2241.
10. Sayed-Mouchaweh, M. (2014). *Discrete event systems: Diagnosis and diagnosability*. New York: Springer.
11. Tabatabaeipour, M., Odgaard, P. F., Bak, T., & Stoustrup, J. (2012). Fault detection of wind turbines with uncertain parameters: A set-membership approach. *Energies*, 5(7), 2424–2448.

12. Hartert, L., & Sayed-Mouchaweh, M. (2014). Dynamic supervised classification method for online monitoring in non-stationary environments. *Neurocomputing*, *126*, 118–131.
13. Sayed-Mouchaweh, M., & Messai, N. (2012). A clustering-based approach for the identification of a class of temporally switched linear systems. *Pattern Recognition Letters*, *33*(2), 144–151.
14. Toubakh, H., & Sayed-Mouchaweh, M. (2015). Hybrid dynamic data-driven approach for drift-like fault detection in wind turbines. *Evolving Systems*, *6*(2), 115–129.
15. Sayed-Mouchaweh, M. (2004). Diagnosis in real time for evolutionary processes in using pattern recognition and possibility theory. *International Journal of Computational Cognition*, *2*(1), 79–112.

Chapter 2

Motor Fault Detection and Diagnosis Based on a Meta-cognitive Random Vector Functional Link Network



Choiru Za'in, Mahardhika Pratama, Mukesh Prasad, Deepak Puthal,
Chee Peng Lim, and Manjeevan Seera

2.1 Introduction

2.1.1 Induction Motor

The induction motor, a type of electric motor, is widely used in industrial equipment such as manufacturing machines, belt conveyors, cranes, lifts, compressors, trolleys, electric vehicles, pumps, and fans [1]. Induction motors are complex devices comprising electromechanical components that can convert electrical power into mechanical movement. They use more than 60% of total electrical energy produced for use in industrial processes [2]. This is a result of their benefits compared with

C. Za'in

School of Engineering and Mathematical Science, La Trobe University, Melbourne,
VIC, Australia

M. Pratama (✉)

School of Computer Science and Engineering, Nanyang Technological University,
Singapore, Singapore
e-mail: mpratama@ntu.edu.sg

M. Prasad

Centre for Artificial Intelligence, School of Software, FEIT, University of Technology Sydney,
Ultimo, NSW, Australia

D. Puthal

School of Electrical and Data Engineering, FEIT, University of Technology Sydney,
Ultimo, NSW, Australia

C. P. Lim

Institute of Intelligent System Research and Innovation, Deakin University, Geelong,
VIC, Australia

M. Seera

Swinburne University of Technology, Kuching, Serawak, Malaysia

direct-current motors: they are more cost-efficient, more rugged, more reliable, and require less maintenance. In addition, induction motors have power ranging from hundreds of watts to megawatts, which is crucial for most industrial production.

To maintain the conditions of an induction motor and to avoid catastrophic failure of industrial processes, early diagnosis of any faults in an induction motor is essential. These faults can occur as a result of electrical, thermal, and mechanical stresses during motor operation. Sensors could be used to measure signals to detect and diagnose motor faults. These sensors, installed at the motor, might measure stator currents, voltage currents, air gaps, external magnetic flux densities, rotor position, rotor speed, output torque, internal temperature, external temperature, case vibrations, emissions, acoustic noise, electromagnetics, machine line, and case vibration.

During motor operation, sensors generate signals that continuously generate an unlimited total number of data. These signals can be analyzed to distinguish whether signals are generated from the normal operating condition or from some failure mode. In the case of a fault in induction motor operation, the values of variables change. These changes, which coincide with the event (fault), can be used to analyze the pattern of the system. Because values of induction motor system variables are generated in continuous mode and an event operates in discrete mode, an induction motor can be categorized as a hybrid system, wherein both continuous and discrete dynamics coexist and interact with each other. In this chapter, faults in an induction motor are detected and diagnosed to process both the discrete and the continuous dynamics of the system. As a result, advanced models/approaches are required to accurately describe the dynamic behavior of such a hybrid system and to detect and diagnose the faults.

A low-cost and efficient method is required to effectively detect a fault in an induction motor,. In terms of the components used in an induction motor, fault detection and diagnosis schemes have concentrated on three in particular: the stator, the rotor, and the bearing. Most recent research, however, has been directed toward the use of only a stator current because it provides the most accurate indicator of induction motor faults [3]. Motor current signature analysis (MCSA) is a common and popular method to detect faults in induction motors. This method is convenient because it uses information on only the stator currents, in comparison with the high costs of a continuous monitoring system applied in expensive or load-critical machines. The detailed method to detect and diagnose faults of induction motors is discussed in Sect. 2.2.

Furthermore, more comprehensive discussions with regard to the induction motor as a hybrid system and approaches used to analyze an induction motor's signals are described in Sects. 2.1.2 and 2.1.3, respectively.

2.1.2 Hybrid Dynamic System

A hybrid dynamic system, also known as a hybrid system, can be defined as a system driven by continuous and discrete dynamic interactions, emerging from a complex industrial system such as mechatronic system, manufacturing system, automotive engine control, or embedded control system. Hybrid systems are typically found in the components of an industrial system and operate over time within the industrial process.

Discretely controlled continuous systems (DCCSs), one class of hybrid system, are widespread among processes used in industrial plants. A DCCS involves continuous and discrete dynamics and continuous and discrete control, switching between several discrete modes in response to discrete control events at the discretion of a discrete controller. For example, a converter system, one example of a DCCS described by Toubakh and Sayed-Mouchaweh[4], is used to control the amount of power required by a generator. This is done by supplying the load current (I) by regulating the three elementary switches in the converter. In this system, the converter controls the interaction between continuous dynamics and a discrete event within the system. The system and its mechanisms are detailed as follows. The generator, called a doubly fed induction generator (DFIG), has two converters: grid side converters and rotor side converters. The first maintains the converter torque and the blade angle of a wind turbine given the wind speed, based on the controller's reference power. The second maintains the power by controlling the current supplied to the DFIG. Here we focus on the rotor side converter to investigate an element of the system called the multicellular converter (MCCS), which controls the supply of load current to the DFIG.

MCCS receives the reference output voltage (V_s) and the reference voltages of the output capacitors (V_c) from the controller. The continuous dynamic of the converter is described by a state variables vector: $X = [V_{c1}, V_{c2}, I]$, where V_{c1} and V_{c2} are the reference floating voltages of capacitors C_1 and C_2 , respectively, and I represents the current flowing toward the DFIG. The values of V_{c1} , V_{c2} , and I are adjusted given the formula [4] that represents continuous and discrete dynamic switching inside the system. This mechanism allows output to be generated in response to the controller's reference output voltage (V_s). As a result, an optimal amount of power is supplied to the DFIG. While the values of V_{c1} and V_{c2} represent the system's continuous dynamic, the value of I represents the system's discrete dynamic. The discretely controlled load current I is regulated by setting the three elementary switches S_j , j , and $\in \{1, 2, 3\}$ in the three-cell converter of the MCCS, where each discrete switch S_j has two discrete modes: open and closed. The dynamic evolution of the continuous and discrete variables of the converter systems are further investigated elsewhere [4].

Fault detection and diagnosis in a hybrid system is challenging because the continuous and discrete dynamics are mutually dependent and interact. Therefore, both their discrete and continuous dynamics must be considered. The fault of this converter system is considered to be related to the degradation of the MCCS's performance as a result of the chemical aging of its capacitors. This aging can be

recognized by monitoring the rise of the equivalent serial resistance value in the capacitor, which causes the output voltage (V_s) fed to the DFIG to drop.

According to a previous experiment [4], simple and multiple faults diagnosis of the converter capacitor are designed. Nine drift scenarios are conducted, with three drift speed variations applied in each capacitor to measure the change in the equivalent serial resistance value in order to observe faults in the MCCS's capacitor. Three classes are defined, representing the types of faults related to the MCCS capacitor: a simple fault in capacitor 1 (C_1), a simple fault in capacitor 2 (C_2), and multiple faults in both C_1 and C_2 . Six features are then derived from the residual value (difference of real voltage and reference voltage related to each capacitor [V_c]). Six feature spaces are generated from six discrete modes of the converters, where each discrete mode describes the sensitivity of features to some events/fault types, indicated by class label. In this experiment, MCCS faults are diagnosed by analyzing the pattern between features and class label with the use of an auto-adaptive dynamical clustering algorithm [5].

In general, faults in hybrid systems occur as a result of a gradual or abrupt change in the values of variables that describe the system's continuous dynamics in a discrete mode. These changes can be regarded as a drift, which can be observed in the system under operating conditions until the failure takes over completely. This drift can help detect faults early, before the system halts. In the case of the converter system described above, drift can be observed from the residual value (V_c). Because of the dynamic nature of the hybrid system, the drift of the variables' values can be observed only in discrete modes while the continuous dynamics described by the affected variables are active.

Because induction motors are categorized as hybrid systems, detecting and diagnosing faults in induction motors in real time is challenging. This is because of the dynamical behavior of induction motor operation, when signals are generated continuously. The system evolves over time, describing a new pattern of the system. Therefore, in a rapidly changing environment like that in an induction motor, advanced machine learning can be a solution to endow the classifier with adaptive capacity [6]. The next section describes our approach and the background of early machine learning approaches in detecting and diagnosing the faults of induction motors.

2.1.3 *Our Approach*

In the past two decades, many statistical and artificial intelligence techniques have been developed to detect and diagnose faults in induction motors; these include artificial neural networks (NNs) [7], fuzzy logic systems, genetic algorithms, and support vector machines. These techniques use features extracted from stator-current signals acquired during induction motor operation.

The vast majority of existing approaches characterize an offline learning principle that requires a complete data set covering all possible conditions during

the manufacturing operation. Furthermore, these approaches keep preceding data samples and require multiple passes over all data; therefore they are hardly scalable to be able to keep pace with the fast sampling rate of industrial processes. The stator current signals (signatures) are extracted using machine learning or statistical techniques to build an accurate model to predict motor conditions.

In an induction motor, signals acquired from sensors are continuously generated, with an unknown total number of data. This phenomenon is well known as a potential cause of “big data” with the famous “4 Vs” as characteristics [8, 9]: volume, velocity, variety, and veracity. Because of its iterative nature, this situation disqualifies conventional approaches and requires full access to all data before the process runs. This phenomenon requires a simple and fast learning algorithm to be practical for online real-time deployment. This issue [10] is also the underlying reason why randomized NNs (RNNs), which have their roots in the 1990s, have regained popularity. Some work has been done in this area to incorporate a random method into an NN system [11, 12].

However, the use of RNNs for online processing – as is the case in most industrial processes – deserves further investigation, at least for four issues: (1) The issue of structural complexity remains an open issue to be resolved, especially in the context of incremental learning. A hidden node pruning strategy has been designed [13]. Nevertheless, this approach undermines the logic of online learning, because they rely on a multistaged training mechanism. (2) Currently applied RNNs have not adequately addressed the issue of uncertainty, because most RNNs are built on the type 1 hidden nodes. Although the interval type 2 RNN has been proposed [14], this algorithm constitutes a batched learning machine, which acts offline. (3) Most RNNs adopt a static network topology that cannot adapt to changing learning environments. The notion of a dynamic RNN was devised by Feng et al. [15]. Nevertheless, the structural learning scenario does not reflect real data distribution, and therefore we are unable to discover the focal points of data streams. Another seminal work was proposed that makes use of the ensemble approach along with the drift detection method [16]. However, it possesses a demanding complexity because it involves a combination of several base classifiers. (4) To the best of our knowledge, RNNs still do not address the curse of dimensionality because of the absence of an online feature selection. It is worth noting that because of the intricacy of industrial process, RNNs often involve a large number of variables across several measurements.

To correct these drawbacks, a novel random vector functional link (RVFL) network, namely the evolving type 2 RVFL network (eT2RVFLN), has been proposed [17] to resolve four issues: uncertainty, concept drift, temporal system behavior, and high dimensionality. An eT2RVFLN applies the meta-cognitive learning principle, which consists of three phases: what to learn, how to learn, when to learn [18]. The cognitive part is constructed under an interval type 2 recurrent fuzzy NN. The what-to-learn phase applies a sample deletion mechanism, which selects important data streams for model updates in the how-to-learn phase. The how-to-learn phase actualizes the quickness of the RVFL learning theory combined with the evolving learning principle. This scenario allows hidden nodes to be automatically generated,

pruned, and recalled on the fly, with no tuning of hidden nodes while network parameters except output weights are randomly generated. The when-to-learn phase exploits a sample reserve strategy, which is useful to refine the network structure at the end of the training process. All of these attributes allow the algorithm to process the data stream, which has the 4 Vs as characteristics, online (to process every datum in a single-pass mode), simply (to reduce the number of training data thanks to the what-to-learn and how-to-learn methods), and quickly. Thus it able to cope with the scalability issue.

This chapter proposes a novel method to detect and diagnose induction motor faults online using the recently published eT2RVFLN. Real-world experiments using a laboratory-scale test rig were carried out to diagnose motor conditions, namely broken rotor bars, unbalanced voltages, stator winding faults, and eccentricity problems. This work applies multiclass classification, extending from a condition of only a two-class classification problem (healthy and faulty). It is worth noting that multifault diagnoses in induction motors were not applied in this experiment. However, some variables can be observed when some events occur. In this case, multiple continuous variables are observed and analyzed when induction motor operation experiences the multiple fault events, which can be assigned as new classes. The algorithm then builds the model from previous events in order to classify future data.

The eT2RVFLN was deployed to identify motor conditions in the online mode relying only on sensor data streams. Fault diagnosis processes were carried out in the main memory in only a one-pass learning scenario, with the absence of secondary memory or archival storage. Another unique feature of our approach is the three phases of meta-cognitive learning, namely what to learn, how to learn, and when to learn. This approach result in better memory efficiency and less tedious labelling effort than conventional online learners. Moreover, its interval type 2 hidden node has a better degree of tolerance against uncertainty than the standard type 1 hidden node and is appealing for most industrial applications because of noisy sensor data, noisy measurement, and false sensor readings. The eT2RVFLN characterizes a fully evolving characteristic whereby a hidden node can be generated, pruned, and recalled on the fly; it also is equipped with an online feature selection to cope with a high input dimension. Our rigorous experiments indicate that our approach delivered predictive accuracies comparable to those of its counterparts in both clean and noise-corrupted (20 dB) data. It also imposed lower complexities than other algorithms in terms of hidden node, network parameter, execution time, and input attribute. The meta-cognitive learning scenario contributed to bringing sample use and labelling effort to a low level, which is appealing in practice.

In the rest of this chapter, Sect. 2.2 outlines a literature survey of fault detection in induction motors; Sect. 2.3 discusses the details of the architecture of the eT2RVFLN algorithm; Sect. 2.4 covers the experimental design; Sect. 2.5 elaborates the experimental procedure; and Sect. 2.6 concludes the chapter.

2.2 Fault Detection and Diagnosis in Induction Motors

This chapter discusses background knowledge related to general methods to detect and diagnose faults in induction motors. Section 2.2.1 discusses the attributes related to fault detection and diagnosis in induction motors, which includes the variables, the type of faults, and the component(s) used to generate signals from an induction motor. Section 2.2.2 discusses the source of signals used to monitor an induction motor's condition.

2.2.1 *Fault Detection and Diagnosis Features in an Induction Motor*

The efficiency of induction motors comes from the variable speed that controls the speed and torque of the motor. However, it suffers from common mechanical and electrical problems. Unstable load changes and overloads are common causes of mechanical issues, whereas electrical issues occur because of an unstable power supply. This condition can lead to a catastrophic situation such as overheating, harmonics, and a shorter operational life of the motor, which damages the security of plant operations overall. Detection of an induction motor's fault early, before it occurs, is essential to reduce the costs for maintenance and for downtime within the overall production process.

In general, motor condition is monitored by sensors placed in the load-critical part of the machine; these sensors feed accurate and up-to-date data on the status of the motor to be monitored online [19]. They also measure stator currents, voltage currents, air gaps, external magnetic flux densities, rotor position, rotor speed, output torque, internal temperature, external temperature, and case vibrations, among other variables. These measurements are highly correlated to many types of motor faults: conductor shorts and openings, bearing failures, cooling failures, broken rotor bars, stator windings, eccentricity problems, and unbalanced voltages. Figure 2.1 describes physical and electrical measurements taken from induction motor operation to detect many failures in induction motor. Furthermore, Fig. 2.2 shows induction motor parts.

In this chapter, the signals used to monitor induction motor conditions are derived only from stator motor component measurements, as highlighted in Fig. 2.1. Because of the dynamical behavior of induction motors, the drift of variables are observed to analyze the pattern of hybrid system behavior based on the observed event (fault). To do this, some scenarios have been conducted to measure the signals in several motor conditions in order to learn the pattern of the system. Five events/fault types indicating a healthy motor, broken rotor bar problems, unbalanced voltages, stator windings problems, and eccentricity problems are observed as class labels. These labels represent the discrete mode of the induction motor condition.

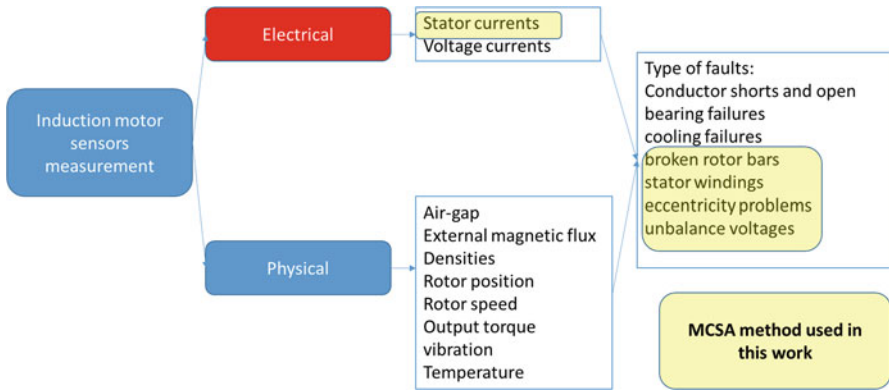


Fig. 2.1 Various induction motor measurements and its various types of faults

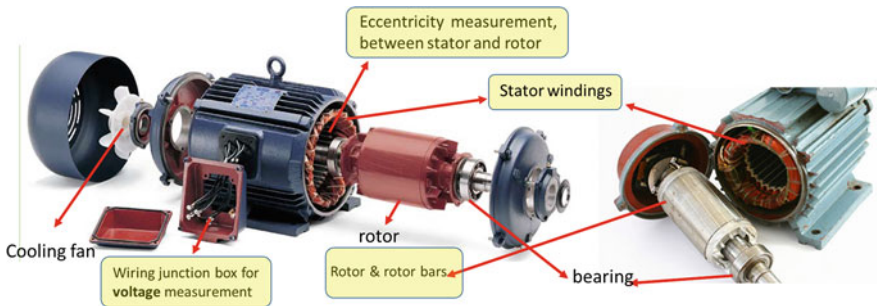


Fig. 2.2 Induction motor components and related measurements (source: <http://www.industrybuying.com>)

Some features/variables are suggested to be sensitive to several events [20]. This means that the values of features/variables drift when events/faults occur. These features are generated and derived from measurements of stator current operation acquired from sensors. The correlation between observed features related to the events/type of faults in induction motors is summarized in Table 2.1. It can be seen that an induction motor is categorized as a hybrid system, and its continuous dynamic values are described by variable drift, whereas “event” represents the discrete dynamic of the system. To learn the pattern of the system, some statistical learning or machine learning techniques can be used to identify the system pattern and decide which features are particularly sensitive to some events. Therefore, whenever feature/variable drift occurs, a degradation of the system can be recognized as an early fault of induction motors.

Table 2.1 Correlation of observed features related to the event/type of fault in induction motors

Causes	Features sensitive to the event	Event/type of fault in induction motors	Effect on the system
Severe vibration	1. Frequency in the stator current spectrum [21]	Broken rotor bars	A few broken rotor bars might not cause a drastic failure of the motor; however, they will cause motor-starting problems
Magnitudes of phase or line voltages are different	1. Residual voltage derived from phase voltage	Unbalance voltage	Shortens the life of a motor and reduces motor performance
Short circuit between a phase winding and the ground or between two phases due to turn-to-turn faults [2]	1. Spectrum of voltage in a search coil placed in the middle of the machine shaft [22]	Stator windings	Lead to a major short circuit and cause a destructive effect on the stator coils
Bearing faults	1. Frequency of eccentricity in the component [23] 2. Rotation frequency [24]	Eccentricity problems	Stator and rotor damage

2.2.2 *Fault Detection Methods from Single and Multiple Sources*

MCSA is widely known as an easy and effective method to detect and diagnose faults in induction motors. This is a nonintrusive method to detect faults by investigating electrical signals/signatures. MCSA extracts discriminative input features from signal harmonics as a discriminative data feature based on power spectral density (PSD). These features are used to feed a machine-learning algorithm featuring generalization capability, and then they detect and diagnose induction motor faults on the fly.

Many low-cost and nonintrusive methods to detect and diagnose faults have come into the picture [25–27]. For instance, fast Fourier transform (FFT) is used to analyze the signals of the motor's stator current. FFT detects and diagnoses many induction motor faults under various load conditions. Rule extraction techniques were used to detect and diagnose motor faults using pattern classification from numerical input-output data [28]. A neuro fuzzy system was used to extract the rules from unbalanced supply, unbalanced mechanical load, encoder, and voltmeter failure features [29]. Decision trees and an adaptive neuro-fuzzy inference system were also used to diagnose motor faults [30]. In other areas, a safety model based on fuzzy logic was also developed [31] for maritime and offshore safety purposes. This method provides information about the risk level of the system using Mamdani-type inference systems to assess consequences related to personnel, the organization, and the environment. Engine vibration has been analyzed using radial basis function [32].

Detection and diagnosis of faults in an induction motor can be categorized based the number of sources used to generate the data: a single source or multiple sources. By contrast, faults can be classified as a single fault or as multiple faults. Intelligent approaches have been developed in multiple-fault detection analysis. Detection of faults in rotor bars in induction motors has been proposed using back propagation for a feed-forward NN [26]. This method transfers current signals to magnitude using FFT. Feed-forward NN classifies sideband frequency magnitudes to detect broken rotor bars. A similar problem was also used by Sadeghian et al. [33] to detect broken rotor bars using wavelet packet decomposition, which analyzes signals and multiple frequency resolutions to be fed to a multilayer perceptron. Similar input from stator-current signals were used using multiple discriminant analysis, which classifies these signals to detect a broken rotor bar.

On the other hand, the same single source can be analyzed to diagnose multiple faults. The predictive filter method [25] predicts broken rotor bars and inter-turn stator winding faults by separating fundamental from harmonic components of current signals using a predictive filter and fuzzy model. Lee et al. [34] used Fourier and wavelet transforms to acquire the sideband and detail value characteristics from many motor conditions. A dynamic polynomial classifies the detail value characteristic of the motor into four categories: broken rotor bar, bowed rotor, faulted bearing, and eccentricity faults.

Liu et al. [35] proposed the detection of faults in induction motors using multiple process variables to detect broken rotor bars. They use a combination of fuzzy measures and fuzzy integrals to arrive at a single classification. Ondel et al. [36] processed the current voltage measurement to extract the most relevant diagnostic feature using k-nearest neighbors. The Kalman filter algorithm [36] was used to classify broken rotor bars afterward.

Multiple inputs and the detection and diagnosis of multiple faults in induction motors have been investigated in the current literature. Adjallah et al. [37] used two sources of input (three-phase stator currents and vibration signals). A genetic algorithm was then used to reduce the input dimension and training of neural network. The next step, adaptive resonance theory, was used to detect unbalanced phase and eccentricity problems of induction motors. Rotor faults, eccentricity, and bearing faults from vibration and current signal have been predicted [30]. Features have been selected using a decision tree that produces a crisp rule. The crisp rules acquired from the decision tree were converted to fuzzy if-then rules to identify the structure of an adaptive network-based fuzzy inference system [38].

2.3 eT2RVFLN Architecture

This section discusses the architecture of an eT2RVFLN, which exhibits three pillars of meta-cognitive learning policies: what to learn, how to learn, and when to learn. The what-to-learn phase selects training samples using a certainty-based learning method. The how-to-learn phase of eT2RVFLN is where the cognitive components are adjusted (learning module), whereas the when-to-learn phase controls when the sample reserves are mined after all main training samples have been learned. All learning policies of eT2RVFLN are discussed in Sects. 2.3.1 and 2.3.2, which describe the what-to-learn, how-to-learn, and when-to-learn phases, respectively.

2.3.1 Cognitive Architecture of an eT2RVFLN

Multivariate Gaussian functions in the hidden layer and a nonlinear Chebyshev polynomial form of a generalized interval type-2 functional NN topology put forward the generalized interval type 2 Gaussian fuzzy rule. Type 2 fuzzy sets, which were proposed by Zadeh, are extensions of ordinary fuzzy sets (namely type 1) to handle uncertainty. The eT2RVFLN's fuzzy rule is defined as follows:

$$R_i : \text{IF } X \text{ is close to } \tilde{R}_i \text{ Then } y_i^o = x_e \Omega_i,$$

where $\tilde{R}_i = [R_i, \bar{R}_i]$ represents an interval type 2 multidimensional kernel compiled from a multidimensional kernel with uncertain means. y_i^o Denotes the output of

the i th rule in the o th class, and $\Omega_i = \mathfrak{R}^{(2u+1) \times m}$ represents the output weight vector. $x_e \in \mathfrak{R}^{(2u+1) \times m}$ Stands for the expanded input vector. The interval-valued multivariate Gaussian function is mathematically defined as follows:

$$\tilde{R}_i = \exp \left(- \left(X_n - \tilde{C}_i \right) \Sigma^{-1} \left(X_n - \tilde{C}_i \right)^T \right) \tilde{C}_i = [C_{i,1}, C_{i,2}] \quad (2.1)$$

where $\tilde{C}_i \in \mathfrak{R}^{1 \times u}$ denotes uncertain centroids of the Gaussian function, where the upper centroid $C_{i,2}$ is set to be larger than the lower centroid $C_{i,1}$ to form the footprint of uncertainty and the number of hidden nodes is represented by u . $\Sigma^{-1} \in \mathfrak{R}^{u \times u}$ Labels a nondiagonal inverse covariance matrix, which triggers non-axis parallel ellipsoidal clusters arbitrarily rotated in any direction. Thus, this non-axis parallel ellipsoidal cluster is capable of constructing a more reliable input space partition.

However, the multivariate Gaussian function works in high-dimensional space and is not able to be projected directly into single-dimensional space. Therefore, it is not directly compatible with the interval type 2 fuzzy inference process. The transformation strategy [27] is applied to construct a lower-dimensional representation of the multivariate Gaussian function. This strategy is selected because it offers a fast mechanism to elicit a radius of a neuron from a non-axis parallel ellipsoidal cluster, although it leads to too-small radii when processing an ellipsoidal cluster being rotated 45° . As an alternative, the radii can be derived from the eigenvalue and eigenvector of the covariance matrix, but this scenario imposes a higher complexity because the eigenvalue and eigenvector must be elicited in every observation. The radius of the non-axis parallel ellipsoidal cluster is written as follows:

$$\sigma_i = \frac{r}{\sqrt{\Sigma_{ii}}}, \quad (2.2)$$

where Σ_{ii} denotes the diagonal element of the multivariate Gaussian function in the i th coordinate. The Mahalanobis distance between the datum and the distribution's i th cluster is denoted by r . The Mahalanobis distance is a multidimensional generalization measuring how many standard deviations of data form the mean of the i th cluster. On the other hand, no transformation takes place at the neuron level because the ellipsoidal cluster's centroid is directly compatible without modification.

The expanded input vector $x_e \in \mathfrak{R}^{1 \times (2u+1)}$ is used to improve the local mapping ability of standard Takagi Sugeno Kang (TSK) output node using a nonlinear mapping of the Chebyshev polynomial inspired by the expansion layer of the functional link NN [39].

Standard TSK rule has two parts: the rule antecedent and the rule consequent. It has two different types: zero order and first order. The zero-order TSK model is a fuzzy system whose rule in the consequent part is constant, whereas the first-order TSK model is a fuzzy system whose rule in the consequent part is a linear function. It is worth noting that the zero-order or first-order TSK rule consequent does not

exemplify a proper local approximation trait because it features a low degree of freedom (a linear hyperplane in the output space). A nonlinear mapping of the Chebyshev polynomial is written as follows:

$$T_{n+1}(x) = 2x_j T_n(x_j) - T_{n-1}(x_j), \quad (2.3)$$

where $T_0(x_j) = 1$, $T_1(x_j) = x_j$, $T_2(x_j) = 2x_j^2 - 1$. For example, given that we have a two-dimensional input pattern $X = [x_1, x_2]$, the expanded input vector is generated as $x_e = [1, x_1, T_2(x_1), x_2, T_2(x_2)]$. It is worth noting that term 1 presents the output node's intercept to circumvent all local submodels from going through the origin, which may result in nontypical gradients.

The Gaussian function with a fixed standard deviation and uncertain means $\tilde{C}_i = [C_j^i, 1, C_j^i, 2]$ is used to express an uncertain degree of membership, expressed as follows:

$$\tilde{\mu}_{i,j} = \exp\left(-\left(\frac{x_j - \tilde{c}_{i,j}}{\sigma_{i,j}}\right)^2\right), \quad \tilde{c}_i = [C_j^i, 1, C_j^i, 2] \quad (2.4)$$

$$\bar{\mu}_{i,j} = \begin{cases} N(c_{j,1}^i, \sigma_{i,j}; x_j) & x_j < c_{j,1}^i \\ 1, c_{j,1}^i \leq x_j < c_{j,2}^i \\ N(c_{j,2}^i, \sigma_{i,j}; x_j) & x_j > c_{j,2}^i \end{cases} \quad (2.5)$$

$$\underline{\mu}_{i,j} = \begin{cases} N(c_{j,2}^i, \sigma_{i,j}; x_j) & x_j \leq \frac{(c_{j,1}^i + c_{j,2}^i)}{2} \\ N(c_{j,1}^i, \sigma_{i,j}; x_j) & x_j > \frac{(c_{j,1}^i + c_{j,2}^i)}{2} \end{cases} \quad (2.6)$$

The t-norm operator generates the interval firing strength $\tilde{R}_i = [\underline{R}_i, \bar{R}_i]$, as follows:

$$\underline{R}_i = \prod_{j=1}^u \underline{\mu}_{i,j}, \quad \bar{R}_i = \prod_{j=1}^u \bar{\mu}_{i,j}. \quad (2.7)$$

A type of reduced set expressing the crisp's output from the interval's set is usually produced by a type reduction mechanism using the K-M iterative procedure. This scenario, however, imposes a high computational cost as a result of the requirement for a multipass iterative computation to obtain the lower (L) and upper (R) end points. Therefore, the $q \in \mathfrak{R}^{1 \times m}$ design coefficient is used to control the proportion of R and L nodes. The design coefficient is written as follows:

$$\begin{aligned}
y_o &= \frac{(1 - q_o) \sum_{i=1}^P \underline{R}_i y_{i,o} + q_o \sum_{i=1}^P \overline{R}_i y_{i,o}}{\sum_{i=1}^P \underline{R}_i + \overline{R}_i} \\
&= \frac{(1 - q_o) \sum_{i=1}^P \underline{R}_i x_e \Omega_{i,o} + q_o \sum_{i=1}^P \overline{R}_i x_e \Omega_{i,o}}{\sum_{i=1}^P \underline{R}_i + \overline{R}_i},
\end{aligned} \tag{2.8}$$

where P is the number of hidden nodes. The design factor $q \in \mathfrak{R}^{1 \times m}$ is randomly generated without being adapted in the training process to adopt the spirit of RVFL network. The classification decision is generated by taking a maximum operator across all output nodes, as follows:

$$y = \max_{o=1, \dots, m} \hat{y}_o \tag{2.9}$$

where m is the number of class labels. An overview of the learning architecture of the eT2RVFLN is shown in Fig. 2.3.

2.3.2 Meta-cognitive Learning Policy of the eT2RVFLN

This section outlines the meta-cognitive part of the eT2RVFLN. Section 2.3.2.1 elaborates the what-to-learn part, which actively selects relevant samples for the training process. The training pattern extracted from the what-to-learn method becomes the input for the how-to-learn part that controls the evolution of cognitive component, which is elaborated in Sect. 2.3.2.2. The training samples that do not meet the learning criteria of the how-to-learn phase are stored as reserved samples. These samples are to be used after the main (selected) training samples have been fully learned. The when-to-learn aspect is elaborated in Sect. 2.3.2.3. The pseudocode of the eT2RVFLN learning policy is detailed in Fig. 2.4.

2.3.2.1 What to Learn

The what-to-learn part of an eT2RVFLN is controlled by a certainty-based active learning scenario, which has a capability to extract relevant samples. This module also affects to reduction of labelling efforts, because not all samples trigger the annotation effort of the operators. This mechanism can be envisaged in strict sense as a semisupervised mechanism. This method extends the what-to-learn of meta-cognitive classifiers [18, 40], which happen to be a fully supervised mechanism. Such a mechanism has no effect on labelling effort, although the number of training samples can be reduced because the contributions of data streams is evaluated with the presence of the ground truth of the class label. In the evolving systems domain, active learning has been studied [41], but the studies still have

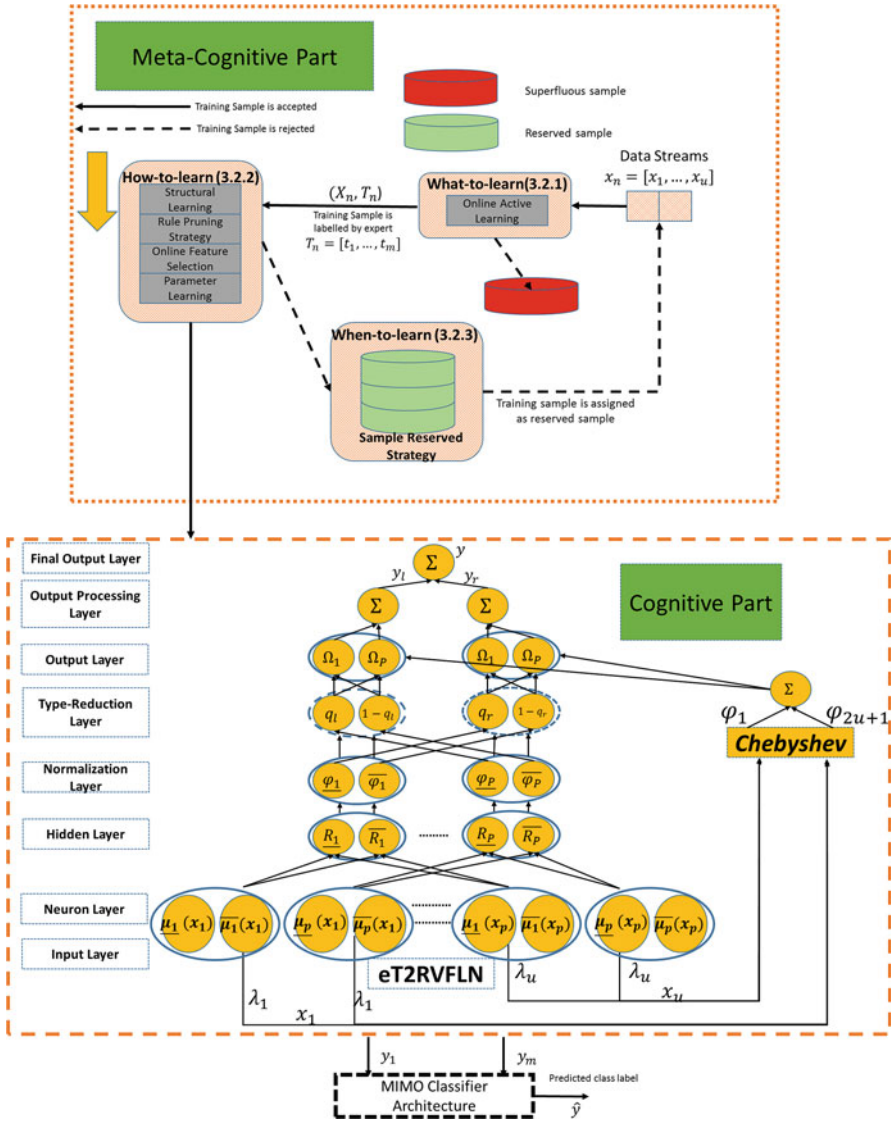


Fig. 2.3 Learning architecture of the eT2RVFLN [17]

not taken into account possible concept changes in data distribution. It is evident that labelling effort increases dramatically in the presence of concept drift because this situation diminishes the degree of confidence of the model. We put forward a Bayesian concept to perform the certainty-based active learning scenario in the input and output space, where the Bayesian posterior probability quantifies the conflict incurred by data streams. A substantial conflict in the output space happens when the

```

Define: Training Data  $(X_n, T_n) = (x_1, \dots, x_u, t_1, \dots, t_m)$ 
Predefined Parameters  $\Phi = 10^5, B = 0.2, s = 0.05, \varpi = 10^{-15}$ 
/*Phase 1: What to Learn Strategy: Active Learning Strategy*/
For  $o=1$  to  $m$  Do
  For  $i=1$  to  $P$  Do
    Compute the output of classifiers (8) and the posterior
    probability(10),(11)
  End For
End For
IF (22) Then
Label the data stream using expert knowledge
End IF
/*Phase 2: Hidden Node Growing Scenario*/
For  $i=1$  to  $P$  Do
Compute the global density (15)
End For
Determine the winning rule  $win = \arg \max_{i=1, \dots, P} \hat{P}(R_i|X)$ 
IF (16) Then
  Add the hidden node(17),(18)
Else IF (27)
  Assign the datum as reserved sample  $(X_{NS+1}, TS_{NS+1})=(X_n, T_n)$ 
End IF
/*Phase 3: Hidden node Pruning Mechanism*/
For  $i=1$  to  $P$  Do
Compute RMI(19)
  IF (20) Then
    Prune  $i$ -th hidden node
  End IF
End For

/*Phase 4: Feature Selection Mechanism*/
For  $j=1$  to  $u$  Do
  For  $o=1$  to  $m$  Do
    Compute the symmetrical uncertainty  $SU(x_j, t_o)$ 
    IF  $SU(x_j, t_o) < \gamma$  Then
      Prune the  $j$ -th feature
    Else IF
      Append  $j$ -th as relevant variables  $x_{r_e}, N_{r_e} = N_{r_e} + 1$ 
    End IF
  End For
End For
For  $j=1$  to  $N_{r_e}$ 
  For  $re=1$  to  $N_{r_e}, j \neq re$ 
    Analyze the redundancy of input attributes  $SU(x_j, x_{re})$ 
    IF  $SU(x_j, T) < SU(x_j, x_{re})$  Then
      Prune  $re$ -th input variable
    End IF
  End For
End For
/*Phase 5: Adaptation of local sub-model*/
For  $i=1$  to  $P$  Do
Adjust the local sub-model (22)-(26)
End For
/*Phase 6: When-to-learn-sample reserved strategy*/
IF (27)
For  $i=1$  to  $NS$  Do
Execute phase 1-6 using reserved sample  $(X_{NS}, TS_N)$ 
End For
End IF

```

Fig. 2.4 Learning policy of eT2RVFLN

datum lies in an adjacent proximity to the decision boundary, whereas the conflict is apparent in the input space when a datum lies on a class-overlapping region as a result of an unclear cluster.

In the output space, the conflict is investigated by the classifier's truncated output as follows:

$$p(\hat{y}_o|X)^{\text{output}} = \min(\max(\text{conf}_{\text{final}}, 0), 1), \text{conf}_{\text{final}} = \frac{\hat{y}_1}{\hat{y}_1 + \hat{y}_2}, \quad (2.10)$$

where $p(\hat{y}_o|X)^{\text{output}}$ defines the output posterior probability. The most dominant and second most dominant outputs are depicted by \hat{y}_1 and \hat{y}_2 , respectively. The conflict in the output space is related to the quality of the decision boundary to handle the overlapping region. In the input space, a posterior probability is estimated using joint-category and class probability $p(\hat{y}_o|R_i)$, as follows:

$$P(\hat{y}_o|R_i) = \left(\frac{\sum_{i=1}^P (\hat{y}_o|R_i) P(X | R_i) P(R_i)}{\sum_{o=1}^m \sum_{i=1}^P (\hat{y}_o|R_i) P(X | R_i) P(R_i)} + \frac{\sum_{i=1}^P (\hat{y}_o|R_i) P(X |\bar{R}_i) P(R_i)}{\sum_{o=1}^m \sum_{i=1}^P (\hat{y}_o|R_i) P(X |\bar{R}_i) P(R_i)} \right) \quad (2.11)$$

$$P(\hat{y}_o|R_i) = \frac{\log(N_i^o + 1)}{\sum_{o=1}^m \log(N_i^o + 1)}, \quad (2.12)$$

where N_i^o represents the number of populations of the i th cluster falling into the o th class, and $p(\hat{y}_o|X)^{\text{input}}$ is the Bayesian class posterior probability in the input space. This measures the degree of class overlap as a result of the use of the class posterior probability $P(\hat{y}_o|R_i)$ in Eq. (2.11). A strong confusion is indicated, as follows:

$$p(\hat{y}|X)^{\text{output}} < \theta \text{ or } p(\hat{y}|X)^{\text{input}} < \theta, \quad (2.13)$$

where θ represents the conflict threshold and is initialized as $\theta = \frac{1}{m} + 0.2(\frac{1}{m})$ under the assumption that data are uniformly distributed. This condition triggers the annotation effort of the operator and in turn the training process, because an operator's feedback is required in this condition to clear up the strong confusion. The conflict threshold is adjusted adaptively to be well suited to dynamic data streams $\theta_{N+1} = \theta_N(1 \pm s)$; it increases whenever a sample is accepted for a training process, whereas it decreases whenever a sample is ruled out from a training process. Furthermore, a budget B is set during a training process and depicts the maximum number of labelling processes in the training process. The training process is terminated and a model is fixed when a budget has finished.

2.3.2.2 How to Learn

The how-to-learn mechanism updates the cognitive component using the passage of a sample from the what-to-learn mechanism. This mechanism is constructed by four learning strategies: hidden node growing mechanism, hidden node pruning mechanism, feature selection mechanism, and parameter learning mechanism.

2.3.2.2.1 Hidden Node Growing Mechanism

This mechanism is equipped with a knowledge-exploratory module that automatically controls the hidden node growth. The new hidden node grows when a sample incurs sufficient novelty that allows the knowledge base to expand or when it occupies the most populated region in the input space. The novelty of the datum can be calculated by the firing strength in the interval type 2 functional NN because it reflects the degree of compatibility with the current network structure:

$$\text{FS}_{P+1} = \frac{R_{P+1} + \bar{R}_{P+1}}{2}, \quad (2.14)$$

where $\underline{R}_i, \bar{R}_i$ denote the lower and upper lower firing strengths, as shown in Eq. (2.6). Nevertheless, the shortcoming of this method is that the outliers are prone to sitting far away from the zone of influence of the existing hidden nodes because the firing strength results from a point-to-point calculation without considering the overall data distribution. To remedy this drawback, the conflict measure should

encompass information of all historical data without formally keeping them in the memory, which is impractical in the online learning setting. The firing strength of the hypothetical rule ($P + 1$) for all samples is calculated as follows:

$$FS_{P+1} = \frac{\sum_{n=1}^N R_{p+1} + \bar{R}_{p+1}}{2N}, \quad (2.15)$$

where N denotes the number of training sample observations. Eq. (2.15) triggers the hypothetical cluster [42], which describes a focal point of training data. The inverse quadratic function, a Gaussian-like function that enables global density, is used to quantify sequentially. This function is used because of its capability to support a recursive operation in Eq. (2.15), which could not be supported by a Gaussian function.

A new hidden node is introduced if the following criterion is met:

$$FS_{P+1} > \max_{i=1,\dots,P} (FS_i) \text{ or } FS_{P+1} < \min_{i=1,\dots,P} (FS_i). \quad (2.16)$$

The first term of Eq. (2.16), $FS_{P+1} > \max_{i=1,\dots,P} (FS_i)$, shows that the newly created hidden node lies in a dense region surrounded by most of the other samples. Such a data point brings advantage to the summarization power of the network structure. On the other hand, the second term of Eq. (2.16), $FS_{P+1} < \min_{i=1,\dots,P} (FS_i)$, depicts an input region beyond the coverage of the existing topology. This situation can be regarded as a precursor to changing learning environments because a new sample has a very low density. Thus, a datum should be crafted as a new hidden node to improve the generalization potential of the network structure. The parameters of a newly added neuron are set as follows:

$$\tilde{C}_{P+1} = X_N \pm \Delta X, \sum_{P+1}^{-1} = \text{rand}(A), A \in \mathfrak{R}^{u \times u}, \quad (2.17)$$

where the new rule's centroid is set as a new data sample $\bar{C}_{P+1} = X_N + \Delta X$, $\underline{C}_{P+1} = X_N - \Delta X$, where ΔX is the uncertainty factor to form the region of uncertainty. $A \in \mathfrak{R}^{u \times u}$ denotes the inverse covariance matrix, randomly generated as with the RVFL network theory. The eT2RVFLN's hidden node growing mechanism is different from the drift detection method [16] because it still retains a single-model architecture that demands fewer complexities than that of the ensemble classifier. Furthermore, the hidden node growing mechanism is parameter free and thus does not depend on a problem-specific threshold.

The new local submodel is set as the local submodel of the winning hidden node to accelerate the training process, because it is supposed to have an adjacent relationship with the new node. A large, positive, definite covariance matrix is generated to emulate the real solution provided by the batched learning scenario [43].

$$\Omega_{P+1} = \Omega_{\text{win}}, \Psi_{P+1} = \omega I, \quad (2.18)$$

where ω defines a large positive constant, set as $\Theta = 10^5$, and Ψ_{p+1} is the new rule's output covariance matrix. The local learning principle is applied in the eT2RVFLN to adjust the local subsystems, each of which is loosely coupled and has a unique covariance matrix. Therefore, the growth, pruning, and adaptation of a particular node weakly influences the convergence of other hidden nodes. Commitment of the structural learning mechanism does not require a special setting of a global covariance matrix.

The Bayesian concept is used to select the winning hidden node, possessing the maximum posterior probability as $\text{win} = \arg \max_{i=1, \dots, P} \hat{P}(R_i|X)$. The key characteristic of the Bayesian concept in obtaining the winning hidden node is its prior probability, where the winning hidden node is extracted in probabilistic fashion. This is practical to determine the winning hidden node when candidates lie at similar distances from the sample of interest.

2.3.2.2.2 Hidden Node Pruning Mechanism

The rule-pruning mechanism, namely type 2 relative mutual information (T2RMI), is adopted in the eT2RVFLN to reduce structural complexity by removing inconsequential hidden nodes. This mechanism is based on a modified version of the relative mutual information (RMI) work [44] to handle an interval type 2 hidden node.

T2RMI is used to recognize mutual information between the hidden node and the class label. Even though the correlation between a neuron and the target variable can be measured by both linear and nonlinear approaches, the nonlinear measure is used in T2RMI because the interaction between two variables is often nonlinear and cannot be accurately quantified by a linear measure [45]. This correlation is computed by the symmetrical uncertainty method and carries at least three benefits: (1) simplicity, (2) low bias for multivalued features, and (3) insensitivity to the order of two variables [46].

The T2RMI method is defined as follows:

$$\text{RMI}(\tilde{R}, Y) = \left(\frac{(1-q) 2I(\underline{R}_i, Y)}{H(\underline{R}_i) + H(Y)} + \frac{q 2I(\overline{R}_i, Y)}{H(\overline{R}_i) + H(Y)} \right), \quad (2.19)$$

where $I(\overline{R}_i, Y) = H(\overline{R}_i) + H(Y) - H(\overline{R}_i, Y)$ is the information gain that represents the mutual information between the hidden node and the class label. Entropy of (\overline{R}_i) is denoted as $H(\overline{R}_i)$, whereas the joint entropy of \overline{R}_i, Y is denoted as $H(\overline{R}_i, Y)$. This symmetrical uncertainty computation hovers around $[0, 1]$, which shows dependence between the \overline{R}_i, Y values, where a value of 0 means that \overline{R}_i, Y are uncorrelated.

The hidden node is discarded if the following criterion is met:

$$\text{RMI} < \text{mean}(\text{RMI}) - 2\text{std}(\text{RMI}), \quad (2.20)$$

where $\text{mean}(\text{RMI})$ is the empirical mean of RMI and $\text{std}(\text{RMI})$ is the standard deviation of RMI.

2.3.2.2.3 Online Feature Selection Mechanism

In the concept of online learning, feature selection during preprocessing becomes unfeasible because of the nature of online learning, which demands a fast and autonomous process. In eT2RVFLN, an incremental input pruning mechanism is applied to remove the inconsequential input attributes during the training process without significantly reducing the generalization process. The incremental input pruning strategy is built on the analysis of relevance and redundancy based on the Markov blanket theory [47]. This strategy ensures that the input attribute will never become important in the future when an input feature carries a Markov blanket to another input attribute. In other words, it remedies the instability issue in traditional input pruning methods, which happens as a result of discontinuity. In general, the input attribute can be categorized by four criteria: irrelevant, weakly relevant, weakly relevant but nonredundant, and strongly relevant.

The selection process is initialized by measuring the relevance of features by examining the correlation between input features and the target class. The next phase is to measure the feature redundancy. These two correlation measures, namely C-correlation and F-correlation (Eq. [2.19]) are formulated as described just below.

Definition 1 (C-Correlation) [47] Feature x_j and class T have a correlation termed *C-correlation* and denoted by $\text{SU}(x_j, T)$. This measure concerns the analysis of relevance of input attributes.

Definition 2 (F-Correlation) [47] *F-correlation*, which investigates the correlation between a pair of features $x_j, x_i (i \neq j)$, is used to approximate redundancy and is denoted as $\text{SU}(x_j, x_i)$.

Note that $\text{SU}(x_j, T)$ and $\text{SU}(x_j, x_i)$ can be calculated by following the same formulas as $\text{RMI}(\tilde{R}, Y)$ in Eq. (2.19). C-correlation is implemented to detect inactive input features that are irrelevant or slightly relevant to the target concept. Irrelevant features are discarded directly, whereas slightly relevant features are subject to further investigation using the F-correlation test. That is, mutual information of input features needs to be evaluated for those features coming through the C-correlation test, $\text{SU}(x_j, T) \geq \gamma$. The value of γ is set as 0.8, as in ref. 39. Redundant features are indicated by $\text{SU}(x_j, T) < \text{SU}(x_j, x_i)$, and any features meeting this condition are to be pruned without a substantial loss of accuracy. The two-step strategy aims to alleviate computational cost, because irrelevant features captured by the C-correlation test are removed without entering the F-correlation test. Both tests are fully executed in the single-pass mode; this feature differentiates them from that in the original work [48] and is scalable for online real-time deployment.

2.3.2.2.4 RVFL Learning Mechanism

The theory of RVFLN was introduced by Huang et al. [49], and its universal approximation condition was mathematically proven later by Mitra et al. [45]. The RVFLN is founded under two solid concepts:

1. The RVFL is a universal approximator for continuous functions on a bounded finite-dimensional set.
2. The RVFL is an efficient universal approximator with a rate of approximation error convergence to zero of order $O\left(C/\sqrt{P}\right)$, where C is independent of P .

These two theorems hold under strict conditions of the hidden node as well as the scope of random regions [50], where the hidden node must be absolutely integrable and an interval of random parameters must be carefully selected—it is impossible to satisfy universal approximation conditions with any hidden nodes and/or any intervals. The compact form of the eT2RVFLN is formulated as:

$$T = \Omega\varphi, \varphi^\downarrow T$$

$$= \Omega, \varphi^o = \begin{bmatrix} x_e^1 h_1^0(X_1), & \cdots, & x_e^{2u+1} h_1^0(X_N) \\ \vdots & \dots & \vdots \\ x_e^1 h_p^0(X_1), & \cdots, & x_e^{2u+1} h_p^0(X_N) \end{bmatrix} \in \mathfrak{R}^{P \times N}, \quad (2.21)$$

where φ^\downarrow is the Moore-Penrose generalized inverse matrix. It is the most widely known type of matrix pseudoinverse. This matrix is commonly used to find the minimum (Euclidean) norm solution to a system of linear equations with multiple solutions. Some methods can be used to calculate the Moore-Penrose generalized inverse matrix. The orthogonal method can be formed $(\varphi^T, \varphi)^{-1} \varphi^T$ if $\varphi^T \varphi$ is not singular. Otherwise, the regularization technique should be applied. Note that the original RVFL network uses the steepest descent algorithm to adjust the output node, but other research clearly mentioned that a closed-form solution can be applied as an alternative whenever a matrix inversion is feasible [47].

The aforementioned learning method realizes the scheme of batch learning. This method assumes that the complete data set is available and can be revisited in multiple passes. For that reason, fuzzily weighted generalized recursive least squares (FWGRLS) method is deployed to fine-tune the local subsystem analytically. FWGRLS constitutes a local learning version of the generalized least squares method [46]. The cost function of the FWGRLS method is formulated as:

$$E(\Omega_n) = \sum_{n=1}^N ((t_n - y_n)^T R_n^{-1} (t_n - y_n) + 2\varpi \xi(\Omega_n) + (\Omega_n - \Omega_0)^T P_0^{-1} (\Omega_n - \Omega_0)), \quad (2.22)$$

where $R^n \in R^{(2u+1) \times (2u+1)}$ is the covariance matrix of the modelling error, whereas Ω_0 , ω , and $\xi(\Omega_n)$ represent the initial weight vector, regularization parameter, and generalized decay function, respectively.

The adaptation formulas of FWGRLS are expressed as:

$$\psi(n) = \Psi(n-1) F(n) \left(\frac{R(n)}{\Lambda_i(n)} + F(n) \Psi(n-1) F^T(n) \right)^{-1} \quad (2.23)$$

$$\Psi(n) = \Psi(n-1) - \psi(n) F(n) \Psi(n-1) \quad (2.24)$$

$$\Omega_i(n) = \Omega_i(n-1) - \varpi \Psi_i(n) \nabla \xi(\Omega_i(n-1)) + \Psi(n) (t(n) - y(n)) \quad (2.25)$$

$$y(n) = x_{en} \Omega_i(n) \text{ and } F(n) = \frac{\partial y(n)}{\partial \Omega(n)} = x_{en}, \quad (2.26)$$

where $\nabla \xi(\Omega_i(n-1))$ represents the gradient of the generalized decay function and $\Lambda_i(n) = (1 - q_o) \underline{R}_i + q_o \overline{R}_i$. The quadratic weight decay function is used as $\xi(\Omega_i(n-1)) = 0.5 \Omega_i(n-1)^2$ to proportionally lower the weight vector to its current value. The Hessian matrix $R_n \in \mathfrak{R}^{(2u+1) \times (2u+1)}$ is used in generalized least squares, and ω is set as $\omega = 10^{-15}$. This approach can be extended to the chunk-by-chunk adaptation scheme with ease.

2.3.2.3 When to Learn

The training sample is assigned as a reserved sample (XS_N, TS_N), given that the following criterion is observed:

$$\min_{i=1, \dots, P} (FS_i) \leq FS_{P+1} \leq \max_{i=1, \dots, P} (FS_i) \text{ and } Y_N = t_N. \quad (2.27)$$

This condition shows that a datum does not carry urgent information to the system and that the classifier is confident of its own prediction. Nevertheless, some samples may be important in updating the model later to fill the gaps uncovered by underlying training samples. Such samples are put into a buffer and are used after fully depleting centric training samples. In practice, because of the nature of data streams, which may be unbounded, reserved samples are consumed when the system is in the idle mode, and the training process is terminated when the number of reserved samples is constant.

2.4 Experimental Design

Five motors comprise one healthy and four faulty motors, which have symptoms of broken rotor bars, unbalanced voltage, stator windings, and eccentricity problems; these are measured to record the stator currents' signals. In all faulty motors, various biasing values of unbalanced voltages, turn shorts, eccentricity, and load were operated as described in Table 2.2.

The objective of this case study is to detect and classify five different induction motor conditions using our eT2RVFLN algorithm. Because an induction motor is characterized as a hybrid system and its signals are generated continuously, which leads to a problem with big data and its famous 4 Vs, eT2RVFLN, which has the capability to evolve and adapt to the new pattern, is proposed to solve this problem. eT2RVFLN is designed to cope with four issues: complexity, uncertainty, concept drift, and high dimensionality. It is an extremely fast training process with simple, fast, and easy-to-use attributes. It is a meta-cognitive learning policy: the what-to-learn, how-to-learn, and when-to-learn portions represent the effective online process of machine learning. To compare algorithm performance, four classifiers, namely eTS [51], simpl_eTS [52], DFNN [53], and FAOSPFNN [54], are compared in our experiment.

Data are collected by acquiring current signals of three-phase stator currents from five different induction motor conditions in the real experiment: healthy (or fault-free), broken rotor bar, unbalanced voltage, stator winding fault, and eccentricity problem. For comparison purposes, various scenarios are applied to the faulty motors by biasing one of three-phase unbalanced voltages, turn shorts, eccentricity, and load. The scenario of biasing values applied to each type of faulty motor is described in Table 2.2. The data collection procedure is depicted in Fig. 2.5, which based on the work of Seera et al. [20].

To acquire stator current signals, a three-phase power supply powers the induction motor and is connected to the oscilloscope using three current probes. This oscilloscope captures signals and feeds the data in real time to a server in the station room. Data are transformed to the frequency domain using the PSD technique. The PSD method comprises the 1st to the 19th harmonic, each of which is formed using a 1000-Hz frequency spectrum. To balance the three-phase system, the triple harmonic voltages should be disabled. However, this condition depends on machine supply and constructional imbalances.

Table 2.2 Biasing values applied in the experiment to each type of faulty motor condition

Operation	Value
Three-phase unbalanced voltages	5% and 10%
Turn shorts	10%
Eccentricity	30% dynamic and 10% static
Load	25%, 50%, 75%, and 100%

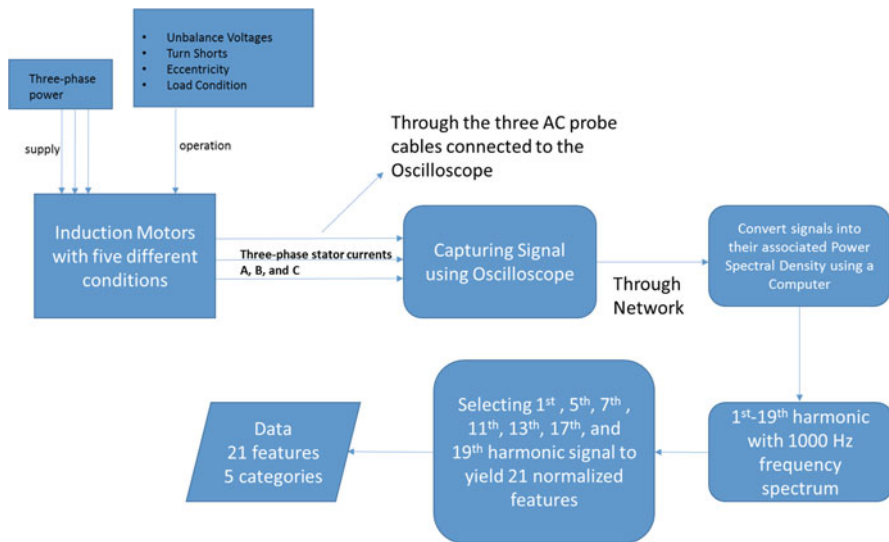


Fig. 2.5 Data collection procedure

Features were selected using expert knowledge to extract only 7 of 19 harmonics: the 1st, 5th, 7th, 11th, 13th, 17th, and 19th harmonics from each phase current A, B, and C, leading to 21 total input features. Data were normalized to ensure the stability of the adaptation phase. The eT2RVFLN analyzed the data pattern using these 21 input features and built the data model continuously to detect and diagnose any faults of the induction motor by classifying the data into five different classes of induction motor conditions: healthy, broken rotor bars, unbalanced voltages, stator winding faults, and eccentricity problems.

2.5 Numerical Results

The experimental framework was carried out under the following computer specifications: Intel Core i7-6700 CPU at 3.4 GHz and with a 16-GB memory. All simulations were run under MATLAB R2016a win-64. Our algorithm (eT2RVFLN) is benchmarked with four other algorithms—eTS, simpl_eTS, DFNN, and FAOSPFNN—for the following reasons:

1. DFNN and FAOSPFNN are used because of their ability to automatically generate rules but are semievolving classifiers because they still work under a batched learning scenario.
2. eTS and simpl_eTS are used to compare against eT2RVFLN because of its evolving characteristic. However, both of them still apply the type 1 hidden node

featuring crisp and certain properties. Neither of them implement the three pillars of meta-cognitive learning, namely what to learn, how to learn, and when to learn.

Data are randomly shuffled and partitioned into two parts: training (70%) and testing (30%). Two types of data are used, namely clean data and noisy data, the latter of which result from the addition of 20-dB noise. The algorithm performance is measured against three criteria: learning time, number of rules, and classification rate in the testing samples. In addition, 50-fold random permutations were also carried out as another experimental procedure to examine the consistency of the eT2RVFLN. Table 2.3 portrays the classifiers' performance using clean and noisy data without random permutation, whereas the numerical results with random permutation are listed in Table 2.4. Both experiments in Tables 2.3 and 2.4 are conducted with input pruning procedure. By contrast, Tables 2.5 and 2.6 show the same steps of the experiments conducted in Tables 2.3 and 2.4, but without the input pruning procedure.

Referring to Tables 2.3 and 2.4, eT2RVFLN obviously generates accuracy comparable to that of the other algorithms and causes much lower complexity than the other algorithms in both clean and noisy data, although this is only slightly

Table 2.3 Performance comparison of induction motor data set classification using the eT2RVFLN with four other benchmarked algorithms without random permutation but with input pruning

Performance of data test	eTS		DFNN		FAOSPFNN		eT2RVFLN		simpl_eTS	
	Clean	Noisy	Clean	Noisy	Clean	Noisy	Clean	Noisy	Clean	Noisy
Classification rate (testing)	1	0.71	1	0.7	0.55	0.217	0.978	0.637	1	0.643
Learning time	0.254	0.308	0.152	0.151	0.13	0.13	0.065	0.086	0.0375	0.0374
Training samples	140	140	140	140	140	140	93	94	140	140
Input attributes	21	21	21	21	21	21	20	19	21	21
Number of rules	9	9	1	1	8	12	1	1	1	1

Table 2.4 Performance comparison of induction motor data set classification using the eT2RVFLN with four other benchmarked algorithms, with 50-fold random permutation and input pruning

Performance of data test	eTS		DFNN		FAOSPFNN		eT2RVFLN		simpl_eTS	
	Clean	Noisy	Clean	Noisy	Clean	Noisy	Clean	Noisy	Clean	Noisy
Classification rate (testing)	0.996	0.834	0.998	0.702	0.563	0.273	0.97	0.836	0.979	0.702
Average learning time (s)	0.234	0.235	0.151	0.152	0.61	0.86	0.08	0.073	0.039	0.039
Average training samples	140	140	140	140	140	140	140	140	140	140
Average input attributes	21	21	21	21	21	21	19.9	20.8	21	21
Average number of rules	9.26	9.28	1	1	25.6	38	1.14	1.02	1	1

Table 2.5 Performance comparison of induction motor data set classification using the eT2RVFLN with four other benchmarked algorithms, without random permutation and without input pruning

Performance of data test	eTS		DFNN		FAOSPFNN		eT2RVFLN		simpl_eTS	
	Clean	Noisy	Clean	Noisy	Clean	Noisy	Clean	Noisy	Clean	Noisy
Classification rate (testing)	1	0.817	1	0.6833	0.917	0.717	1	0.85	1	0.683
Learning time	0.254	0.308	0.221	0.159	0.056	0.134	0.0651	0.0697	0.0375	0.0374
Training samples	140	140	140	140	140	140	140	140	140	140
Input attributes	21	21	21	21	21	21	21	21	21	21
Number of rules	9	9	1	1	4	6	1	1	1	1

Table 2.6 Performance comparison of induction motor data set classification using the eT2RVFLN with four other benchmarked algorithms, with 50-fold random permutation but without input pruning

	eTS		DFNN		FAOSPFNN		eT2RVFLN		simpl_eTS	
	Clean	Noisy	Clean	Noisy	Clean	Noisy	Clean	Noisy	Clean	Noisy
Classification rate (testing)	0.99.6	0.834	0.998	0.7020	0.943	0.823	1	0.838	0.979	0.7023
Average learning time (s)	0.234	0.235	0.148	0.153	0.048	0.129	0.0694	0.078	0.039	0.039
Training samples	140	140	140	140	140	140	140	140	140	140
Input attribute	21	21	21	21	21	21	21	21	21	21
Average number of rules	9.26	9.28	1	1	3.68	9.32	1	1.02	1	1

lower than that caused by simpl_eTS. It is worth noting that these experiments apply an input pruning procedure for eT2RVFLN, which slightly decreases the accuracy. In general, under the 50-fold random permutation procedure, the accuracy of the eT2RVFLN, as well as that of the other algorithms, increases slightly.

The data used in the experiment without random permutation (Tables 2.3 and 2.5) are original ordered data. This means that every class is trained in order, whereas in other experiments (with random permutation; Tables 2.4 and 2.6), the rows are shuffled, which generates better generalization.

Referring to Tables 2.5 and 2.6, eT2RVFLN obviously outperforms the other algorithms in both experiments (under clean and noisy data) in terms of accuracy. These experiments were carried out without sample deletion and input pruning.

In general, algorithms such DFNN and FAOSPFNN present a batched learning scheme that is hardly scalable to cope with a fast sampling rate, although the DFNN delivered the second highest classification rate (after eT2RVFLN; Table 2.6) over its counterpart. The eT2RVFLN possesses online dimensionality reduction, which is capable of substantially alleviating network complexity by deleting poor input attributes. In addition, the eT2RVFLN also uses fewer samples during the training process. This is shown in Table 2.3; it uses 20 and 19 attributes, and

Table 2.7 Module complexity for four benchmarked algorithms

Algorithm	Module complexity
eTS	$O(P(m + n + 1) + (m + n) + \rho(P(m + n + 2)))$
Simpl_eTS	$O(P(m + n + 1) + 2(m + n) + \rho(P(m + n + 1)) + P)$
DFNN	$O(P(Nm + m))$
FAOSPFNN	$O(P(Nm))$

93 and 94 samples, for the experiments using clean and noisy data, respectively. This mechanism has a positive impact on the numerical results, where it attained the lowest number of input features. Unlike other consolidated algorithms, the eT2RVFLN imposed the lowest sample consumption because inconsequential data streams can be detected and ruled out from the training process. This mechanism is proven effective to speed up execution time, which is reflected in our numerical results. In terms of speed, the eT2RVFLNN experienced the second best run times in comparison with simpl_eTS. However, the execution times shown in both Tables 2.3 and 2.4 are the average running time for every single datum. The average running time for learning the entire data set will be faster because of the deletion of inconsequential data streams during learning.

The computational complexity of the eT2RVFLN can be obtained by analyzing the computational cost of each learning module. In the what-to-learn module, the computational burden of WTN = $O(mP)$. The how-to-learn module shows the computational power of HTN = $O(P + m + 2Pm + n^2m + ((n + m) + P(n + 2m)))$, whereas the when-to-learn module costs the computational complexity of WETN = $NS \times HTN$. The complexity of the eT2RVFLN with all modules is $WTN + \rho(HTN) + WETN$, where ρ represents the probability of data streams admitted as training samples. The computational complexity for the other benchmarked algorithms are summarized in Table 2.7.

P represents the number of rules, whereas N , n , and m represent the number of historic data, the number of variable inputs, and the number of variable outputs, respectively. Table 2.7 shows that, based on computational cost, the eTS and simpl_eTS are more economical than the DFNN and FAOSPFNN. This is because DFNN and FAOSPFNN retrain all data in every iteration, which causes higher complexity in comparison with the learning procedure of eTS and Simpl_eTS, which train incoming data in a single-pass mode.

2.6 Conclusion

This chapter proposes a new approach for detecting and diagnosing induction motor faults online based on a recently developed meta-cognitive learning algorithm, namely the eT2RVFLN. Real-time experiments using a laboratory-scale test bed was undertaken; sensor data were collected and preprocessed while relevant features were extracted. The advantage of the eT2RVFLN is obvious for online fault

detection: it deploys three learning scenarios—what to learn, how to learn, and when to learn—in the strictly online environment. The what-to-learn and when-to-learn schemes make it possible for training samples to be significantly reduced and for training samples to be effectively managed, which improves predictive accuracy. The how-to-learn phase itself offers a combination between evolving and random learning perspectives, leading to a fast, easy-to-use, and flexible model. The advantage of our approach to detect faults was experimentally validated in the online detection of induction motor faults, where it delivered convincing performance in accuracy, scalability, and simplicity.

References

1. Montanari, M., Peresada, S. M., Rossi, C., & Tilli, A. (2007). Speed sensorless control of induction motors based on a reduced-order adaptive observer. *IEEE Transactions on Control Systems Technology*, *15*, 1049–1064.
2. Cusidó, J., Romeral, L., Ortega, J. A., Rosero, J. A., & Espinosa, A. G. (2008). Fault detection in induction machines using power spectral density in wavelet decomposition. *IEEE Transactions on Industrial Electronics*, *55*, 633–643.
3. Benbouzid, M. E. H. (2000). A review of induction motors signature analysis as a medium for faults detection. *IEEE Transactions on Industrial Electronics*, *47*, 984–993.
4. Toubakh, H., & Sayed-Mouchaweh, M. (2016). Hybrid dynamic classifier for drift-like fault diagnosis in a class of hybrid dynamic systems: Application to wind turbine converters. *Neurocomputing*, *171*, 1496–1516.
5. Traore, M., Duviella, E., & Lecoeuche, S. (2009). Comparison of two prognosis methods based on neuro fuzzy inference system and clustering neural network. *IFAC Proceedings Volumes*, *42*, 1468–1473.
6. Sayed-Mouchaweh, M., & Lughofer, E. (2012). *Learning in non-stationary environments: Methods and applications*. New York: Springer.
7. Bangalore, P., & Tjernberg, L. B. (2015). An artificial neural network approach for early fault detection of gearbox bearings. *IEEE Transactions on Smart Grid*, *6*, 980–987.
8. Martin, T. (2005). Fuzzy sets in the fight against digital obesity. *Fuzzy Sets and Systems*, *156*, 411–417.
9. López, V., del Río, S., Benítez, J. M., & Herrera, F. (2015). Cost-sensitive linguistic fuzzy rule based classification systems under the MapReduce framework for imbalanced big data. *Fuzzy Sets and Systems*, *258*, 5–38.
10. Broomhead, D. S., & Lowe, D. (1988). *Radial basis functions, multi-variable functional interpolation and adaptive networks*. Royal Signals and Radar Establishment Malvern (UK).
11. Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*, 281–305.
12. Chen, C. P., & Wan, J. Z. (1999). A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *29*, 62–72.
13. Rong, H.-J., Ong, Y.-S., Tan, A.-H., & Zhu, Z. (2008). A fast pruned-extreme learning machine for classification problem. *Neurocomputing*, *72*, 359–366.
14. Deng, Z., Choi, K.-S., Cao, L., & Wang, S. (2014). T2fela: Type-2 fuzzy extreme learning algorithm for fast training of interval type-2 TSK fuzzy logic system. *IEEE Transactions on Neural Networks and Learning Systems*, *25*, 664–676.
15. Feng, G., Huang, G.-B., Lin, Q., & Gay, R. (2009). Error minimized extreme learning machine with growth of hidden nodes and incremental learning. *IEEE Transactions on Neural Networks*, *20*, 1352–1357.

16. Mirza, B., Lin, Z., & Liu, N. (2015). Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift. *Neurocomputing*, *149*, 316–329.
17. Pratama, M., Zhang, G., Er, M. J., & Anavatti, S. (2017). An incremental type-2 meta-cognitive extreme learning machine. *IEEE Transactions on Cybernetics*, *47*, 339–353.
18. Subramanian, K., Suresh, S., & Sundararajan, N. (2013). A metacognitive neuro-fuzzy inference system (McFIS) for sequential classification problems. *IEEE Transactions on Fuzzy Systems*, *21*, 1080–1095.
19. Millan-Almaraz, J. R., Romero-Troncoso, R., Contreras-Medina, L. M., & Garcia-Perez, A. (2008). Embedded FPGA based induction motor monitoring system with speed drive fed using multiple wavelet analysis. In *international symposium on industrial embedded systems, 2008. SIES 2008* (pp. 215–220).
20. Seera, M., Lim, C. P., Ishak, D., & Singh, H. (2012). Fault detection and diagnosis of induction motors using motor current signature analysis and a hybrid FMM–CART model. *IEEE Transactions on Neural Networks and Learning Systems*, *23*, 97–108.
21. Douglas, H., Pillay, P., & Ziarani, A. (2005). Broken rotor bar detection in induction machines with transient operating speeds. *IEEE Transactions on Energy Conversion*, *20*, 135–141.
22. Penman, J., Sedding, H., Lloyd, B., & Fink, W. (1994). Detection and location of interturn short circuits in the stator windings of operating motors. *IEEE Transactions on Energy Conversion*, *9*, 652–658.
23. Cameron, J., Thomson, W., & Dow, A. (1986). Vibration and current monitoring for detecting airgap eccentricity in large induction motors. In *IEEE proceedings B (electric power applications)* (pp. 155–163).
24. Dorrell, D. G., Thomson, W. T., & Roach, S. (1997). Analysis of airgap flux, current, and vibration signals as a function of the combination of static and dynamic airgap eccentricity in 3-phase induction motors. *IEEE Transactions on Industry Applications*, *33*, 24–34.
25. Rodríguez, P. V. J., Negrea, M., & Arkkio, A. (2008). A simplified scheme for induction motor condition monitoring. *Mechanical Systems and Signal Processing*, *22*, 1216–1236.
26. Arabacı, H., & Bilgin, O. (2010). Automatic detection and classification of rotor cage faults in squirrel cage induction motor. *Neural Computing and Applications*, *19*, 713–723.
27. Lau, E. C., & Ngan, H. (2010). Detection of motor bearing outer raceway defect by wavelet packet transformed motor current signature analysis. *IEEE Transactions on Instrumentation and Measurement*, *59*, 2683–2690.
28. Abe, S., & Lan, M.-S. (1995). A method for fuzzy rules extraction directly from numerical data and its application to pattern classification. *IEEE Transactions on Fuzzy Systems*, *3*, 18–28.
29. Palmero, G. S., Santamaria, J. J., de la Torre, E. M., & González, J. P. (2005). Fault detection and fuzzy rule extraction in AC motors by a neuro-fuzzy ART-based system. *Engineering Applications of Artificial Intelligence*, *18*, 867–874.
30. Yang, B.-S., Oh, M.-S., & Tan, A. C. C. (2009). Fault diagnosis of induction motor based on decision trees and adaptive neuro-fuzzy inference. *Expert Systems with Applications*, *36*, 1840–1849.
31. Sii, H. S., Ruxton, T., & Wang, J. (2001). A fuzzy-logic-based approach to qualitative safety modelling for marine systems. *Reliability Engineering & System Safety*, *73*, 19–34.
32. Brotherton, T., Chadderdon, G., & Grabill, P. (1999). Automated rule extraction for engine vibration analysis. In *1999 IEEE aerospace conference proceedings* (pp. 29–38).
33. Sadeghian, A., Ye, Z., & Wu, B. (2009). Online detection of broken rotor bars in induction motors by wavelet packet decomposition and artificial neural networks. *IEEE Transactions on Instrumentation and Measurement*, *58*, 2253–2263.
34. Lee, S.-h., Wang, Y.-q., & Song, J.-i. (2010). Fourier and wavelet transformations application to fault detection of induction motor with stator current. *Journal of Central South University of Technology*, *17*, 93–101.
35. Liu, X., Ma, L., & Mathew, J. (2009). Machinery fault diagnosis based on fuzzy measure and fuzzy integral data fusion techniques. *Mechanical Systems and Signal Processing*, *23*, 690–700.

36. Ondel, O., Boutleux, E., Clerc, G., & Blanco, E. (2008). FDI based on pattern recognition using Kalman prediction: Application to an induction machine. *Engineering Applications of Artificial Intelligence*, *21*, 961–973.
37. Adjallah, K. H., Han, T., Yang, B.-S., & Yin, Z.-J. (2007). Feature-based fault diagnosis system of induction motors using vibration signal. *Journal of Quality in Maintenance Engineering*, *13*, 163–175.
38. Jang, J.-S. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, *23*, 665–685.
39. Patra, J. C., & Kot, A. C. (2002). Nonlinear dynamic system identification using Chebyshev functional link artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *32*, 505–511.
40. Savitha, R., Suresh, S., & Kim, H. J. (2014). A meta-cognitive learning algorithm for an extreme learning machine classifier. *Cognitive Computation*, *6*, 253–263.
41. Lughofer, E., & Buchtala, O. (2013). Reliable all-pairs evolving fuzzy classifiers. *IEEE Transactions on Fuzzy Systems*, *21*, 625–641.
42. Pratama, M., Er, M. J., Anavatti, S. G., Lughofer, E., Wang, N., & Arifin, I. (2014). A novel meta-cognitive-based scaffolding classifier for sequential non-stationary classification problems. In *2014 IEEE international conference on fuzzy systems (FUZZ-IEEE)* (pp. 369–376).
43. Pratama, M., Anavatti, S. G., & Lughofer, E. (2014). GENEFIS: Toward an effective localist network. *IEEE Transactions on Fuzzy Systems*, *22*, 547–562.
44. Han, H., Wu, X.-L., & Qiao, J.-F. (2014). Nonlinear systems modeling based on self-organizing fuzzy-neural-network with adaptive computation algorithm. *IEEE Transactions on Cybernetics*, *44*, 554–564.
45. Mitra, P., Murthy, C., & Pal, S. K. (2002). Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*, 301–312.
46. Oentaryo, R. J., Pasquier, M., & Quek, C. (2011). RFCMAC: A novel reduced localized neuro-fuzzy system approach to knowledge extraction. *Expert Systems with Applications*, *38*, 12066–12084.
47. Yu, L., & Liu, H. (2004). Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, *5*, 1205–1224.
48. Zhang, R., Lan, Y., Huang, G.-B., & Xu, Z.-B. (2012). Universal approximation of extreme learning machine with adaptive growth of hidden nodes. *IEEE Transactions on Neural Networks and Learning Systems*, *23*, 365–371.
49. Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*, *70*, 489–501.
50. Huang, G.-B., Chen, L., & Siew, C. K. (2006). Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Transactions on Neural Networks*, *17*, 879–892.
51. Angelov, P. P., & Filev, D. P. (2004). An approach to online identification of Takagi-Sugeno fuzzy models. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *34*, 484–498.
52. Angelov, P., & Filev, D. (2005). Simpl_eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models. In *The 14th IEEE international conference on fuzzy systems, 2005. FUZZ'05* (pp. 1068–1073).
53. Wu, S., & Er, M. J. (2000). Dynamic fuzzy neural networks—a novel approach to function approximation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *30*, 358–364.
54. Wang, N., Er, M. J., & Meng, X. (2009). A fast and accurate online self-organizing scheme for parsimonious fuzzy neural networks. *Neurocomputing*, *72*, 3818–3829.

Chapter 3

Optimal Adaptive Threshold and Mode Fault Detection for Model-Based Fault Diagnosis of Hybrid Dynamical Systems



Om Prakash, A. K. Samantaray, and R. Bhattacharyya

3.1 Introduction

Fault Detection and Isolation (FDI) plays a crucial part in the health monitoring of engineering systems to ensure their safety, reliability and maintainability [1–3]. FDI of Hybrid Dynamical Systems (HDS) is complicated because the dynamical behaviour of such systems includes family of continuous dynamics in which each continuous dynamics is triggered by a specific combination of discrete modes (supervisory controller mode or autonomous mode). Examples of HDS are embedded systems, hydraulic systems with controlled actuators, pressure relief valves, check valves, and switched electrical/electronic systems, etc.

In model-based quantitative FDI, residuals are the main indicators of any inconsistency in the system's operation. Generally, any non-zero value of residual indicates a fault, but residual also shows a non-zero value due to modelling, parameter and measurement uncertainties involved in the system. Thus, to distinguish a fault from the uncertainties effects, usually, interval model-based and statistical testing-based methods are employed [4]. In the model-based quantitative FDI, interval-based technique is widely preferred for the generation of residual threshold. This technique is suitable for FDI when the uncertainties distribution of the parameters and the measurements are unknown, but their bounds are known. Thus, when a system operates according to its expected behaviour generated from a model then the residuals also remain bounded within generated set of designed thresholds. The thresholds are designed by accounting for the modelling errors, parameter uncertainties and measurement noises. A robust diagnosis system design

O. Prakash · A. K. Samantaray (✉) · R. Bhattacharyya
Systems, Dynamics and Control Laboratory, Department of Mechanical Engineering,
Indian Institute of Technology-Kharagpur, Kharagpur, West Bengal, India
e-mail: omimech21@iitkgp.ac.in; samantaray@mech.iitkgp.ernet.in;
rmail@mech.iitkgp.ernet.in

tries to strike a balance between the misdetection or delayed detection of fault and false detection. Misdetection refers to inability to detect the presence of a fault whereas false detection refers to detecting a fault where it is actually absent.

Usually, the residual thresholds for robust diagnosis are generated based on the worst case conditions of the model, parameter and measurement uncertainties. Such thresholds are called adaptive thresholds in which summation of absolute values of the contributions from all parametric uncertainties is taken together with an additional small static threshold needed to account for measurement noises. The parametric uncertainties arise out of measurement of system parameter values either directly or through parameter estimation. Also, parametric uncertainties can include the unknown direction drifts of the parameter values after long time operation since the last time when those parameter values were explicitly measured or estimated. The adaptive threshold is robust because it considers that all parameters can have maximum possible deviation in arbitrary direction, i.e. either higher or lower than the corresponding estimated parameter values. This inflates the threshold and for small faults, the residual may not cross the threshold. For larger faults, the time taken to cross the threshold may be significant leading to detection delay.

In model-based quantitative FDI, various methods exist for residual generation like Analytical Redundancy Relations (ARRs)-based, observer-based, parity relation-based and parameter estimation-based methods [5]. Among the various existing methods, ARRs-based method is more popular for residuals and thresholds generation. In fact, these can be easily derived using Bond Graph (BG) modelling technique. BG is a multi-energy domain modelling tool [6–8] which is useful for integrated system design and provides a common framework for modelling, simulation, controller development and diagnosis system development. For HDS, Global-ARRs (GARRs) are used in place of ARRs. GARRs are valid at any working mode of the system and can be obtained from the Hybrid BG (HBG) model as proposed in [5, 9–12]. ARRs or GARRs are expressions for model constraints and residuals are their numerical values when those are evaluated using the measured data and nominally known parameter values.

For the uncertain dynamical system, the existing approaches propose the adaptive threshold as an uncertain part of ARR and decouple the certain/nominal part of the ARR from its uncertain part. The numerical evaluation of nominal part yields nominal residual and that of the uncertain part gives the numerical residual thresholds. This kind of ARR and threshold partitioning is easily implemented in BG modelling tool using the concept of Linear Fractional Transformation (LFT) [13–15]. In fact, the dynamic model, its simulation, ARR and threshold equations derivation can be done using the common BG modelling framework, which is used in this chapter.

The fault isolation depends on the fault signatures observed due to a fault. A fault signature is an encoding of the set of residuals which behave abnormally due to a certain fault. If the fault signature due to a specific fault is different from all other fault signatures due to other faults then the specific fault can be isolated. Thus, fault isolation is possible only when a specific fault causes deviations in specific residuals in some specific manner from which a reverse one-to-one mapping between the

residual responses linking to faults can be uniquely established. Sensor placements can be used to design residuals that are structured, i.e., all fault signatures are different from each other. BG model-based approach is useful in designing the sensor placements which give structured residuals.

In a HDS, isolation and identification of true faults are the other key issues after detecting a fault. This is much more complicated in comparison to continuous dynamic systems because the discrete mode fault may occur besides or together with a parametric fault in such a system [16–18]. Examples of discrete mode faults are switch failure, valve stuck on/off fault, control command communication fault, mode transition failure, etc. Fault detection due to discrete mode fault is usually easy since it produces significant changes in the system dynamics. But the fault isolation is tricky since the discrete mode fault in a component shares the same fault signature as the partial/full parametric fault associated with the same component. This isolation task also gets more complicated when more than one components share the same fault signature, i.e., there is no unique mapping between the observed fault symptoms to the faults. Thus, a Set of Suspected Faults (SSF) is usually generated. SSF includes the unknown actual faulty parameter along with some other parameters whose faults give rise to the same fault signature as that by the actual faulty parameter.

When the fault effects are indistinguishable from fault signatures alone, parameter estimation of elements in SSF has been proposed for fault isolation [5, 19–22]. However, the inclusion of discrete mode faults in the SSF complicates the parameter estimation task. For any fault occurring in the system, we need to first confirm whether the residual inconsistency is due to a mode fault or due to a parametric fault. If the inconsistency is due to a discrete mode fault then the parameter estimation is irrelevant and the process needs to be reconfigured or shutdown. If the residual inconsistency is confirmed to be not due to a discrete mode fault then it is assumed to be due to a parametric fault and the discrete fault parameter can be removed from the SSF. Then parameter estimation is required to estimate the magnitudes of the remaining parameters in the SSF with known mode information. Thus, the core objective of HDS fault diagnosis scheme is to discriminate between the discrete mode fault and parametric fault and to isolate the true fault with estimation of its severity so that the decisions regarding the fault accommodation or system reconfiguration can be appropriately taken.

Some recent research based on BG approach which address the generation of robust thresholds for improving fault diagnosis can be found in [23, 24]. In [23], BG-LFT model along with Interval Extension Functions (IEF) technique is used for the generation of robust optimized threshold. The method was tested and demonstrated on a continuous dynamical system through simulation and showed improved detectability. However, in [24], the problem of false detection due to all the sensitive residuals remaining bounded within their respective threshold at the same time in presence of a small fault is discussed. Therein, multi-thresholds generation technique by using the residual sensitivity relations is proposed as a solution. Also, in [25], a technique for the selection of a subset of most sensitive residuals to the fault among many residual candidates based on sensitivity analysis is discussed to improve fault detectability and isolatability with the objective of minimizing

false alarms. But, this technique is valid only when more numbers of residuals are sensitive to a particular fault. These approaches have focused on a parametric fault in a continuous dynamical system only. Few works reported for improving FDI for the HDS can be found in [16, 26–31]. In [16], ARR set obtained from GARRs is used for mode tracking in the presence of parametric fault in the HDS. In [26], incremental BG method and ARRs-based approach are used for the generation of mode-dependent adaptive threshold and for mode identification, respectively. In [27], sensitivity signature matrices for both discrete mode fault and parametric fault are proposed for improving fault isolatability. In addition, a technique for residual filtering is presented to account for various uncertainties involved in the uncertain HDS. However, threshold generation scheme is not properly discussed. Moreover, the proposed approach only provides the set of suspected discrete and parametric faults after fault diagnosis, but how to isolate the true faults among suspected candidates is not discussed there. In [28, 29], a notion of hybrid possible conflicts is introduced for improving FDI of HDS. For this, HBG technique is utilized to decompose the model into various sub-models and the respective sub-model is used for the identification of a discrete or a parametric fault. In [30], parity space method is used for the residual generation while the set-membership identification approach based on zonotopes is used for the adaptive threshold generation. In [31], worst case adaptive threshold is used for robust diagnosis and only additive fault in the measurement is considered. Other approaches based on decentralized diagnosis architecture and machine learning techniques to discriminate discrete and parametric fault for HDS can be consulted in [32] and [33], respectively.

The primary objective of this chapter is to optimally select the adaptive thresholds by using the residual from the known healthy system state to account for the real uncertainties such as small drifts in parameter values in the running system. This means, after offline design of the FDI system, training in the healthy state of the real system is required. In addition, we propose to discriminate between parametric faults and discrete mode faults by an initial hypothesis based on the magnitude of the residuals. The hypothesis is then validated using further formal procedures. In the present work, we assume that only single fault (parametric fault or discrete mode fault) occurs in the system at any instant. In addition, we also assume that response of residual due to any parametric fault is different from the any suspected discrete mode fault in the system. This is the similar assumption considered in [16]. For example, response of residual due to partial blockage fault in an on-off valve and discrete mode fault like valve stuck-off fault should be different. The symptoms uncertainties are not considered in this work. It is assumed that all the residuals sensitive to the fault violate their thresholds and all fault symptoms are observed in the residuals. Readers may refer [34–37] which consider symptoms uncertainties effects during diagnosis of continuous dynamical systems.








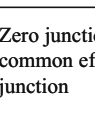
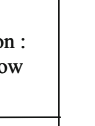


The remaining parts of the chapter are organized as follows. In Sect. 3.2, BG and HBG modelling techniques are briefly discussed. Section 3.3 introduces the HBG-based residual, adaptive threshold generation and common terms used in the FDI. Moreover, it also presents the techniques for optimal selection of adaptive thresholds and discrete mode fault identification. Section 3.4 deals with

application of the proposed diagnosis and thresholding techniques on a hybrid two-tank system through numerical simulation. Finally, Sect. 3.5 gives main conclusions and perspectives.

3.2 Bond Graph

Bond Graph (BG) is a unified multi-energy domain modelling approach for multi-physics energetic systems [6, 8]. BG is a graphical language. A BG model is an interconnected graph of different generic elements as nodes and the edges between the nodes are called bonds. These generic elements and their definitions are briefed in Table 3.1. The elements are used for representing the sources and

Table 3.1 Definitions of the generic BG elements

		Symbol	Constitutive equation	Name
Sources		Se:e 	$\begin{cases} e(t) \text{ given by the source} \\ f(t) \text{ arbitrary} \end{cases}$	Source of effort
		Sf:f 	$\begin{cases} f(t) \text{ given by the source} \\ e(t) \text{ arbitrary} \end{cases}$	Source of flow
Passive elements	Dissipator	 R	$\Phi_R(e, f) = 0$	Resistance
	Energy stores	 C	$\Phi_C(e, q) = 0$	Capacitance
		 I	$\Phi_I(f, p) = 0$	Inertance
Junctions	Transducers	 TF :m	$\begin{cases} e_1 = me_2 \\ f_2 = mf_1 \end{cases}$	Transformer
		 GY :r	$\begin{cases} e_1 = rf_2 \\ e_2 = rf_1 \end{cases}$	Gyrator
	Junctions	 0	$\begin{cases} e_1 = e_2 = e_3 \\ f_1 - f_2 + f_3 = 0 \end{cases}$	Zero junction : common effort junction
		 1	$\begin{cases} f_1 = f_2 = f_3 \\ e_1 - e_2 + e_3 = 0 \end{cases}$	One junction : common flow junction
Sensors	Sensors	 De:e	$\begin{cases} e = e(t) \\ f = 0 \end{cases}$	Sensors (Detectors)
		 Df:f	$\begin{cases} f = f(t) \\ e = 0 \end{cases}$	

the components associated with multi-energy domain system by using a lumped parameter modelling framework. The generalized elements are two active sources (Se , Sf), three passive elements (R , C , I), four power conservation junctions (TF , GY , 0-equal effort and 1-equal flow) and two detectors/sensors (De , Df). Each power bond is associated with two generalized power variables, i.e. effort (e) and flow (f) variables, and product ($e \cdot f$) of these variables gives power in the respective bond. These power variables represent different variables in different domains. For instance, in hydraulic domain, pressure and volume flow rate (or mass flow rate, when density is constant) are represented as effort and flow power variables, respectively.

In the causalled BG model, half arrow in a bond indicates an assumed direction of power flow and perpendicular line at the end of a bond is a causal stroke which describes the mathematical or computational causality. In the causalled power bond, the effort (e) information is directed towards the causal stroke end whereas the flow (f) information is directed in the opposite direction. Also, the full arrow in a bond indicates the signal/information transmitted by the element such as sensor, integrator, etc. The causal and structural properties of BG allow a systematic way of equations generation for study of system dynamics and rule development for fault diagnosis of the system. The causal forms of various BG elements, their corresponding causal equations and block diagrams, and the causality assignment rules are given in Table 3.2. For example, computation form of an electric circuit model which consists of voltage source $Se:V$ and resistor R (shown in Fig. 3.1a) is represented by the causal BG model in Fig. 3.1a₁ and by block diagram in Fig. 3.1a₂. A fixed causality is used for voltage source Se , i.e. it gives the effort V to resistor R and receives the current $i = V/R$ which is the output of R . Likewise, computation form of an electric circuit model which consists of current source $Sf:i$ and resistor R (shown in Fig. 3.1b) is represented by the causal BG model in Fig. 3.1b₁ and by block diagram in Fig. 3.1b₂. A fixed causality is used for current source Sf , i.e. it gives the flow i to resistor R and receives the effort $V = iR$ which is the output of R .

Two passive elements (C and R , respectively) used for modelling hydraulic tank and valve are shown in Fig. 3.2. The constitutive laws, respectively, for C and R are stated as

$$\Phi_C:P = \frac{g}{A} \int (\dot{m}_{in} - \dot{m}_{out}) \cdot dt \cong e = \frac{1}{C} \int f \cdot dt, \quad (3.1)$$

where P is the pressure, A is cross section area of tank, g is acceleration due to gravity, \dot{m}_{in} and \dot{m}_{out} are the rate of mass inflow and outflow, respectively.

$$\Phi_R:\dot{m} = \begin{cases} C_d' A \rho \sqrt{\frac{2(P_1 - P_2)}{\rho}} = C_d \sqrt{\Delta P} & \text{for nonlinear case,} \\ C_d' A \rho (P_1 - P_2) = C_d \Delta P & \text{for linear case.} \end{cases} \quad (3.2)$$

where C_d' and C_d are the actual and equivalent discharge coefficient of the valve, respectively, ΔP is the pressure drop across the valve and ρ is the density of liquid.

Table 3.2 Causality assignment for BG elements

Element	Bond graph	Causal equation	Block diagrams	Rule
Effort source	Se: $e \rightarrow \dashv$	e is known	$f \leftarrow$ System Se: $e \rightarrow$ System	Output of Se (of Sf) is an effort (flow) and is an input for the system.
Flow source	Sf: $f \dashv \rightarrow$	f is known	$e \leftarrow$ System Sf: $f \rightarrow$ System	Rule : The causality is compulsory
0 Junction		$\begin{cases} e_2 = e_1 \\ e_3 = e_1 \\ e_4 = e_1 \\ f_1 = -f_2 + f_3 - f_4 \end{cases}$		Only one effort (here e_1) is input. Rule: Only one bond can have causal stroke near the 0 junction.
1 Junction		$\begin{cases} f_2 = f_1 \\ f_3 = f_1 \\ f_4 = f_1 \\ e_1 = -e_2 + e_3 - e_4 \end{cases}$		Only one flow (here f_1) is an input. Rule: Only one bond can have a causal stroke away from the 1 junction.
TF		$\begin{cases} e_1 = m e_2 \\ f_2 = m f_1 \\ e_2 = \frac{1}{m} e_1 \\ f_1 = \frac{1}{m} f_2 \end{cases}$		Only one effort and one flow are inputs Rule : One causal stroke near TF
GY		$\begin{cases} e_1 = r f_2 \\ e_2 = r f_1 \\ f_2 = \frac{1}{r} e_1 \\ f_1 = \frac{1}{r} e_2 \end{cases}$		Two efforts or two flows are inputs Rule : Two or no causal strokes near GY
C	$\dashv \xrightarrow{e} \text{C:C}_1$ $\dashv \xrightarrow{e} \text{C:C}_1$	$e = \Phi_C(\int f dt) = \Phi_C(q)$ $f = \frac{d}{dt}(\Phi_C^{-1}(e))$	$f \rightarrow \Phi_C(\int f dt) \rightarrow e$ $e \rightarrow \frac{d}{dt}(\Phi_C^{-1}(e)) \rightarrow f$	Integral causality: effort is an output Derivative causality : flow is an output
I	$\dashv \xrightarrow{e} \text{I:I}_1$ $\dashv \xrightarrow{e} \text{I:I}_1$	$f = \Phi_I(\int e dt) = \Phi_I(p)$ $e = \frac{d}{dt}(\Phi_I^{-1}(f))$	$e \rightarrow \Phi_I(\int e dt) \rightarrow f$ $f \rightarrow \frac{d}{dt}(\Phi_I^{-1}(f)) \rightarrow e$	Integral causality : flow is an output Derivative causality : effort is an output
R	$\dashv \xrightarrow{e} \text{R:R}_1$ $\dashv \xrightarrow{e} \text{R:R}_1$	$e = \Phi_R(f)$ $f = \Phi^{-1}_R(e)$	$f \rightarrow \Phi_R(f) \rightarrow e$ $e \rightarrow \Phi^{-1}_R(e) \rightarrow f$	Resistance causality: output is an effort Conductance causality : flow is an output

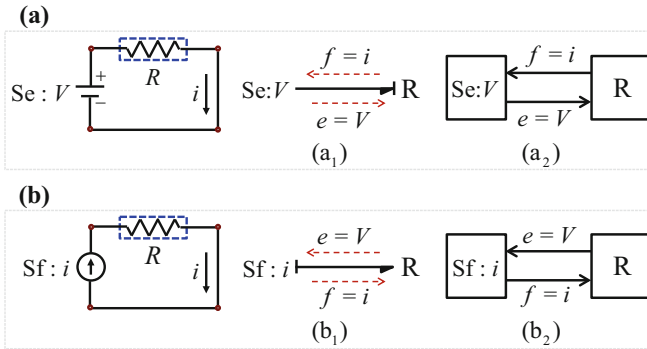


Fig. 3.1 Causalities in BGs. (a) Effort is known for R. (b) Flow is known for R

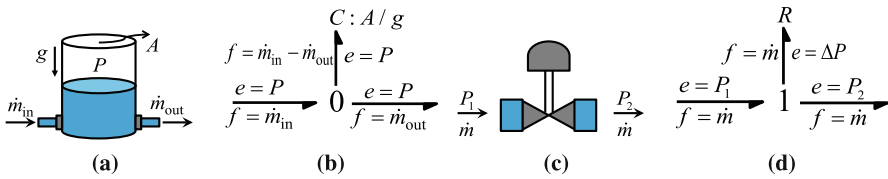


Fig. 3.2 (a) Hydraulic tank. (b) BG model of hydraulic tank. (c) Hydraulic valve. (d) BG model of hydraulic valve

3.2.1 Hybrid Bond Graph (HBG) Model

The extended form of BG technique, called HBG technique [12], is well-suited for modelling of HDS by using the concept of assigning preferred causalities to all energy storage elements and switched junction elements. Since HDS has different dynamics in different modes, its model structure changes due to mode change. However, using preferred causalities in HBG model, all active BG elements in single global model remain active in any working mode without reassigning any new causality to the model.

For example, consider the hybrid hydraulic system shown in Fig. 3.3. The system consists of a hydraulic pump, tank T_1 , linear valve (V_1) and drain pipeline (L_1). The valve V_1 is controlled in open and closed state by a supervisory controller. The water pressure in tank T_1 is measured by the pressure sensor $P_1(t)$. The dynamics of considered system includes the combination of both controlled and autonomous discrete events that change the configuration of the system. The controlled discrete event occurs due to the external command signal (a_{V_1}) given by the supervisory controller to the valve V_1 . Autonomous discrete events occur due to change in internal state or variable of the system. Here, the autonomous discrete event (a_{L_1}) occurs at particular pre-set condition for drain pipeline L_1 , i.e. when pressure $P_1(t)$ exceeds pressure $\rho g H_{L_1}$ corresponding to height H_{L_1} .

Fig. 3.3 Schematic of hybrid tank system

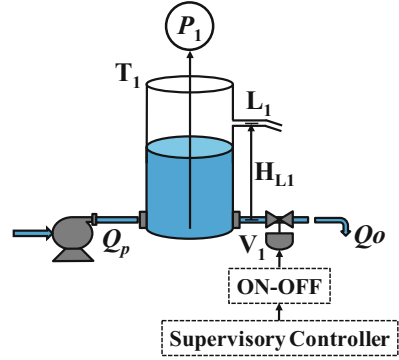
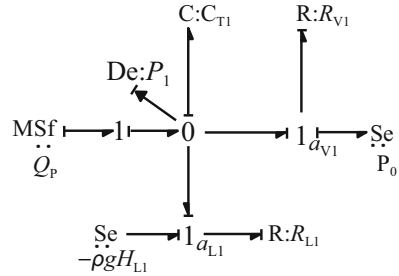


Fig. 3.4 HBG model of hybrid tank system



The HBG model of the considered hybrid system (Fig. 3.3) is presented in Fig. 3.4, where tank T_1 capacity ($C_{T1} = A/g$, A is cross section area of tank), and valve V_1 with the connected pipe and drain pipeline L_1 are modelled by one C -element and two R -elements, respectively; and pump flow (Q_p) is modelled by MSf -element. Output sensor $P_1(t)$ is modelled by effort detector De . In Fig. 3.4, the C -element is assigned with preferred integral causality while the resistors (R_{V1} and R_{L1}) at switched junctions are assigned with preferred conductive causalities. 1-junctions with subscripts a_{V1} and a_{L1} are switched junctions related with discrete events. The flow through the corresponding switched resistors (R_{V1} and R_{L1}) is activated only at their active modes ($a_i = 1$), otherwise there is no flow through these elements at their inactive modes ($a_i = 0$). In any mode, the assigned causalities of the model are always valid, and hence it is called the global model of the HDS.

3.3 Diagnostic HBG Model for Uncertain System

Diagnostic Bond Graph (DBG) technique [38] is employed to directly derive the ARRs from the model. In a DBG, the causalities of the sensor elements are inverted, i.e., the measured data is assumed to be known and the constraint errors, i.e., ARRs, become the outputs of the model. Moreover, the storage elements in the

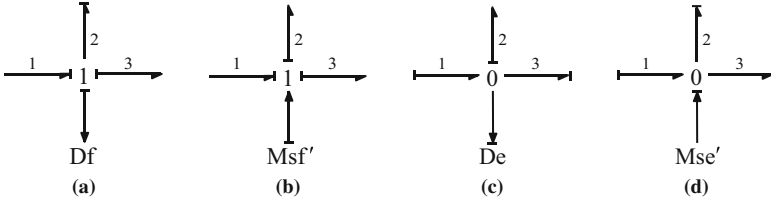


Fig. 3.5 (a) Ideal flow detector in HBG. (b) Dualized flow detector in DHBG. (c) Ideal effort detector in HBG. (d) Dualized effort detector in DHBG

model are assigned differential causality, i.e., the constitutive relations are written in derivative form instead of integral form used in a normal BG for simulation. An extended form of DBG adapted for HDS is called Diagnostic HBG (DHBG) [39] and its outputs are the GARRs. DHBG is well-suited for HDS monitoring. The uncertain system is always associated with both multiplicative (parametric) and additive (sensor noise/measurement) uncertainties. For robust diagnosis, residuals generated from the GARRs are bounded within some time varying thresholds, called adaptive thresholds and are derived from DBG-LFT form model [13–15]. Also, adaptive threshold varies with discrete mode transition in HDS as presented in [26]. Thus, DHBG-LFT form model is used here and it is generated by dualizing the flow detector Df and effort detector De of HBG model into imaginary modulated source of flow Msf' and modulated source of effort Mse' , respectively (as presented in Fig. 3.5). This is equivalent to changing the causalities of the sensors. Also, all the storage elements (C and I) are assigned with preferred derivative causalities to eliminate the initial conditions of the states from the GARRs expressions. In addition, the nominal parts of the parameters and measurements are decoupled from their uncertain parts by using the BG elements in LFT form model.

3.3.1 Modelling Parameter Uncertainty

Generally, parameter uncertainties arise out of measurement of system parameter values either directly or through parameter estimation. It may as well be considered that deviations in parameter values may occur over long use of the components and the relative limits on such deviations are known. Thus, the ideal parameter of the system is unknown. It may be assumed that the measured/estimated parameter lies within a known bounded interval around the ideal parameter value and can be written as

$$\begin{aligned}\theta_j &= \theta_{jn} \pm \Delta\theta_j \\ \text{or } \theta_j &= \theta_{jn}(1 \pm \delta_{\theta_j})\end{aligned}\quad (3.3)$$

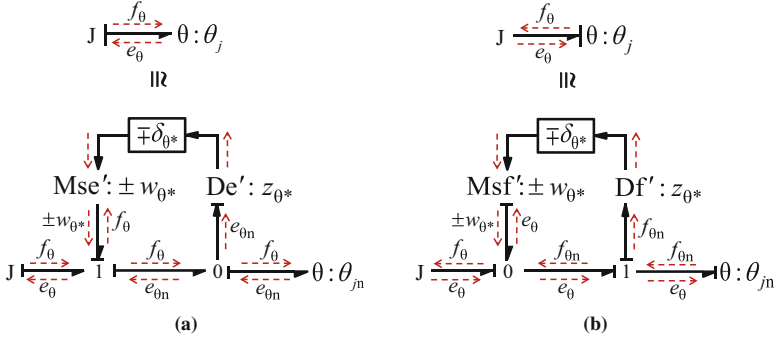


Fig. 3.6 BG-LFT model of uncertain θ (a) receives flow and returns effort causal form (b) receives effort and returns flow causal form

where $\Delta\theta_j$ and $\delta_{\theta_j} = (\Delta\theta_j/\theta_{jn})$ denote absolute and relative deviations of nominal/ideal parameter value θ_{jn} , respectively, $\theta_j \in \{I, C, R, TF, GY\}$.

DHBG-LFT form technique is used to model the parameter uncertainty of the uncertain HDS system. In this technique, the ideal parameter θ_{jn} is decoupled from its uncertain part $\pm\Delta\theta_j$. The uncertain part is handled as a disturbance either in the form of additional effort or flow which depends on the kind of BG element and its causal structure in the model. In DHBG model, parameter $\theta_j \in (I)$ and $\theta_j \in (C)$ must take the causal structures as presented in Fig. 3.6a and b, respectively, while $\theta_j \in (R)$ may take any form of causal structure as presented in Fig. 3.6a or b. In Fig. 3.6, $J \in \{1, 0, 1_{sw}, 0_{sw}\}$ denotes junction element which may be a normal or switched junction element. According to causal form presented in Fig. 3.6a, the additional disturbance in the form of modulated source of effort ($MSe': \pm w_{\theta^*}$) is brought to the junction-1 by multiplying the relative deviation ($\mp\delta_{\theta^*}$) with the nominal effort ($e_{\theta n}$) measured by an imaginary effort detector ($De': z_{\theta^*}$). Likewise, according to causal form presented in Fig. 3.6b, the additional disturbance in the form of modulated source of flow ($MSf': \pm w_{\theta^*}$) is brought to the junction-0 by multiplying the relative deviation ($\mp\delta_{\theta^*}$) with the nominal flow ($f_{\theta n}$) measured by an imaginary flow detector ($Df': z_{\theta^*}$). Note that subscript θ^* depends on the constitutive law of respective element. For instance, let us consider the $\theta_j \in (R)$ in the conductive causality which is represented in form as presented in Fig. 3.6b. If the true parameter value of a resistor R is not known exactly, it can be expressed as $R_n \pm \Delta R = R_n(1 \pm \delta_R)$, where R_n denotes nominal parameter value and $\pm\Delta R = \pm\delta_R R_n$ is the uncertain part of R . The constitutive law of linear R -element modelled in conductive causality is given as

$$f_R = \frac{1}{R_n \pm \Delta R} e_R = \frac{1}{R_n} (1 \mp \delta_{1/R}) e_R = \frac{e_R}{R_n} \mp w_{1/R} = f_{Rn} \mp w_{1/R} \quad (3.4)$$

where $(\mp \delta_{1/R}/R_n)e_R = \mp w_{1/R}$ is the additional contribution of flow because of the uncertain part of the parameter and may be treated as a disturbance. Note that $\delta_{1/R}$ is the uncertainty in estimating the value of conductance $1/R$.

Likewise, other BG-elements (TF and GY) with uncertainties in the parameter values can be modelled by using BG-LFT form [14].

3.3.2 Modelling Measurement Uncertainty

The ideal sensor measurement can also be assumed to lie within a known bounded interval and can be written as

$$Y_i = Y_{in} \pm \Delta Y_i \tag{3.5}$$

where Y_{in} denotes nominal/ideal measurement of the sensor and ΔY_i denotes measurement uncertainty. $Y_i \in \{Msf', Mse'\}$ corresponds to detectors Df and De , respectively, of the uncertain HBG model.

The uncertain flow measurement part $\pm \Delta f'$ and effort measurement part $\pm \Delta e'$ are also decoupled from their nominal flow part f'_n and effort part e'_n , respectively, and are modelled with the imaginary flow source Msf' and effort source Mse' at the respective junctions [15] as shown in Fig. 3.7a and b, respectively. Note that flow detector can be placed only at a common flow junction (1-junction) and the effort detector can be placed at a common effort junction (0-junction). Thus, nominal Msf' and Mse' occur at 1 and 0 junctions, respectively. According to Fig. 3.7a and b with the measurement uncertainties in flow and effort sensors, respectively, flow and effort in the bond numbers 1, 2 and 3 are obtained as $f'_n \pm \Delta f'$ and $e'_n \pm \Delta e'$, respectively. These information are, likewise, propagated to the rest of connected bonds at the respective junctions in the model. However, this is an unnecessary

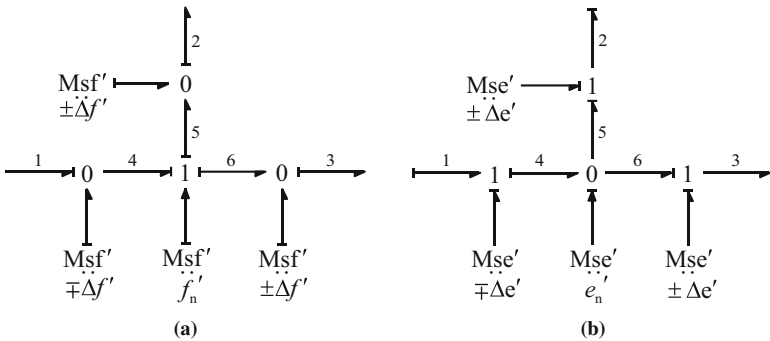


Fig. 3.7 (a) Uncertainty modelling of flow detector. (b) Uncertainty modelling of effort detector

exercise to add uncertain flows and efforts to all bonds connected to the main junction with the sensor; instead the uncertain part can be directly added to the nominal measurement value as a separate parameter.

Readers may refer [14] and [15] for more details for modelling the multiplicative (parametric) and additive (sensor noise/measurement) uncertainties by using BG tools.

3.3.3 ARR/GARR and Adaptive Threshold

DHBG-LFT model is used here to derive the equations for the GARRs and adaptive thresholds in the robust fault diagnosis. The general form of $GARR_i(\mathbf{U}, \mathbf{Y}, \boldsymbol{\theta}, \mathbf{MD})$ of an uncertain HDS may be expressed as

$$GARR_{ni}(\mathbf{U}, \mathbf{Y}, \boldsymbol{\theta}, \mathbf{MD}) \pm (\lambda_i + \lambda_{Si}) = 0 \quad (3.6)$$

where $GARR_{ni}$, λ_i and λ_{Si} represent the nominal residual (r_{ni}) ($i = 1, 2, \dots, n$; n is the number of residuals), uncertain part due to parameter uncertainties and small static uncertain part needed to account for measurement uncertainties, respectively. Also, $\mathbf{U} \in \{Se_n, Sf_n\}$ denotes known nominal input vector, $\mathbf{Y} \in \{Mse_n, Msf_n\}$ denotes nominal measurements (dualized to sources due to causality inversion), $\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_j, \dots, \theta_p]^T$ denotes a known parameter vector comprising p number of nominal parameters, $\mathbf{MD} = [a_1, a_2, \dots, a_k, \dots, a_m]^T$ denotes the switched junction mode vector comprising m number of discrete parameters, $a_k \in \{0, 1\}$ and $\lambda_i \in w_{\theta_j}$, $\lambda_{Si} \in \{w_{Mse'}, w_{Msf'}\}$.

For example, the DHBG-LFT model of the hybrid tank system (Fig. 3.3) is presented in Fig. 3.8, where the effort detector $De:P_1$ of HBG model (Fig. 3.4) is dualized into $Mse':P_1$. Also, the nominal part of the parameters C_{T1} , C_{dV1} and C_{dL1} are decoupled from their uncertain parts by using $\pm \delta_{C_{T1}}$, $\pm \delta_{C_{dV1}}$ and $\pm \delta_{C_{dL1}}$ as multiplicative uncertainties, respectively [14]. In addition, nominal part of output measurement (P_1) and input measurement (Q_P) are decoupled from their uncertain parts as additive uncertainties $\pm \Delta P_1$ and $\pm \Delta Q_P$ at the respective junctions [15]. The imaginary flow detector (Df^*) (corresponding to pressure sensor P_1 in Fig. 3.8) is used to derive a nominal GARR and an adaptive threshold as

$$Df^* = f_{25} = f_8 = f_5 - f_6 - f_7 - f_9 = f_2 - (f_{16} - f_{17}) - (f_{22} - f_{23}) - (f_{14} - f_{13}) = 0, \quad (3.7)$$

where $f_2 = Q_P \pm \Delta Q_P$, $f_{16} = C_{T1} \frac{d}{dt}(P_1 \pm \Delta P_1)$, $f_{17} = \mp w_{C_{T1}} = \pm C_{T1} \frac{d}{dt}(P_1 \pm \Delta P_1) \cdot \delta_{C_{T1}} = \pm \delta_{C_{T1}} C_{T1} \frac{d}{dt}(P_1)$, $f_{22} = a_{V1} C_{dV1}(P_1 \pm \Delta P_1)$, $f_{23} = \mp w_{C_{dV1}} = \pm a_{V1} C_{dV1}(P_1 \pm \Delta P_1) \cdot \delta_{C_{dV1}} = \pm a_{V1} \delta_{C_{dV1}} C_{dV1} P_1$, $f_{14} = a_{L1} C_{dL1} \{(P_1 \pm \Delta P_1) - \rho g H_{L1}\}$, and $f_{24} = \mp w_{C_{dL1}} = \pm a_{L1} C_{dL1} \{(P_1 \pm \Delta P_1) - \rho g H_{L1}\} \cdot \delta_{C_{dL1}} = \pm a_{L1} \delta_{C_{dL1}} C_{dL1} (P_1 - \rho g H_{L1})$.

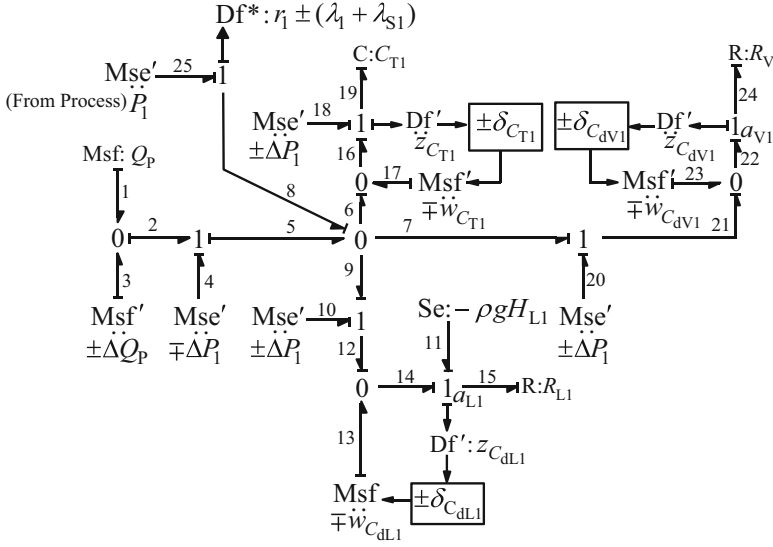


Fig. 3.8 DHBG-LFT model of hybrid tank system

Thus, after putting all the known flow values in (3.7), we obtain the $GARR_1$ in the form presented in (3.6) in terms of nominal $GARR_{n1}$ and uncertain parts λ_1 and λ_{S1} of $GARR_1$.

$$GARR_{n1} = \left(Q_P - C_{T1} \frac{d}{dt}(P_1) - a_{v1} C_{dv1} P_1 - a_{L1} C_{dL1} (P_1 - \rho g H_{L1}) \right), \quad (3.8)$$

$$\lambda_1 = \left| \left(\delta_{CT1} C_{T1} \frac{d}{dt}(P_1) \right) \right| + |(a_{v1} \delta_{Cdv1} C_{dv1} P_1)| + |(a_{L1} \delta_{CdL1} C_{dL1} (P_1 - \rho g H_{L1}))|, \quad (3.9)$$

$$\lambda_{S1} = |(\Delta Q_P)| + \left| \left(C_{T1} \frac{d}{dt}(\Delta P_1) \right) \right| + |(a_{v1} C_{dv1} \Delta P_1)| + |(a_{L1} C_{dL1} \Delta P_1)|. \quad (3.10)$$

The numerical evaluations of nominal part $GARR_{ni}$ and the uncertain part $(\lambda_i + \lambda_{Si})$ of $GARR_i$, as presented in (3.6), using \mathbf{U} , \mathbf{Y} , $\boldsymbol{\theta}$ and \mathbf{MD} along with the different known uncertainties bounds, provide the residual $r_{ni}(t)$ and adaptive threshold $\varepsilon_i(t)$, respectively as

$$r_{ni}(t) = \text{Eval} \{GARR_{ni}(\mathbf{U}, \mathbf{Y}, \boldsymbol{\theta}, \mathbf{MD})\} \text{ and } \varepsilon_i(t) = \pm \text{Eval} \{(\lambda_i + \lambda_{Si})\}. \quad (3.11)$$

Note that the derivatives of noisy measurements and discrete mode transition generally amplify the residual noise. So, usually a low-pass filter is used to filter the noises in the residuals. Since the absolute values or magnitudes of different

uncertain parts' contribution is added in the adaptive threshold, the small λ_{S_i} part may be neglected in comparison to λ_i . Except in very fast rate transient situations, the contribution of λ_{S_1} due to measurement uncertainties is very small in comparison to the parameter uncertainties λ_1 in the adaptive threshold and it may be neglected to avoid more inflated threshold.

3.3.4 Fault Signature Matrix and Coherence Vector

GARRs are used for generating fault signatures for different parametric or discrete mode faults. Fault signatures depend on the residual responses/sensitivities to parameter deviations [5, 7, 12]. In the proposed work, dynamic fault signatures for parametric and mode faults, termed as Global Fault Sensitivity Signature Matrix (GFSSM) and Mode Change Sensitivity Signature Matrix (MCSSM) [27], are used.

GFSSM [27] is an extended form of GFSM (Global Fault Signature Matrix) [39] which is dynamic in nature as every element of GFSSM is updated with instantaneous direction of change of residual with the parameter variation. This matrix includes sensitivity to fault direction, i.e. increasing ($\theta_j \uparrow$) and decreasing ($\theta_j \downarrow$) trends of the faulty parameter. Its elements are obtained as

$$\text{GFSSM}_{ji}^{\uparrow} = \begin{cases} -\text{sign}(\partial r_i / \partial \theta_j), & \text{if } r_i \text{ is sensitive to increasing } \theta_j \uparrow, \\ 0, & \text{otherwise,} \end{cases} \quad (3.12)$$

$$\text{GFSSM}_{ji}^{\downarrow} = \begin{cases} \text{sign}(\partial r_i / \partial \theta_j), & \text{if } r_i \text{ is sensitive to decreasing } \theta_j \downarrow, \\ 0, & \text{otherwise,} \end{cases} \quad (3.13)$$

where r_i is the i th column residual, $i \in \{1, 2, \dots, n\}$, n is the number of residuals, θ_j is the j th row parameter of the GFSSM, $j \in \{1, 2, \dots, p\}$, p is the number of parameters. Thus, for each parameter, there are usually two rows in GFSSM. However, some parameters can vary in one direction (such as leakage) and hence, they have only one row entry in the GFSSM.

MCSSM [27] is an extended form of MCSM (Mode Change Signature Matrix) [39], which is also dynamic in nature like GFSSM and it also has an ability to distinguish between increasing ($a_k \uparrow$, changes from 0 to 1) and decreasing ($a_k \downarrow$, changes from 1 to 0) trends of discrete mode fault. Its elements are obtained as

$$\text{MCSSM}_{ki}^{\uparrow} = \begin{cases} -\text{sign}(\partial r_i / \partial a_k), & \text{if } r_i \text{ is sensitive to increasing } a_k \uparrow, \\ 0, & \text{otherwise,} \end{cases} \quad (3.14)$$

$$\text{MCSSM}_{ki}^{\downarrow} = \begin{cases} \text{sign}(\partial r_i / \partial a_k), & \text{if } r_i \text{ is sensitive to decreasing } a_k \downarrow, \\ 0, & \text{otherwise,} \end{cases} \quad (3.15)$$

where a_k is the k th row discrete parameter of the MCSSM, $a_k \in \{0, 1\}$, $k \in \{1, 2, \dots, m\}$ and m is the number of discrete parameters.

A coherence vector (**CV**), whose standard form is $\mathbf{CV} = [cv_1(t), cv_2(t), \dots, cv_n(t)]$, where $cv_i(t) \in \{0, +1, -1\}$, $i = 1, 2, \dots, n$, is commonly used to generate the alarm state of the supervised/monitored plant. The respective element, $cv_i(t)$, of coherence vector (**CV**) depends on a decision procedure $\Theta(r_i(t))$ and is obtained as

$$cv_i(t) = \Theta(r_i(t)) = \begin{cases} 0, & \text{if } -\varepsilon_i(t) \leq r_i(t) \leq \varepsilon_i(t) \\ +1, & \text{if } r_i(t) \geq \varepsilon_i(t) \\ -1, & \text{otherwise.} \end{cases} \quad (3.16)$$

During nominal working of the plant, all $cv_i(t)$ are zero; otherwise any non-zero value of $cv_i(t)$ in the **CV** indicates that some inconsistency in the plant operation and an alarm is raised. Once an alarm is raised, the generated **CV** is matched with the GFSSM and MCSSM for fault isolation and the parameter which has a unique match with the **CV** is isolated as the faulty parameter [27]. Detectability index (D_b) and isolatability index (I_b) are also included in the signature matrices to indicate the detectability and isolatability condition of the faulty component by assuming single fault occurrence in the system. For instance, if $D_b = 1$ and $I_b = 1$ are mentioned for a particular component, then the fault in the component can be detected and isolated. For fault detection of any component, D_b must be 1. However, if the $I_b = 0$ and $D_b = 1$ then GFSSM/MCSSM generate the suspected set of faulty parameters (which have common fault signature in the matrices) with the possible fault directions. These fault directions can then be used in the parameter estimation process.

3.3.5 Proposed Method for Optimal Threshold and Mode Fault Detection

Sensitivity of residual to different faults and filtering out/countering uncertainties are important aspects of robust FDI in a monitored system. Detecting discrete mode fault is usually easier since it produces significant changes in the system dynamics but its isolation may be a challenging task in the case of unstructured fault signature matrix. On the other hand, detecting a small parametric fault requires an appropriate selection of residual threshold with the objective of less misdetection, small time delay in fault detection and low false alarm rate. Selection of an appropriate threshold is not a trivial task, since quantification of uncertainties effect in residual is difficult. In this regard, this section presents a method for optimal selection of adaptive thresholds by using the residuals from the known healthy system states in different working modes. This is supposed to be an experimental step. However, here, we only aim to demonstrate the procedure with the help of measurements generated from a will-fully disturbed model that replaces the real plant outputs. It

is shown in (3.12) and (3.13) that the residual can deviate in different directions according to increasing ($\theta_j \uparrow$) or decreasing ($\theta_j \downarrow$) variation of the parameters. For residual evaluation, we use the measured or estimated parameter with the assumption that it lies around the true value of the parameter of the system. We propose to use the residual from the known healthy system state and try to estimate the direction of deviation of the measured parameter value from its true value for optimal threshold selection. In addition to this, a mode fault identification technique is proposed in the case of unstructured fault signature matrix.

In case of a single parametric fault in j th parameter, $GARR_i$ equation may be approximated as

$$GARR_i(\mathbf{U}, \mathbf{Y}, \boldsymbol{\theta}, \mathbf{MD}) \simeq \left| \frac{\partial GARR_i}{\partial \theta_j^f} \Delta \theta_j^f \right| + \varepsilon_i \quad (3.17)$$

and in case of a single discrete mode fault, $GARR_i$ equation may be approximated as

$$GARR_i(\mathbf{U}, \mathbf{Y}, \boldsymbol{\theta}, \mathbf{MD}) \simeq \left| \frac{\partial GARR_i}{\partial a_k^f} \Delta a_k^f \right| + \varepsilon_i, \quad (3.18)$$

where optimum adaptive threshold is defined as $\varepsilon_i \simeq \pm \left| \sum_{l=1}^{n_u} \left(\alpha_l \frac{\partial GARR_i}{\partial \theta_l^u} \Delta \theta_l^u \right) \right|$, $\alpha_l \in \{0, +1, -1\}$ is the deviation direction of the l th uncertain parameter θ_l^u , $\Delta \theta_j^f$ and $\Delta \theta_l^u$ are the absolute deviations in the faulty parameter θ_j^f and the uncertain parameter θ_l^u , respectively, n_u is the number of uncertain parameters involved in the i th $GARR_i$, $|\Delta a_k^f| = 1$ is the absolute deviation of discrete mode fault a_k^f either changing from 0 to 1 or 1 to 0.

In the present work, we define three thresholds ε_i^{wc} , ε_i^{opt} and ε_i^{afa} where ε_i^{wc} denotes threshold based on worst condition of parameter uncertainty variation, ε_i^{opt} denotes the optimal threshold obtained from the proposed optimization technique and ε_i^{afa} denotes envelope of the evaluated residual using real measurements of the plant during known healthy state of the system. The envelope of a signal is a smooth curve outlining its extremes obtained with the aid of Hilbert transform of signal. Thus, residual signal has lower and upper envelopes. The upper envelope can be obtained as

$$\varepsilon_i^{afa}(t) = \sqrt{r_i^2(t) + \hat{r}_i^2(t)} \quad (3.19)$$

where $r_i(t)$ is the i th residual and $\hat{r}_i(t)$ is the Hilbert transform of i th residual.

For any real signal $r_i(t)$, Hilbert transform $\hat{r}_i(t)$ in the time interval $-\infty \leq t \leq \infty$ can be defined as [40]

$$\hat{r}_i(t) = (\text{p.v.}) \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{r_i(\tau)}{t - \tau} d\tau = r_i(t) * \frac{1}{\pi t}, \quad (3.20)$$

where $*$ denotes convolution operator and p.v. is the Cauchy principal value.

Usually, when the magnitude of a parameter value deviates outside its uncertainty bound then it is considered as a fault. Selecting threshold based on worst condition ensures no false alarm, but at the same time it reduces the detectability of small magnitude parametric faults. Thus, in the present work, optimum threshold is used for improving FDI. Then an optimal threshold $\varepsilon_i^{\text{opt}}$ ($\varepsilon_i^{\text{afa}} < \varepsilon_i^{\text{opt}} \leq \varepsilon_i^{\text{wc}}$) is selected based on minimization of the following objective function with the constraint condition that the threshold never intersects the envelope, $\varepsilon_i^{\text{afa}}(t)$, of the residual:

$$\min_{\alpha_l} J(\alpha_l) = \sum_{i=1}^n \frac{1}{2} \int_0^T \left(\frac{\varepsilon_i(t) - \varepsilon_i^{\text{afa}}(t)}{\varepsilon_{i \max}} \right)^2 dt, \quad (3.21)$$

subject to: $\varepsilon_i(t) - \varepsilon_i^{\text{afa}}(t) > 0$,

where T is time interval of nominal operation data of the system used in the optimization process, $\varepsilon_{i \max}$ is the maximum threshold value, n is the number of residuals and $\alpha_l \in \{0, +1, -1\}$.

For (3.8), by optimally enveloping the residual with a tighter threshold, the estimated threshold parameters are α_1, α_2 and α_3 , $\alpha_l \in \{0, +1, -1\}$, $l = 1, 2, 3$. The new threshold, $\varepsilon_1 \leq (\lambda_1 + \lambda_{S1})$, is then defined as

$$\varepsilon_1 \simeq \left| \left(-\alpha_1 \delta_{C_{T1}} C_{T1} \frac{d}{dt} (P_1) - \alpha_2 a_{V1} \delta_{C_{dV1}} C_{dV1} P_1 - \alpha_3 a_{L1} \delta_{C_{dL1}} C_{dL1} (P_1 - \rho g H_{L1}) \right) \right| \quad (3.22)$$

Using (3.12)–(3.15) on GARR_{n1} (given in (3.8)), the fault signatures for parametric faults and discrete mode faults for the hybrid tank system (Fig. 3.3) as presented in Tables 3.3 and 3.4, respectively, are obtained. Also, absolute values of residual sensitivity to different parameters are presented in the last columns of Tables 3.3 and 3.4, respectively. Note that only the valve stuck-on fault ($a_{V1} \uparrow$) and stuck-off fault ($a_{V1} \downarrow$) are taken into consideration as discrete mode faults for this system.

Table 3.3 GFSSM for the hybrid tank system

Parameter (θ_j)	$\text{GARR}_1(r_1)$	$\left \frac{\partial \text{GARR}_1}{\partial \theta_j} \right $
$C_{dV1} \uparrow$	$+a_{V1} \text{sign}(P_1(t))$	$ a_{V1}(P_1(t)) $
$C_{dV1} \downarrow$	$-a_{V1} \text{sign}(P_1(t))$	$ a_{V1}(P_1(t)) $
$C_{dL1} \uparrow$	$+a_{L1} \text{sign}(P_1(t) - \rho g H_{L1})$	$ a_{L1}(P_1(t) - \rho g H_{L1}) $
$C_{dL1} \downarrow$	$-a_{L1} \text{sign}(P_1(t) - \rho g H_{L1})$	$ a_{L1}(P_1(t) - \rho g H_{L1}) $
$C_{T1} \uparrow$	$+ \text{sign}\left(\frac{d}{dt}(P_1)\right)$	$\left \frac{d}{dt}(P_1) \right $
$C_{T2} \downarrow$	$- \text{sign}\left(\frac{d}{dt}(P_1)\right)$	$\left \frac{d}{dt}(P_1) \right $

Table 3.4 MCSSM for the hybrid tank system

Parameter (a_k)	GARR ₁ (r_1)	$\left \frac{\partial \text{GARR}_1}{\partial a_k} \right $
$a_{V_1} \uparrow$	$+\text{sign}(C_{dV_1}P_1(t))$	$ C_{dV_1}P_1(t) $
$a_{V_1} \downarrow$	$-\text{sign}(C_{dV_1}P_1(t))$	$ C_{dV_1}P_1(t) $

Autonomous mode (a_{L1}) is not considered as a source of discrete mode fault since it occurs due to internal changes in the states of the system and can be known by using the measurements from the system and set condition for its activation.

Suppose a partial blockage fault occurs in the valve V_1 ($C_{dV_1} \downarrow$) of the hybrid tank system (Fig. 3.3) and residual r_1 violates the adaptive threshold ε_1 . Under the single fault assumption, $C_{dV_1} \downarrow$ fault is not isolatable since residual GARR₁ is sensitive to all faults (C_{dV_1} , C_{dL1} , C_{T1} , and a_{V_1}) if mode $a_{L1} = 1$. Therefore, parameter estimation is needed for the unique single fault isolation. However, presence of the discrete mode fault (a_{V_1}) in the SSF complicates the parameter estimation task. In this regard, we propose to discriminate between parametric faults and the discrete mode faults by an initial hypothesis based on the magnitude of residual deviation after a fault. It is clear from the GFSSM and MCSSM (Tables 3.3 and 3.4) that the magnitude of residual sensitivities with respect to different parameters is not identical. A closer look at (3.8) shows that the residual would exceed the threshold (also refer (3.18) and Table 3.4) by at least a certain magnitude $|C_{dV_1}P_1|$ when there is discrete mode fault in valve V_1 , i.e., the value of $a_{V_1} = 0$ when it should have been 1 and vice versa.

Thus, in case of hypothesized discrete mode fault, we can rewrite (3.18) as

$$D_{a_k}^{r_i} = |\text{GARR}_i(\mathbf{U}, \mathbf{Y}, \boldsymbol{\theta}, \mathbf{MD})| - \left| \frac{\partial \text{GARR}_i}{\partial a_k^f} \Delta a_k^f \right| \leq \varepsilon_i, \quad (3.23)$$

where $D_{a_k}^{r_i}$ is the difference between absolute value of residual (r_i) deviation after a fault and sensitivity of residual with respect to suspected discrete parameter a_k .

Also, the initial relative deviation $\delta_{\theta_j}^f$ of each suspected parametric fault can be roughly estimated by using the initial residual deviation after a fault and its sensitivity with respect to parameter variation as presented in (3.17). Thus, the magnitude of relative deviation $\delta_{\theta_j}^f$ of the parametric fault θ_j^f can be roughly estimated as

$$\delta_{\theta_j}^f = \frac{|\text{GARR}_i(\mathbf{U}, \mathbf{Y}, \boldsymbol{\theta}, \mathbf{MD})|}{\left| \frac{\partial \text{GARR}_i}{\partial \theta_j^f} \theta_j^f \right|}, \text{ where } 0 \leq \delta_{\theta_j}^f \leq 1. \quad (3.24)$$

In an unstructured fault signature matrix, the initial estimated relative deviation $\delta_{\theta_j}^f$ of each suspected parameter can also be used to minimize the number of candidates in SSF. Such a minimization of elements in SSF reduces the computational burden during the parameter estimation. For example, if the magnitude of residual $|\text{GARR}_1|$ deviation after a blockage fault in the valve V_1 ($C_{dV_1} \downarrow$) of the hybrid

tank system (Fig. 3.3) is more than the absolute value $|(P_1(t) - \rho g H_{L1})|$ at the mode $a_{L1} = 1$ and $a_{V1} = 1$, then the estimated relative deviation $\delta_{C_{dL1}}^f$ for a suspected parameter is found to be more than one, which indicates C_{dL1} is faultless parameter since $0 \leq \delta_{C_{dL1}}^f \leq 1$ and therefore, it can be removed from the initial hypothesized SSF. This way, the magnitude of residual sensitivities with respect to other parameters in SSF can be tested to minimize the size of SSF.

Parameter estimation can be further improved by using the fault directions of SSF candidates obtained from GFSSM. For this, constrained parameter estimation technique is proposed for the suspected parametric faults by creating the proper bounds on the suspected parameters according to their known fault directions. These bounds are generated based on previous known nominal values of these parameters and their possible maximum deviations after a fault in the system, which are derived from the deep knowledge/understanding of the system and are called technological specifications. To further speed up the parameter estimation technique, a Sensitivity BG (SBG) approach [20] is used to supply the gradient information of the objective function during the parameter estimation process. Gradient projection method coupled with Gauss-Newton optimization technique is a very efficient constraint optimization technique with simple bounds on the parameters. The gradient projection method is more efficient, particularly, when constraints include only bounds on the parameters [41]. The minimization of objective function is expressed as:

$$\begin{aligned} \min_{\theta} J(\theta) &= \frac{1}{2} \sum_{j=k-q}^k \mathbf{r}^T(t_j) \cdot \mathbf{W} \cdot \mathbf{r}(t_j) \\ \text{subject to: } \theta_L &\leq \theta \leq \theta_U \end{aligned} \quad (3.25)$$

where $\mathbf{r}(t_j) = \mathbf{y}(t_j, \theta) - \hat{\mathbf{y}}(t_j)$ is a residual/error vector, $\mathbf{y}(t_j, \theta)$ is model's output vector and $\hat{\mathbf{y}}(t_j)$ is sensor's output vector at a time t_j , k is the sampled data of current instant, $q \geq 0$ is the sampled data of past time of fixed width, θ_L and θ_U are lower and upper parameter bounds, respectively, and $\mathbf{W} \in \mathbb{R}^{n \times n}$ is a positive semi-definite weighting function which can be assumed as a unit matrix.

It is assumed here that there is no measurement error and all the states are directly computable from the measurements. If the states cannot be computed, then the model initial conditions cannot be specified. In such situation, the optimization can be performed by minimizing the residuals obtained by evaluating the GARRs, which are derived from the BG model in derivative causality (see [19] for details).

3.4 Case Study: Bench Mark Hybrid Two-Tank System

In this section, the proposed diagnosis and thresholding techniques are applied on a two-tank benchmark example system shown in Fig. 3.9. The system consists of a Proportional-Integral (PI) controlled pump, two water tanks (T_1 and T_2) connected

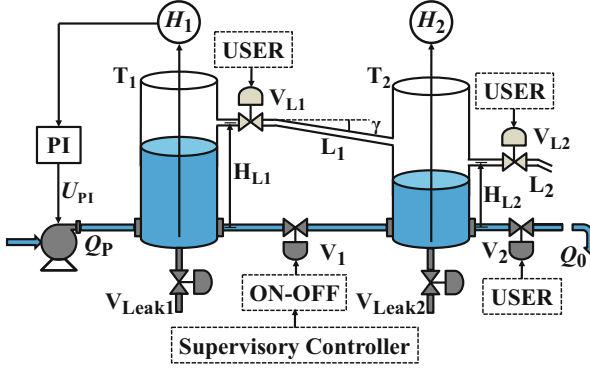


Fig. 3.9 Schematic of hybrid two-tank system

by a main pipeline with the nonlinear valve (V_1) and one main discharge pipeline with a nonlinear valve (V_2) from the tank T_2 . In addition, the system is equipped with one auxiliary drain pipeline (L_1) with linear valve (V_{L1}) between tanks T_1 and T_2 and one main drain pipeline (L_2) with linear valve (V_{L2}) from tank T_2 . The valve V_1 is controlled in open and closed state by a supervisory controller, whereas the valves V_2 , V_{L1} and V_{L2} are by default always in open state for this system and can be manually controlled by the user. The water level in tanks T_1 , T_2 and pump flow to tank T_1 are measured by the installed sensors $H_1(t)$, $H_2(t)$ and $Q_P(t)$, respectively.

The considered system includes the combination of both controlled and autonomous discrete events. For example, controlled discrete event occurs due to the external command signal (a_{V1}) given by the supervisory controller to the valve V_1 , while the two autonomous discrete events (a_{L1} and a_{L2} , respectively, for drainage of water from the upstream sides of V_{L1} and V_{L2}) occur at particular pre-set conditions (i.e., when level $H_1(t)$ exceeds height H_{L1} and level $H_2(t)$ exceeds height H_{L2} , respectively). In the conceptual benchmark system, two imaginary nonlinear valves V_{Leak1} and V_{Leak2} , respectively, are used to simulate the leakage faults in tanks T_1 and T_2 . The saturation characteristic of pump (Φ_P) and output law of PI-controller (Φ_{PI}) are, respectively, stated as

$$Q_P = \begin{cases} U_{PI}, & 0 \leq U_{PI} \leq f_{max} \\ 0, & U_{PI} \leq 0 \\ f_{max}, & U_{PI} \geq f_{max} \end{cases} = \Phi_P(U_{PI}), \quad (3.26)$$

$$\begin{aligned} U_{PI} &= K_P(S_{pt} - \rho \cdot g \cdot H_1(t)) + K_I \int (S_{pt} - \rho \cdot g \cdot H_1(t)) dt \\ &= \Phi_{PI}(H_1(t)), \end{aligned} \quad (3.27)$$

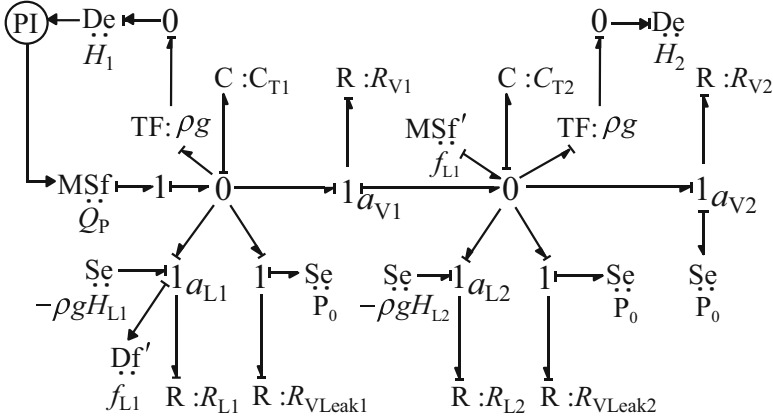


Fig. 3.10 HBG model of hybrid two-tank system

where f_{\max} is the maximum pump flow, U_{PI} is a controller output, S_{pt} is a controller set point, K_P is the proportional and K_I is the integral gain.

The HBG model of the considered hybrid system (Fig. 3.9) is shown in Fig. 3.10 where tank capacities for T_1 and T_2 ($C_{Ti} = A_i/g$, A_i is cross section area of tank, T_i , $i = 1, 2$) and valves (V_1 , V_2 , V_{L1} , V_{L2} , V_{Leak1} and V_{Leak2}) with connected pipes are modelled by two C -elements and six R -elements, respectively; and pump flow (Q_P) is modelled by Msf -element. Output sensors $H_1(t)$ and $H_2(t)$ are modelled by two De -elements. An imaginary detector Df' is also used to measure the flow (f_{L1}) through drain valve V_{L1} and provide the same flow by using an imaginary flow source $Msf' = f_{L1}$ to the other 0-junction corresponding to tank T_2 . In Fig. 3.10, 1-junctions with subscript a_{V1} , a_{V2} , a_{L1} and a_{L2} are switched junctions related with discrete modes. The flow through any resistor (R_{V1} , R_{V2} , R_{L1} and R_{L2}) at a switched junction is activated only in the active mode ($a_{Vi} = 1$) and is zero or absent in the inactive mode ($a_{Vi} = 0$).

3.4.1 ARR/GARRs for Hybrid Two-Tank System

The DHBG-LFT model of the hybrid two-tank system is shown in Fig. 3.11 where the two effort detectors $De:H_1$ and $De:H_2$ of HBG model (Fig. 3.10) are dualized into two sources $Mse':H_1$ and $Mse':H_2$. In addition, the nominal part of each parameter is also decoupled from its uncertain part and the different uncertain parts of the parameters are fed into the model at the respective junction by using the switched junction concept (1_{α_i}), where $\alpha_i \in \{0, +1, -1\}$ for generating the various possible thresholds (3^l). Two imaginary flow detectors ($Df^*:r_i \pm \lambda_i$) provide the two global constraint relations, i.e., GARR $_i$ with their uncertain parts λ_i ($i = 1, 2$).

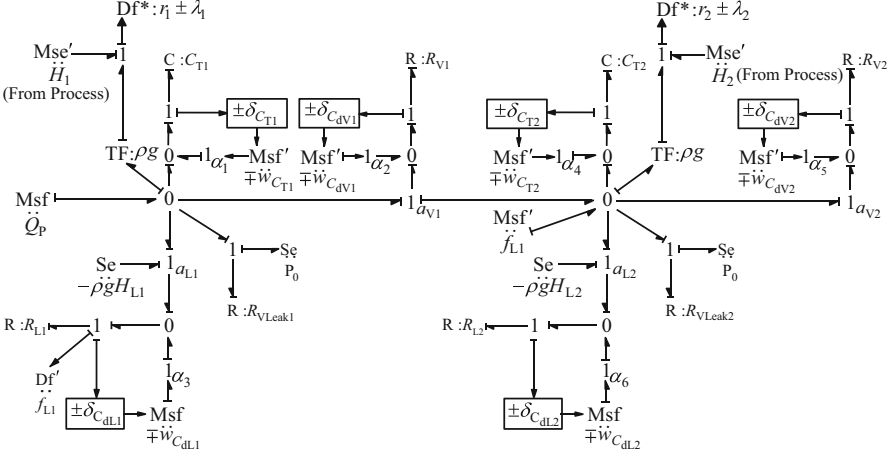


Fig. 3.11 DHBG-LFT form model of hybrid two-tank system

Established procedures for equation derivation from a causalled bond graph model [11–14, 42] are used to obtain the following GARRs and their uncertain parts for residuals and thresholds evaluation, respectively.

$$\begin{aligned} \text{GARR}_1 : Q_P - C_{T1} \frac{d}{dt} (\rho g H_1(t)) - a_{V1} C_{dV1} \sqrt{|\rho g (H_1(t) - H_2(t))|} \\ \cdot \text{sign}(H_1(t) - H_2(t)) - a_{L1} C_{dL1} \rho g (H_1(t) - H_{L1}) \\ - C_{dLeak1} \sqrt{|\rho g H_1(t)|} \pm \lambda_1 = 0 \end{aligned} \quad (3.28)$$

$$\begin{aligned} \text{GARR}_2 : a_{V1} C_{dV1} \sqrt{|\rho g (H_1(t) - H_2(t))|} \cdot \text{sign}((H_1(t) - H_2(t)) \\ + a_{L1} C_{dL1} \rho g (H_1(t) - H_{L1}) - C_{T2} \frac{d}{dt} (\rho g H_2(t)) - a_{V2} \\ \cdot C_{dV2} \sqrt{|\rho g H_2(t)|} - a_{L2} C_{dL2} \rho g (H_2(t) - H_{L2}) - C_{dLeak2} \\ \cdot \sqrt{|\rho g H_2(t)|} \pm \lambda_2 = 0 \end{aligned} \quad (3.29)$$

where $a_{L1} = \begin{cases} 0, & \text{if } H_1(t) \leq H_{L1} \\ 1, & \text{if } H_1(t) > H_{L1} \end{cases}$, $a_{L2} = \begin{cases} 0, & \text{if } H_2(t) \leq H_{L2} \\ 1, & \text{if } H_2(t) > H_{L2} \end{cases}$.

The general form of adaptive thresholds ε_1 and ε_2 , respectively, for the residuals r_1 and r_2 can be obtained by using uncertain parts λ_1 and λ_2 as

$$\begin{aligned} \varepsilon_1 = \lambda_1 = \left| -\alpha_1 \delta_{C_{T1}} C_{T1} \frac{d}{dt} (\rho g H_1(t)) - \alpha_2 a_{V1} \delta_{C_{dV1}} C_{dV1} \sqrt{|\rho g (H_1(t) - H_2(t))|} \right. \\ \left. - \alpha_3 a_{L1} \delta_{C_{dL1}} C_{dL1} \rho g (H_1(t) - H_{L1}) \right| \end{aligned} \quad (3.30)$$

$$\varepsilon_2 = \lambda_2 = \left| \alpha_2 a_{V1} \delta_{C_{dV1}} C_{dV1} \sqrt{|\rho g(H_1(t) - H_2(t))|} + \alpha_3 a_{L1} \delta_{C_{dL1}} C_{dL1} \rho g(H_1(t) - H_{L1}) - \alpha_4 \delta_{C_{T2}} C_{T2} \frac{d}{dt} (\rho g H_2(t)) - \alpha_5 a_{V2} \delta_{C_{dV2}} C_{dV2} \sqrt{|\rho g H_2(t)|} - \alpha_6 a_{L2} \delta_{C_{dL2}} C_{dL2} \rho g(H_2(t) - H_{L2}) \right| \quad (3.31)$$

where $\alpha_l \in \{0, +1, -1\}$, $l = 1, 2, \dots, 6$.

In this study, we assume that sensors, actuators (pump) and the controllers (PI-controller) are faultless. Sensor fault detection can be done using hardware redundancy [7]. FDI of pump and PI-controller may be done separately by using the following ARR_s derived from their characteristics relationships

$$\text{ARR}_3 : Q_P - \Phi_P(U_{PI}) = 0, \quad (3.32)$$

$$\text{ARR}_4 : U_{PI} - \Phi_{PI}(H_1(t)) = 0. \quad (3.33)$$

3.4.2 Optimum Adaptive Threshold for Hybrid Two-Tank System

The objective function for the optimum threshold for hybrid two-tank system is expressed as

$$\min_{\alpha_i} J(\alpha_i) = \sum_{i=1}^2 \frac{1}{2} \int_0^T \left(\frac{\varepsilon_i(t) - \varepsilon_i^{\text{afa}}(t)}{\varepsilon_{i \max}} \right)^2 dt, \quad (3.34)$$

subject to: $\varepsilon_i(t) - \varepsilon_i^{\text{afa}}(t) > 0$,

where $\alpha_l \in \{0, +1, -1\}$, $l = 1, 2, \dots, 6$, $i = 1, 2$.

For the selection of optimum thresholds, our main objective is to find the exact deviation direction of the respective parameter from its true nominal value used in the threshold evaluations as given in (3.30) and (3.31), respectively. For this, the residuals evaluated using the measurements of the real healthy system (from real experiment) are required so that the optimum thresholds out of many possible thresholds (3^l) can be selected. However, in the present work, the measurements have been generated from the simulation of a model in which the parameter values have been purposefully perturbed a little (within the uncertainty limit) from the corresponding nominal values. The HBG model is converted into the corresponding Matlab-Simulink model for simulation and then the obtained measurements are used in another simple Matlab-Simulink program to solve the optimization problem (3.34). The nominal parameters used in the unperturbed Simulink model are presented in Table 3.5. Furthermore, to realize parameter and measurement uncertainty, each parameter and the measurement have been deviated within their

Table 3.5 Parameters used in the simulation model

Symbol	Description	Parameter value
K_P	Proportional gain of controller	1 ms
K_I	Integral gain of controller	5×10^{-2} m
S_{pt}	Set point of the PI-controller	0.5 m
f_{max}	Maximum outflow from pump	1.5 kg/s
A_i	Cross-sectional area of tank T_i ($i = 1, 2$)	2.16×10^{-2} m ²
C_{dVi}	Discharge coefficient of valve V_i including connected pipeline ($i = 1, 2$)	1.593×10^{-2} kg ^{1/2} m ^{1/2}
C_{dLi}	Discharge coefficient of valve V_{Li} including connected drain pipeline ($i = 1, 2$)	1×10^{-3} ms
C_{dLeaki}	Discharge coefficient of V_{Leaki} ($i = 1, 2$)	0 kg ^{1/2} m ^{1/2}
H_{L1}	Height of the drain pipe L_1 of tank T_1 from datum	0.58 m
H_{L2}	Height of the drain pipe L_2 of tank T_2 from datum	0.40 m
P_0	Atmospheric pressure	0 N/m ²
ρ	Density of water	1000 kg/m ³
g	Acceleration due to gravity	9.81 m/s ²

uncertainties bounds in the Simulink model as $\delta_{C_{T1}} = \delta_{C_{T2}} = \delta_{C_{dV1}} = \delta_{C_{dV2}} = \delta_{C_{dL1}} = \delta_{C_{dL2}} \leq |0.05|$ and maximum 2% sensor noise $\delta_{H_1} = \delta_{H_2} = \delta_{Q_P} \leq |0.02|$, respectively. These changes are randomly effected and the following results correspond to one such randomly perturbed case. Note that the parameter values are perturbed for real plant measurement generation whereas the residuals and thresholds are evaluated assuming the nominal parameter values. As per definition, with all the small parameter deviations (uncertainties) effected to the plant model, the system is supposedly in healthy state and the residuals are expected to remain bounded within respective adaptive thresholds.

The system model is simulated in a faultless condition for a simulation time duration of 300 s by using a fixed step size of 0.02 s and by assigning all initial state values to zero. All the measured input (Q_P) and outputs (H_1 and H_2) values of the healthy operating system along with the information of controlled mode (open for 80 s and closed for 30 s periodically), autonomous modes and known parameter values are fed into the developed threshold optimization program.

After optimization, deviation directions of the parameters are obtained as $\alpha_1 = +1$, $\alpha_2 = +1$, $\alpha_3 = +1$, $\alpha_4 = -1$, $\alpha_5 = 0$ and $\alpha_6 = 0$ corresponding to optimal thresholds out of 3^6 or 729 number of different generated thresholds (considering all possible deviation directions of the parameters). The responses of residuals r_1 and r_2 along with their envelopes ε_1^{afa} and ε_2^{afa} , obtained optimum adaptive thresholds ε_1^{opt} and ε_2^{opt} , different generated thresholds and the worst condition adaptive thresholds ε_1^{wc} and ε_2^{wc} are plotted in Fig. 3.12 for the duration of 60–140 s for better clarity. It is clear from Fig. 3.12 that the optimum thresholds ε_1^{opt} and ε_2^{opt} selected as per estimated variations of the residuals for the system provide better detectability and avoid false alarms in corresponding modes as compared to the classical worst condition based adaptive thresholds ε_1^{wc} and ε_2^{wc} . Thus, the obtained

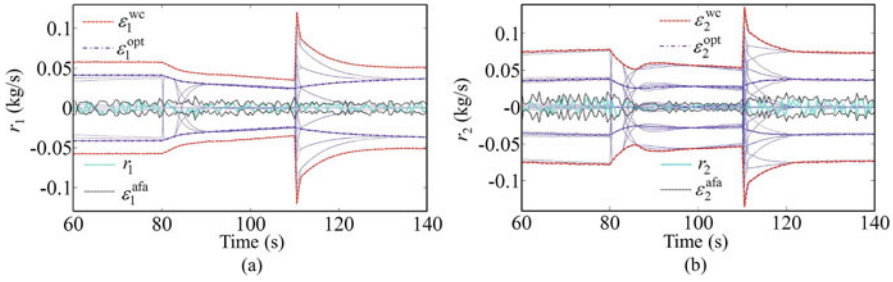


Fig. 3.12 Response of (a) residual r_1 (b) residual r_2 with optimum adaptive thresholds along with the worst condition adaptive thresholds during nominal operation

Table 3.6 Injected fault cases in the simulation model

Case	Parameter	Description	Nominal value	Faulty value	Start time (s)	End time (s)
1	$C_{dV2} \downarrow$	Blockage in V_2	C_{dV2}	$0.93C_{dV2}$	150	300
2	$\alpha_{V1} \downarrow$	Stuck-off fault in V_1	1	0	220	300

optimum thresholds ε_1^{opt} and ε_2^{opt} are finally selected for the FDI study of hybrid two-tank system for detecting parametric faults or discrete mode faults in the system. However, the other generated thresholds either give the false alarm in a particular mode or may be wider than optimum thresholds resulting in mis-detection; thus, the optimization algorithm does not select them for robust fault diagnosis. Since the parameter values may deviate further in arbitrary directions after further long time operation, it is advisable to update the thresholds regularly at pre-set time intervals, say in few days or weeks, depending on the technological specifications.

3.4.3 FDI Study for Hybrid Two-Tank System Using Proposed Technique

In this section, the proposed technique for discrete mode fault and parametric fault identification is applied to the hybrid two-tank system. Two different fault scenarios related to a parametric fault and a discrete mode fault, as presented in Table 3.6, have been tested. Applying (3.12)–(3.15) on (3.28) and (3.29), the GFSSM and MCSSM for the hybrid two-tank system, as given in Table 3.7, are obtained. For the parameters C_{dLeak1} and C_{dLeak2} related to leakage fault in tank T_1 and T_2 , respectively, only increasing possibility is considered from a practical viewpoint (technological specifications), whereas for the other parameters (i.e., C_{dV1} , C_{dV2} , C_{dL1} , C_{dL2}), both increasing (i.e., leakage) and decreasing possibilities (i.e., blockage) are considered. In order to demonstrate the discrete mode fault identification based on magnitude of residual deviation after a fault, possibilities of discrete mode faults in all the valves V_1 , V_2 , V_{L1} and V_{L2} are considered.

Table 3.7 Dynamic fault signature matrices for the two-tank system

	Parameter	General case			Specific case (0–300 s)			
		r_1	r_2	D_b	r_1	r_2	D_b	I_b
GFSSM	$C_{dV1} \uparrow$	$+a_{V1} \text{sign}(H_1 - H_2)$	$-a_{V1} \text{sign}(H_1 - H_2)$	a_{V1}	$+a_{V1}$	$-a_{V1}$	a_{V1}	0
	$C_{dV1} \downarrow$	$-a_{V1} \text{sign}(H_1 - H_2)$	$+a_{V1} \text{sign}(H_1 - H_2)$	a_{V1}	$-a_{V1}$	$+a_{V1}$	a_{V1}	0
	$C_{dV2} \uparrow$	0	$+\text{sign}(H_2)$	1	0	+1	1	0
	$C_{dV2} \downarrow$	0	$-\text{sign}(H_2)$	1	0	-1	1	0
	$C_{dL1} \uparrow$	$+a_{L1}$	$-a_{L1}$	a_{L1}	$+a_{L1}$	$-a_{L1}$	a_{L1}	0
	$C_{dL1} \downarrow$	$-a_{L1}$	$+a_{L1}$	a_{L1}	$-a_{L1}$	$+a_{L1}$	a_{L1}	0
	$C_{dL2} \uparrow$	0	$+a_{L2}$	a_{L2}	0	$+a_{L2}$	a_{L2}	0
	$C_{dL2} \downarrow$	0	$-a_{L2}$	a_{L2}	0	$-a_{L2}$	a_{L2}	0
	$C_{dLeak1} \uparrow$	+1	0	1	+1	0	1	0
	$C_{dLeak2} \uparrow$	0	+1	1	0	+1	1	0
MCSSM	$a_{V1} \uparrow$	$+\text{sign}(H_1 - H_2)$	$-\text{sign}(H_1 - H_2)$	$1 - a_{V1}$	+1	-1	$1 - a_{V1}$	0
	$a_{V1} \downarrow$	$-\text{sign}(H_1 - H_2)$	$+\text{sign}(H_1 - H_2)$	a_{V1}	-1	+1	a_{V1}	0
	$a_{V2} \downarrow$	0	$-\text{sign}(H_2)$	a_{V2}	0	-1	a_{V2}	0
	$a_{L1} \downarrow$	$-\text{sign}(H_1)$	$+\text{sign}(H_1)$	a_{L1}	-1	+1	a_{L1}	0
	$a_{L2} \downarrow$	0	$-\text{sign}(H_2)$	a_{L2}	0	-1	a_{L2}	0

For example, for the valve V_1 , both discrete stuck-on fault ($a_{V1} \uparrow$) and stuck-off fault ($a_{V1} \downarrow$) possibilities are considered. For the other valves (V_2 , V_{L1} and V_{L2} , which are by default always open for this system), we assume that the valve can be mistakenly closed by the plant operator during operation and may be considered as a fault in such a situation. All these possibilities considered in Table 3.7, called technological specifications, are derived from the deep understanding of the system and these possibilities can vary with the type of the system.

The responses of the residuals (r_1 , r_2) and adaptive thresholds ($\varepsilon_1^{\text{opt}}$, $\varepsilon_2^{\text{opt}}$ and $\varepsilon_1^{\text{wc}}$, $\varepsilon_2^{\text{wc}}$) along with input (Q_P), output measurements (H_1 , H_2) and the known modes information (a_{V1} , a_{L1}) of the system in both the fault cases are presented in Figs. 3.13 and 3.14, respectively. The predicted autonomous mode a_{L1} activation using measurement $H_1(t)$ according to pre-set condition for valve V_{L1} (i.e., when $H_1(t) > H_{L1} = 0.58$ m) for both the cases is shown in Figs. 3.13h and 3.14h. However, no autonomous mode a_{L2} activation is noticed at pre-set condition for drain valve V_{L2} (i.e., when $H_2(t) > H_{L2} = 0.40$ m) in both cases. Therefore, $a_{L2} = 0$ throughout the observation period of 0–300 s. It is also observed from Figs. 3.13 and 3.14 that the output measurement $H_1(t)$ is always greater than the measurement $H_2(t)$ in both the fault cases in the observation period of 0–300 s. Thus, the dynamic fault signature matrices (GFSSM and MCSSM) presented in Table 3.7, as general case, can be represented by the static value, as specific case, for the considered duration of observation. In the table, the detectability index D_b is a binary number whose values 1 and 0, respectively, indicate sensitivity and insensitivity of at least one residual to the change in the parameter. The isolatability index I_b is another binary number whose values 1 and 0, respectively, indicate

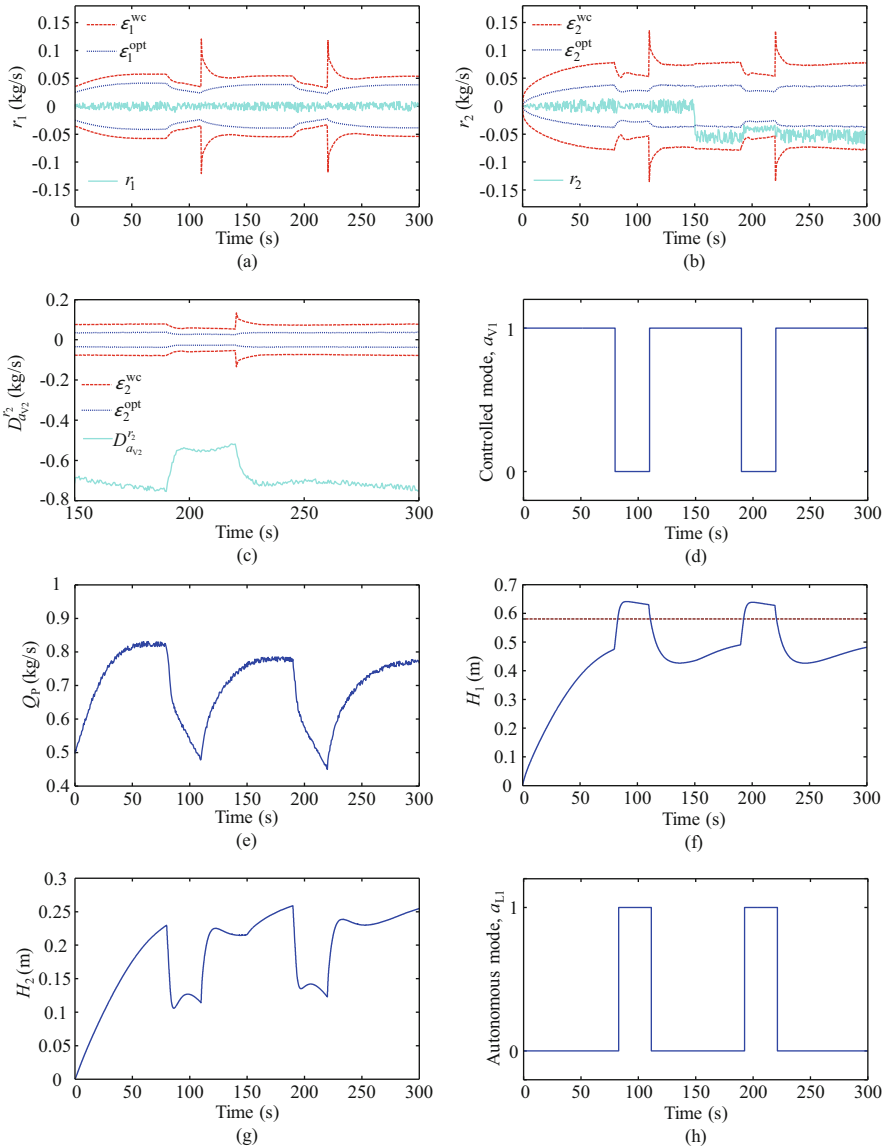


Fig. 3.13 Response of (a) Residual r_1 . (b) Residual r_2 . (c) Difference, $D_{a_{V2}}^{r_2}$ using residual r_2 and its sensitivity with respect to a_{V2} , after a blockage fault in valve V_2 . (d) Controlled mode a_{V1} . (e) Input Q_p . (f) Level H_1 . (g) Level H_2 . (h) Predicted autonomous mode a_{L1} activation using measurement H_1

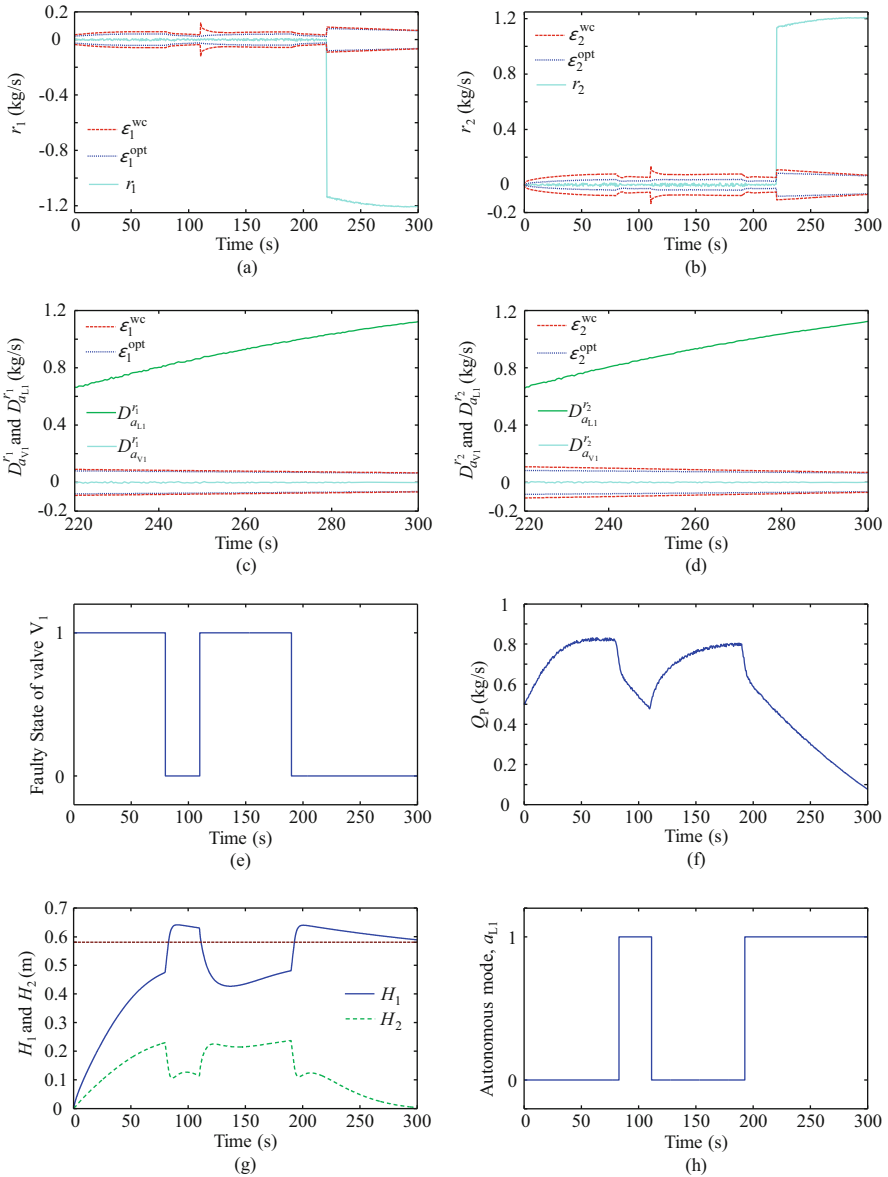


Fig. 3.14 Response of (a) Residual r_1 . (b) Residual r_2 . (c, d) Differences, $D_{a_{V1}}^{r_1}$, $D_{a_{L1}}^{r_1}$ and $D_{a_{V1}}^{r_2}$, $D_{a_{L1}}^{r_2}$ using residuals r_1 and r_2 and their sensitivities with respect to a_{V1} and a_{L1} , respectively, after a valve stuck-off fault in V_1 . (e) Faulty state of valve V_1 . (f) Input Q_p . (g) Levels H_1 and H_2 . (h) Predicted autonomous mode a_{L1} activation using measurement H_1

uniqueness and multiplicity (ambiguity) of the fault signature of the parameter. Note that the absolute value of sensitivity fault signatures in GFSSM/MCSSM, as presented in Table 3.7 (Specific case), provides the standard GFSM/MCSM in terms of binary signatures without deviation sign. It can be seen that the GFSSM/MCSSM provides better fault isolation as compared to standard GFSM/MCSM. For example, the blockage fault in $C_{dV2} \downarrow$ is distinguishable from leakage fault $C_{dLeak2} \uparrow$ if using GFSSM, but with standard GFSM, these faults ($C_{dV2} \downarrow$ and $C_{dLeak2} \uparrow$) are indistinguishable from each other.

In the fault case-1 (i.e. small fault in parameter C_{dV2}), it is observed from Fig. 3.13a, b that the residual r_1 which is insensitive to this fault remains bounded inside the adaptive thresholds $\varepsilon_1^{\text{opt}}$ and $\varepsilon_1^{\text{wc}}$, whereas the residual r_2 which is sensitive to this fault deviates outside the optimum adaptive threshold $\varepsilon_2^{\text{opt}}$. However, using worst condition adaptive threshold $\varepsilon_2^{\text{wc}}$ in the same fault situation, this fault cannot be detected by the diagnosis module and thus, it leads to a missed detection of the small fault even-though the parameter has changed more than the uncertainty limit. This missed detection is obvious, since the worst condition adaptive threshold $\varepsilon_2^{\text{wc}}$ is broader as compared to optimum adaptive threshold $\varepsilon_2^{\text{opt}}$. Thus, optimum adaptive thresholds appear to provide better detectability of a fault as compared to classical thresholds. After detecting a fault, the next step in the diagnosis module is to isolate the true fault and estimate its severity (fault magnitude) so that the decision regarding the fault accommodation or system reconfiguration can be taken. Just after 150s, the coherence vector $\mathbf{CV} = [0, -1]$ (see Fig. 3.13a, b and residuals crossing the optimal thresholds). After matching the $\mathbf{CV} = [0, -1]$ with the dynamic fault signature matrices in Table 3.7 (Specific Case), the SSF is obtained as $\{C_{dV2} \downarrow, a_{V2} \downarrow\}$. Note that $C_{dL2} \downarrow$ and $a_{L2} \downarrow$ are not included in SSF since measurements indicate $a_{L2} = 0$. According to obtained SSF, $C_{dV2} \downarrow$ is not directly isolatable by the diagnosis module as discrete mode fault $a_{V2} \downarrow$ shares the same signature as $C_{dV2} \downarrow$. Thus, before estimating the fault magnitude of $C_{dV2} \downarrow$, we need to first confirm whether the residual inconsistency is due to a discrete mode fault $a_{V2} \downarrow$ or due to a parametric fault $C_{dV2} \downarrow$. For this, as per (3.23), $D_{a_{V2}}^r = |r_2| - \left| C_{dV2} \sqrt{|\rho g H_2(t)|} \right| \leq \varepsilon_2^{\text{opt}}$ is tested with the optimum adaptive threshold $\varepsilon_2^{\text{opt}}$ just after fault detection (see Fig. 3.13c). It is found that $D_{a_{V2}}^r > \varepsilon_2^{\text{opt}}$, thus, the hypothesis of discrete mode fault $a_{V2} \downarrow$ is proved to be wrong. This indicates that the $C_{dV2} \downarrow$ is the actual fault. In order to estimate the severity of $C_{dV2} \downarrow$ fault, a constrained parameter estimation technique as proposed in (3.25) is triggered with the parameter bound ($0 < C_{dV2} < 0.01593$) and this quickly estimates the fault magnitude of C_{dV2} as $C_{dV2}^f \simeq 0.93C_{dV2}$. If more fault candidates remain in the SSF after discarding the mode fault from the initial hypothesis then parameter estimation technique can be appropriately adapted [5, 20].

In the fault case-2 (i.e. valve V_1 stuck-off fault, $a_{V1} \downarrow$), it is observed from Fig. 3.14a, b that both the residuals r_1 and r_2 , which are sensitive to this fault, deviate outside their respective optimum adaptive thresholds ($\varepsilon_1^{\text{opt}}$ and $\varepsilon_2^{\text{opt}}$) as well as worst condition adaptive thresholds ($\varepsilon_1^{\text{wc}}$ and $\varepsilon_2^{\text{wc}}$). Just after 220 s, the coherence vector $\mathbf{CV} = [-1, +1]$ (see Fig. 3.14a, b). After matching the $\mathbf{CV} = [-1, +1]$

with the dynamic fault signature matrices in Table 3.7 (Specific Case), the SSF is obtained as $\{C_{dV1} \downarrow, C_{dL1} \downarrow, a_{V1} \downarrow, a_{L1} \downarrow\}$. Note that at the time of fault inception, measurements indicate that mode a_{L1} should be 1 (see Fig. 3.14h), but it is still possible that someone inadvertently closed the valve leading to $a_{L1} \downarrow$. According to obtained SSF, $a_{V1} \downarrow$ is not directly isolatable by the diagnosis module, since all the SSF candidates share the common fault signature. However, the number of SSF candidates can be minimized by roughly estimating the initial relative deviation of each suspected parameter using (3.24). In this case, $\delta_{C_{dL1}}^f$ is found to be more than one. Since $0 \leq \delta_{C_{dL1}}^f \leq 1$ is mandatory, $C_{dL1} \downarrow$ is removed from the SSF and the refined SSF can be written as $\{C_{dV1} \downarrow, a_{V1} \downarrow, a_{L1} \downarrow\}$. Again a discrete mode fault identification technique as proposed in (3.23) is triggered first, i.e. the differences $D_{a_{V1}}^r = |r_1| - \left| C_{dV1} \sqrt{|\rho g(H_1(t) - H_2(t))|} \right|$, $D_{a_{V1}}^2 = |r_2| - \left| C_{dV1} \sqrt{|\rho g(H_1(t) - H_2(t))|} \right|$ and $D_{a_{L1}}^r = |r_1| - |C_{dL1} \rho g(H_1(t) - H_{L1})|$, $D_{a_{L1}}^2 = |r_2| - |C_{dL1} \rho g(H_1(t) - H_{L1})|$ for the $a_{V1} \downarrow$ and $a_{L1} \downarrow$, respectively, are tested with the optimum adaptive thresholds just after fault detection (shown in Fig. 3.14c, d, respectively). It is observed from Fig. 3.14c, d that $D_{a_{V1}}^r \leq \varepsilon_1^{\text{opt}}$ and $D_{a_{V1}}^2 \leq \varepsilon_2^{\text{opt}}$, which indicates that $a_{V1} \downarrow$ is the faulty discrete parameter under the single fault assumption. This discrete mode fault $a_{V1} \downarrow$ has severe impact on the behaviour of the system, which can be observed from the magnitude of residuals deviations after this fault.

Thus, the proposed technique is shown to effectively detect and isolate both discrete mode faults and parametric faults in the HDS under single fault situations. Without loss of generality, the dynamically updated modes and parameters to envelope residuals after occurrence of a fault as proposed in [43] can be applied to extend the approach presented here to handle the case of multiple sequential faults.

3.5 Conclusions

In this chapter, a common bond graph modelling framework is used for hybrid system modelling, its simulation, GARRs and thresholds equations derivation, optimum adaptive threshold selection and the rule development for discrete and parametric fault detection and isolation. Optimal adaptive thresholds are chosen by estimating the deviation directions of the parameters or uncertainty direction with aid of real plant measurement data and threshold equations generated from DHBG-LFT form model. The selected optimal thresholds properly bound the real uncertain residuals and thereby, avoid unnecessary false alarms and improve the fault detectability in comparison to the classically used worst condition adaptive thresholds. GFSSM and MCSSM, which have better fault discrimination capability, are adopted for fault isolation [27]. Moreover, a new technique is proposed to discriminate the parametric faults from the discrete mode faults by an initial hypothesis based on magnitude of residual deviation after a fault. It is observed from

the simulation results that different residual sensitivities with respect to different parameter variations can be effectively utilized to discriminate one fault effect from the other. It is also observed that the use of fault direction information from GFSSM and imposition of proper bounds on the parameter deviations improves the parameter estimation method by reducing the parameter search zone.

References

1. Gertler, J. (1998). *Fault detection and diagnosis in engineering systems*. New York: Dekker. ISBN 0-8247-9427-3.
2. Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M., & Schröder, J. (2006). *Diagnosis and fault-tolerant control*. Berlin: Springer.
3. Chen, J., & Patton, R. J. (2012). *Robust model-based fault diagnosis for dynamic systems*. Berlin: Springer Science Business Media.
4. Gelso, E. R., Biswas, G., Castillo, S., & Armengol, J. (2008, September). A comparison of two methods for fault detection: A statistical decision, and an interval-based approach. In *19th International Workshop on Principles of Diagnosis DX* (pp. 261–268).
5. Borutzky, W. (2015). *Bond graph model-based fault diagnosis of hybrid systems*. Cham, Switzerland: Springer.
6. Mukherjee, A., Karmakar, R., & Samantaray, A. K. (2006). *Bond graph in modelling, simulation and fault identification*. New Delhi: I. K. International Pvt. Ltd.
7. Samantaray, A. K., & Ould Bouamama, B. (2008). *Model-based process supervision: A bond graph approach*. London: Springer.
8. Karnopp, D. C., Margolis, D. L., & Rosenberg, R. C. (2012). *System dynamics: Modelling, simulation, and control of mechatronic systems* (5th ed.). Hoboken, NJ: Wiley.
9. Mosterman, P. J., & Biswas, G. (1995). Behaviour generation using model switching: A hybrid bond graph modelling technique. *Transactions of The Society for Computer Simulation*, 27(1), 177–182.
10. Roychoudhury, I., Daigle, M. J., Biswas, G., & Koutsoukos, X. (2011). Efficient simulation of hybrid systems: A hybrid bond graph approach. *Simulation*, 87(6), 467–498.
11. Ghoshal, S. K., Samanta, S., & Samantaray, A. K. (2012). Robust fault detection and isolation of hybrid systems with uncertain parameters. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 226(8), 1013–1028.
12. Wang, D., Yu, M., Low, C. B., & Arogeti, S. (2013). *Model-based health monitoring of hybrid systems*. New York: Springer Science Business Media.
13. Djeziri, M. A., Merzouki, R., Ould Bouamama, B., & Dauphin-Tanguy, G. (2007). Robust fault diagnosis by using bond graph approach. *IEEE/ASME Transactions on Mechatronics*, 12(6), 599–611.
14. Merzouki, R., Samantaray, A. K., Pathak, P. M., & Ould Bouamama, B. (2012). *Intelligent mechatronic systems: Modelling, control and diagnosis*. London: Springer.
15. Touati, Y., Merzouki, R., & Ould Bouamama, B. (2012). Robust diagnosis to measurement uncertainties using bond graph approach: Application to intelligent autonomous vehicle. *Mechatronics*, 22(8), 1148–1160.
16. Arogeti, S., Wang, D., & Low, C. B. (2010). Mode identification of hybrid systems in the presence of fault. *IEEE Transactions on Industrial Electronics*, 57(4), 1452–1467.
17. Daigle, M., Bregon, A., & Roychoudhury, I. (2016). A qualitative fault isolation approach for parametric and discrete faults using structural model decomposition. In *Annual Conference of PHMS*.
18. Prakash, O., & Samantaray, A. K. (2017). Model-based diagnosis and prognosis of hybrid dynamical systems with dynamically updated parameters. In *Bond graphs for modelling, control and fault diagnosis of engineering systems* (pp. 195–232). Cham, Switzerland: Springer.

19. Samantaray, A. K., Ghoshal, S. K., Chakraborty, S., & Mukherjee, A. (2005). Improvements to single-fault isolation using estimated parameters. *Simulation*, *81*(12), 827–845.
20. Samantaray, A. K., & Ghoshal, S. K. (2007). Sensitivity bond graph approach to multiple fault isolation through parameter estimation. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *221*(4), 577–587.
21. Low, C. B., Wang, D., Arogeti, S. A., & Luo, M. (2009, July). Fault parameter estimation for hybrid systems using hybrid bond graph. In *Control Applications, (CCA) Intelligent Control, (ISIC), 2009 IEEE* (pp. 1338–1343). New York: IEEE.
22. Bregon, A., Biswas, G., & Pulido, B. (2012). A decomposition method for nonlinear parameter estimation in TRANSCEND. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *42*(3), 751–763.
23. Jha, M. S., Dauphin-Tanguy, G., & Ould Bouamama, B. (2014, June). Robust FDI based on LFT BG and relative activity at junction. In *European on Control Conference (ECC)* (pp. 938–943). New York: IEEE.
24. Touati, Y., Mellal, M. A., & Benazzouz, D. (2016). Multi-thresholds for fault isolation in the presence of uncertainties. *ISA Transactions*, *62*, 299–311.
25. Khorasgani, H., Eriksson, D., Biswas, G., Frisk, E., & Krysl, M. (2014). Off-line robust residual selection using sensitivity analysis. 25th International Workshop on Principles of Diagnosis (DX-14). Graz, Austria, September 8–11, 2014. <https://doi.org/10.13140/2.1.3090.8809>.
26. Borutzky, W. (2014). Bond graph model-based system mode identification and mode-dependent fault thresholds for hybrid systems. *Mathematical and Computer Modelling of Dynamical Systems*, *20*(6), 584–615.
27. Levy, R., Arogeti, S., Wang, D., & Fivel, O. (2015). Improved diagnosis of hybrid systems using instantaneous sensitivity matrices. *Mechanism and Machine Theory*, *91*, 240–257.
28. Alonso, N. M., Bregon, A., Alonso-González, C. J., & Pulido, B. (2013, September). A common framework for fault diagnosis of parametric and discrete faults using possible conflicts. In *Conference of the Spanish Association for Artificial Intelligence* (pp. 239–249). Berlin, Heidelberg: Springer.
29. Bregon, A., González, C. A., & Pulido, B. (2015). Improving fault isolation and identification for hybrid systems with hybrid possible conflicts. In *DX@ Safeprocess* (pp. 59–66).
30. Vento, J., Blesa, J., Puig, V., & Sarrate, R. (2015). Set-membership parity space hybrid system diagnosis. *International Journal of Systems Science*, *46*(5), 790–807.
31. Rahal, M. I., Ould Bouamama, B., & Meghebar, A. (2016, May). Hybrid bond graph for robust diagnosis to measurement uncertainties. In *2016 5th International Conference on Systems and Control (ICSC)* (pp. 439–444). New York: IEEE.
32. Louajri, H., & Sayed-Mouchaweh, M. (2014, September). Decentralized approach for fault diagnosis of three cell converters. In *Annual Conference of the Prognostics and Health Management Society*, Fort Worth, TX, USA (pp. 265–277).
33. Toubakh, H., & Sayed-Mouchaweh, M. (2016). Hybrid dynamic classifier for drift-like fault diagnosis in a class of hybrid dynamic systems: Application to wind turbine converters. *Neurocomputing*, *171*, 1496–1516.
34. Kościelny, J. M. (1995). Fault isolation in industrial processes by the dynamic table of states method. *Automatica*, *31*(5), 747–753.
35. Cordier, M. O., Dague, P., Lévy, F., Montmain, J., Staroswiecki, M., & Travé-Massuyés, L. (2004). Conflicts versus analytical redundancy relations: A comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *34*(5), 2163–2177.
36. Puig, V., Schmid, F., Quevedo, J., & Pulido, B. (2005, December). A new fault diagnosis algorithm that improves the integration of fault detection and isolation. In *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, Seville, Spain (pp. 3809–3814).

37. Petti, T. F., Klein, J., & Dhurjati, P. S. (1990). Diagnostic model processor: Using deep knowledge for process fault diagnosis. *AIChE Journal*, *36*(4), 565–575.
38. Samantaray, A. K., Medjaher, K., Ould Bouamama, B., Staroswiecki, M., & Dauphin-Tanguy, G. (2006). Diagnostic bond graphs for online fault detection and isolation. *Simulation Modelling Practice and Theory*, *14*(3), 237–262.
39. Low, C. B., Wang, D., Arogeti, S., & Luo, M. (2010). Quantitative hybrid bond graph-based fault detection and isolation. *IEEE Transactions on Automation Science and Engineering*, *7*(3), 558–569.
40. Brandt, A. (2011). *Noise and vibration analysis: Signal analysis and experimental procedures*. New York: Wiley.
41. Nocedal, J., & Wright, S. (2006). *Numerical optimization*. New York: Springer Science and Business Media.
42. Oukl Bouamama, B., Staroswiecki, M., & Samantaray, A. K. (2006). Software for supervision system design in process engineering industry. *IFAC Proceedings Volumes*, *39*(13), 646–650.
43. Prakash, O., Samantaray, A. K., & Bhattacharyya, R. (2017). Model-based diagnosis of multiple faults in hybrid dynamical systems with dynamically updated parameters. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. <https://doi.org/10.1109/TSMC.2017.2710143>

Chapter 4

Diagnosing Hybrid Dynamical Systems Using Max-Plus Algebraic Methods



Gregory Provan

4.1 Introduction

A hybrid system (HS) is a model whose evolution is governed by both continuous and discrete dynamics. A key challenge of diagnosing hybrid systems is the difficulty of performing inference (e.g., tracking and state estimation) on the continuous and discrete aspects, given that each aspect requires different underlying mathematics, algorithms, and often inference tools. Faults might occur within the continuous aspects (e.g., valve that is stuck partially shut) or the discrete aspect (on/off actuator fails in the *on* state) [28]; these faults may evidence themselves as gradual or abrupt faults. Some approaches (e.g., [4]) have aimed to combine the two aspects, while other approaches map the aspects into a single framework, e.g., a probabilistic framework for which we can use particle filters or dynamic Bayesian networks to compute diagnoses [17].

This article develops a computationally efficient diagnostics approach for solving a class of hybrid systems that can be modeled using a Timed Event Graph (TEG) [10], which is used for representing a broad range of timed systems such as process control and manufacturing systems, transportation networks, and communications systems. This restricted class of discrete-event system (DES) allows synchronization without concurrency or selection. We model this class of DES with a max-plus linear discrete-event system, in which inference is linear within the max-plus algebra [2, 7], and hence is computationally more efficient than traditional approaches [27].¹

¹For example, in the max-plus algebra exponentiation reduces to conventional multiplication.

G. Provan (✉)
School of Computer Science, University College Cork, Cork, Ireland
e-mail: g.provan@cs.ucc.ie

Researchers have developed techniques for TEG control by integrating TEG and a max-plus algebra. This framework has been successfully applied to solve control problems that include scheduling, hybrid systems (switching) operations, and just-in-time control [6, 25]. Given the wide use for modelling and control, these methods have received limited attention for HS diagnostics inference.

This article proposes a max-plus algebraic framework for modeling and diagnostic inference of a hybrid system (HS). We describe a discrete-time hybrid system based on a $(\max, +)$ -linear (MPL) algebra defined over a set \mathbb{Z}^+ or $\mathbb{R}_{\max} = \mathbb{R} \cup \{-\infty\}$ [11, 12]. A max-plus algebra is defined over the max-plus semi-ring $(\mathbb{R}_{\max}, \oplus, \otimes)$, which is the set $\mathbb{R}_{\max} = \mathbb{R} \cup \{-\infty\}$ together with operations $x \oplus y = \max(x, y)$ and $x \otimes y = x + y$. The additive and multiplicative identities are taken to be $\epsilon = -\infty$ and $e = 0$, respectively.

We adopt the max-plus algebra, one of many idempotent semi-rings used for computational inference, not only because its operations are associative, commutative and distributive (as in conventional algebra), but also because it transforms inference on system timed dynamics (that are non-linear in a conventional algebra) to be linear in the max-plus algebra [2].

We extend this model to a switching MPL (SMPL) framework [25], which introduces modes that the system switches between. We further extend SMPL systems with stochastic switching behaviours to capture the stochastic nature of faults occurring. We introduce stochastic fault occurrence through a probability distribution over mode transitions. The stochastic SMPL framework provides a rich theoretical basis for describing a set of real-world systems, e.g., piece-wise-affine (PWA) systems in the time-driven domain [26].

We employ a computationally efficient observer-based diagnostics approach to monitor the system and isolate faults. Our diagnostics approach uses the max-plus model for efficient inference, and also restricts the space of diagnoses considered during fault isolation by computing only the most-likely system behaviours (rather than using the space of all possible behaviours).

Our contributions are as follows:

- We model a HS using a switching $(\max, +)$ -linear (SMPL) algebra.
- We define a classical observer-based monitoring framework, and extend this for fault isolation.
- We discuss the computational complexity aspects of inference.

4.2 Problem Statement

This section summarizes our objective and methodology.

4.2.1 Hybrid Systems Model

This section describes our formulation of the hybrid diagnosis problem. We first define a hybrid system.

Definition 1 (Hybrid System) A hybrid system is a 5-tuple $\langle x, \Gamma, \mathcal{F}, (u, \phi) \rangle$, where

- $x \subseteq \mathbb{R}^n$ is the set of continuous state variables, where $x = \{x_1, \dots, x_n\}$.
- $\Gamma = \{\gamma_1, \dots, \gamma_k\}$ is a finite set of system modes.
- $\mathcal{F} = \{f_{\gamma_1}, \dots, f_{\gamma_k}\}$ is a finite set of functions and associated parameter values θ such that for each mode, $\gamma_i, f_{\gamma_i}(t, \theta, x(t)) : \mathbb{R} \times \mathbb{R} \times x \rightarrow x$ defines the continuous behaviour of the system in mode γ_i .
- (u, ϕ) describes the discrete switching behaviour of the system, with $u = \{u_1, \dots, u_m\}$ a finite set of controls that transition the system between modes, and ϕ a switching (transition) function that maps an action, mode and system state vector into a new mode and state vector, i.e., $\phi : u \times \Gamma \times x \rightarrow \Gamma \times x$.

A hybrid system creates a *trajectory*, which is a timed sequence of observations, $\mathbf{Y}_{0,k} = (\mathbf{y}(0), \dots, \mathbf{y}(k))$. We assume that the corresponding state variable trajectory $\mathbf{X}_{1,k}$ is unobservable apart from the initial conditions x_0 . We further assume that control inputs are observable events, but that fault transitions are unobservable events.

Sections 4.4.2 and 4.4.3 describe the continuous and discrete dynamics for \mathcal{F} and (u, ϕ) in greater detail, respectively.

4.2.2 Objective

Consider a discrete-time affine system whose dynamics obeys one of μ possible models (known and observable), with each model corresponding to a system mode. Our objective, given an anomalous output $\hat{\mathbf{y}}(k)$, is to determine the system mode $\gamma(k)$ at time k that most closely generates the observed dynamics. To achieve this goal, we look for the (shortest) sequence of inputs $\mathbf{U}_{0,k} = (\mathbf{u}(0), \dots, \mathbf{u}(k))$ and measurements $\mathbf{Y}_{0,k} = (\mathbf{y}(0), \dots, \mathbf{y}(k))$ such that the output at time k is consistent with only one mode $\gamma(k) \in \Gamma$. Since multiple input sequences of minimal length l may satisfy this requirement, we select the mode that minimizes a given cost function. In the following, we assume that only one model is active during each discrete step in $[0, \dots, N]$.

4.2.3 System Architecture

We address our task using the 3-step process shown in Fig. 4.1.

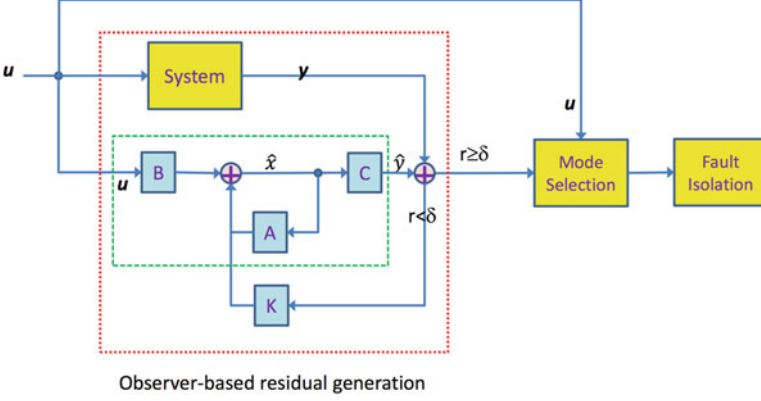


Fig. 4.1 Architecture of approach. The three steps are monitoring and residual generation, mode selection and fault isolation

1. *Observer-based monitoring and residual generation:* In this phase we use an observer-based approach to detect anomalies, using a residual r
2. *Mode selection:* Given an anomaly, this phase computes the set $\Gamma^* \subseteq \Gamma$ of most-likely modes, and runs a simulation (given input $\mathbf{x}(0), U_{0,k}$) for each mode $\gamma_i \in \Gamma^*$ to estimate the corresponding residual value r_i .
3. *Fault isolation:* In this phase we compute the most likely failure mode for the system based on a cost function $\mathcal{J}(\mathbf{y}, \hat{\mathbf{y}})$ and the set of r_i .

In this figure, we adopt the state-space model (as presented in Definition 5), where A, B are the state matrices, C the output matrix, and K the observation matrix.

4.3 Related Work

4.3.1 Algebraic Descriptions of Hybrid Systems

The class of switching MPL systems is related to $(\max, +)$ automata [12], which can also be characterized as non-stationary autonomous max-plus-linear systems with finitely valued dynamics (i.e. systems of the form $\mathbf{x}(k+1) = A(k) \otimes \mathbf{x}(k)$, $\mathbf{y}(k) = C \otimes \mathbf{x}(k)$ where $A(k)$ takes its values in a finite set $\{A(1), \dots, A(N)\}$. The main differences are that the class of systems considered here have an additional input ($\mathbf{u}(k)$), and that we define the switching mechanism completely and explicitly for the switching max-plus-linear systems (which is not true for $(\max, +)$ automata).

We represent max-min-plus-scaling (MMPS) systems in state-space form using the operations maximization, minimization, addition and scalar multiplication. MMPS systems are equivalent to a particular class of hybrid systems, continuous piecewise (PWA) systems [23]. PWA systems are defined by partitioning the state

space of the system in a finite number of polyhedral regions and associating to each region a different affine dynamic [18]. This relation between PWA and MMPS systems enables the study of certain structural properties of PWA systems, such as observability and controllability but also in designing controller schemes like model predictive control (MPC) [5].

Our work differs from other modeling techniques for discrete-event systems, such as Petri nets, extended state machines, event-graphs, formal languages, generalized semi Markov processes, non-linear programming, automata, computer simulation models (see, e.g., [15, 22]), in that we employ an algebraic approach with observers.

4.3.2 Petri Net Models

Timed Petri nets include as a subclass timed event graphs (TEG), where we represent places as “arcs” and transitions as “nodes” [16]. In this case, all places have a single transition upstream (we remove competition in either consumption or supply of tokens in TEG) and a single one downstream (we resolve all potential conflicts in using tokens in places by some predefined policy). We gain computational advantage, although these limitations restrict some application domains; the limitations can generally be satisfied by making some design and scheduling decisions at an abstract (or hierarchical) level.

Diagnosis of Petri nets (e.g., [3]) has a rich history. What is different in our approach is the use of state-space descriptions together with observer-based monitoring, and computationally efficient methods based on the $(\max,+)$ algebra.

4.3.3 Diagnosing Hybrid Systems

Researchers have directed considerable attention to the monitoring and fault diagnosis of hybrid systems, e.g., [1, 28, 29]. Our approach is the first to employ a $(\max,+)$ algebra for this task. The $(\max,+)$ algebraic approach is computationally more efficient than existing approaches, but may suffer from requiring longer time delays for observations for fault isolation, and a limitation to a class of HS that can be diagnosed based on timing anomalies.

4.4 Behaviour Modeling: Switching Max-Plus Linear Systems

This section summarizes the models that we use for the continuous and discrete behaviours of a hybrid system. We describe in turn a max-plus algebra, max-plus linear (MPL) systems, and switching max-plus linear (SMPL) systems. MPL

systems can be generalized to capture a broad range of hybrid systems. By introducing modes we can capture switching behaviours [25]. We can introduce different forms of uncertainty in the model to capture different types of stochastic behaviours, e.g., see [24]. In this article we introduce uncertainty in mode switching, in order to capture uncertainty in the onset of fault modes.

4.4.1 Max-Plus Algebra

This section outlines the basis for our algebraic frameworks. We first define $\epsilon = -\infty$ and $\mathbb{R}_{\max} = \mathbb{R} \cup \{\epsilon\}$.

Definition 2 (Max-Plus-Algebra [2, 8]) A max-plus-algebra $(\mathbb{R}_{\max}, \oplus, \otimes)$, for numbers $x, y \in \mathbb{R}_{\max}$ defines addition (\oplus) and multiplication (\otimes) as follows:

$$x \oplus y = \max(x, y) \quad (4.1)$$

$$x \otimes y = x + y, \quad (4.2)$$

We extend these to matrix operations as follows:

$$[A \oplus B]_{ij} = a_{ij} \oplus b_{ij} = \max(a_{ij}, b_{ij}), \quad (4.3)$$

$$[A \otimes C]_{ij} = \bigoplus_{k=1}^n a_{ik} \otimes c_{kj} = \max_{k=1, \dots, n} (a_{ik} + c_{kj}) \quad (4.4)$$

for matrices $A, B \in \mathbb{R}_{\max}^{m \times n}$ and $C \in \mathbb{R}_{\max}^{n \times p}$.

4.4.2 Continuous Dynamics: Max-Plus Linear Systems

We define our continuous dynamics using a max-plus state space model (Eq. (4.5)). Here, we define dynamics for a single mode, and we extend this to multiple modes in Sect. 4.4.3.

Max-plus-linear (MPL) systems are a class of discrete-event system that allow synchronization but no concurrency or choice [2]. We define MPL systems using two operators, \max and $+$. The \max function models the synchronization between events: an event occurs once all processes it depends on have finished. The $+$ function models the process times: the moment a process finishes must equal the sum of starting time and the time the process takes to finish. MPL systems are called max-plus-linear systems since the underlying temporal algebra has computational complexity that is “linear” in the max-plus algebra [2].

Definition 3 (Max-Plus Linear System)

$$\mathbf{x}(k) = A(k) \otimes \mathbf{x}(k-1) \oplus B(k) \otimes \mathbf{u}(k), \quad (4.5)$$

with $A \in \mathbb{R}_{\max}^{n \times n}$, $B \in \mathbb{R}_{\max}^{n \times m}$ and $C \in \mathbb{R}_{\max}^{m \times n}$, with a number n of states and m of inputs.

The index k denotes the event counter. For MPL systems the state $\mathbf{x}(k)$ ² typically contains the time instants at which the internal events occur for the k th time, the input $\mathbf{u}(k)$ contains the time instants at which the input events occur for the k th time, the output $\mathbf{y}(k)$ contains the time instants at which the output events occur for the k th time.

4.4.3 Switching Max-Plus Linear Systems

We now extend our framework to cover systems that can switch between different modes of operation [25]. We assume that a system operates in some mode $\gamma \in \Gamma$, where $|\Gamma| = 2^\eta$ modes. We partition Γ into a subset Γ_f of η_f fault modes and Γ_n of η_n nominal modes.

Definition 4 (Switching Max-Plus-Linear (SMPL) System) A switching max-plus-linear state space model exists in mode $\gamma(k) \in \dots, n_m$ for event step k as governed by

$$\hat{\mathbf{x}}(k+1) = A^{(\gamma(k))} \otimes \hat{\mathbf{x}}(k) \oplus B^{(\gamma(k))} \otimes \mathbf{u}(k) \quad (4.6)$$

$$\hat{\mathbf{y}}(k) = C^{(\gamma(k))} \otimes \hat{\mathbf{x}}(k), \quad (4.7)$$

in which the matrices $A^{(\gamma(k))}$, $B^{(\gamma(k))}$, $C^{(\gamma(k))}$ are the system matrices for mode $\gamma(k)$.

Mode switching can take place due to continuous transitions and due to discrete transitions (control inputs using a switching function ϕ). We assume that fault transitions are unobservable and occur due to the continuous dynamics of the system. The continuous switching allows us to model mode changes over both nominal and fault modes. Such mode switches include changes in the structure of the system, such as breaking a synchronization or changing the order of events. Each mode γ corresponds to a set of required synchronizations and an event order schedule, which leads to a model with system matrices $(A^{(\gamma(k))}, B^{(\gamma(k))})$ for the γ th model. The mode $\gamma(k)$ determines which max-plus linear model is valid during the k th event.

²In this article, boldface variables denote vectors.

The moments of discrete switching are determined by a switching mechanism ϕ . We partition $\mathbb{R}_{\max}^{n_z}$ into η subsets $Z(i), i = 1, \dots, \eta$. The mode $\gamma(k)$ is now obtained by determining the set $\gamma(k)$ at event step k . So if $\gamma(k) \in Z(i)$, then $\gamma(k) = i$. The switching mechanism is application-dependent; in some systems, it will depend on the state $\mathbf{x}(k-1)$ and input $\mathbf{u}(k)$, while in other examples $\gamma(k)$ will be governed by $\mathbf{w}(k)$.

4.4.4 Stochastic SMPL Systems

In real-world scenarios, fault transitions are stochastic. We can capture that behaviour in the SMPL framework using the mode transition behaviours (switching mechanism) $\gamma(k)$. In our original definition, the functional form of $\gamma(k)$ was left open. We can define stochastic failure-mode transitions, together with deterministic nominal-mode transitions, using a Markov transition matrix [24].

In this article, we assume that faults occur randomly, inducing random mode switches from a nominal mode to a fault mode. Once a fault occurs, it is persistent. We capture this using a stochastic variable π_{ij} , which defines the probability of switching from mode $\gamma_i(k-1)$ at time $k-1$ to mode γ_j at time k :

$$\pi_{ij} = P[\gamma_j(k) | \gamma_i(k-1)].$$

For example, we may have a stochastic switch from a nominal mode $\gamma_i(k-1)$ to a failure mode $\gamma_j(k)$, where the switching probability is 0.01.

We can define a switching probability matrix for the stochastic variable π_{ij} over η modes, with entries given by $\pi_{ij}, i, j = 1, \dots, \eta$ as:

$$P_S = \begin{bmatrix} \pi_{11} & \cdots & \pi_{\eta 1} \\ \vdots & \ddots & \vdots \\ \pi_{1\eta} & \cdots & \pi_{\eta\eta} \end{bmatrix} \quad (4.8)$$

4.4.5 Generality of Approach

Our algebraic approach is general and extensible, in that we can maintain the problem structure and obtain a different problem by changing the semi-ring. For example, we can maintain the problem structure of Definition 4, and simply by changing to the semi-ring $\langle [0, 1], (+, \times) \rangle$, we obtain an HS defined by a dynamic Bayesian network [19]. Furthermore, we can still use the inference architecture of Fig. 4.1 to solve this dynamic Bayesian network.

We make this change by defining x , y and u as stochastic variables, matrices A , B , C as Markov transition matrices. With this modification of the model representation, we can apply the semi-ring operations over the semi-ring $\langle [0, 1], (+, \times) \rangle$.

In an analogous fashion, we can substitute several different semi-rings into Definition 4 to obtain different HS formulations, with no change in inference tools other than the semi-ring operations. For example, [21] defines several types of diagnostics inference that are possible by adopting different semi-ring operations.

4.5 Running Example

We illustrate our concepts using a three-tank system, as shown in Fig. 4.2.

4.5.1 Nominal Model

We denote the tanks as T_1 , T_2 and T_3 . They all have the same area $A_1 = A_2 = A_3 = 3 \text{ [m}^2\text{]}$. We assume that $g = 10$ and the liquid is “pure” water with density $\rho = 1$.

Tank T_1 is filled from a pipe q_0 with a constant flow of $0.75 \text{ [m}^3\text{/s]}$. It drains into T_2 via a pipe q_1 . The liquid level is denoted as h_1 . There is a pressure sensor p_1 connected to T_1 that measures the pressure in Pascals [Pa]. The system has valves V_1, V_2, V_3 as shown in Fig. 4.2.

We can measure the tank pressure values, i.e., the measurement vector is $y = \{p_1, p_2, p_3\}$. Our control task is to maintain set-point heights in each of the tanks. The diagnostic task is to compute faults in tanks T_1 (leaks) and valves V_i , given p_i , for $i = 1, 2, 3$.

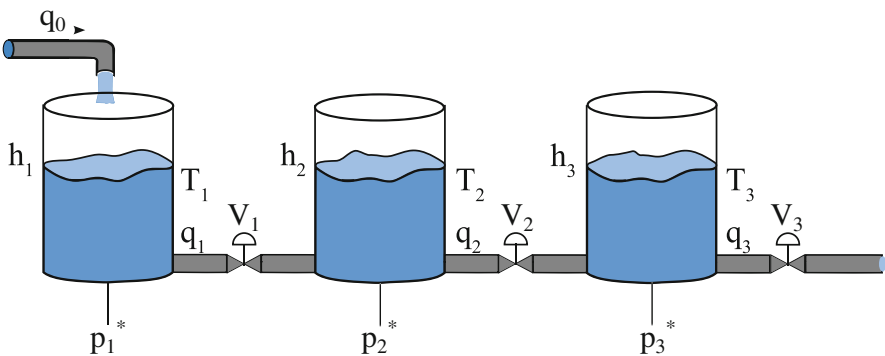


Fig. 4.2 Diagram of the three-tank system

We define our nominal model as follows. According to Torricelli's Law, flow q_i out of tank i , with liquid level h_i , into tank j , is given by:

$$q_i = \zeta \operatorname{sign}(h_i - h_j) \sqrt{2g(h_i - h_j)}, \quad (4.9)$$

where the coefficient ζ is used to model the area of the drainage hole and its friction factor through the hole.

We can use Eq. (4.9) to derive the following equations:

$$\begin{aligned} \dot{h}_1 &= q_0 - c_1 \sqrt{h_1 - h_2} \\ \dot{h}_2 &= c_1 \sqrt{h_1 - h_2} - c_2 \sqrt{h_2 - h_3}, \\ \dot{h}_3 &= c_2 \sqrt{h_2 - h_3} - c_3 \sqrt{h_3}, \end{aligned} \quad (4.10)$$

where the constants c_1, c_2, c_3 summarize the system parameters representing cross-sectional areas, friction factors, gravity, etc.

Finally, we can compute from the water level a pressure given by

$$p_i = \frac{g h_i A}{A} = g h_i \quad (4.11)$$

where i is the tank index ($i \in \{1, 2, 3\}$).

Our control objective is to achieve set-point heights h_1^+ and h_3^- in tanks 1 and 3, respectively: we want to maintain the level of h_1 below its maximum, i.e., $h_1 \leq h_1^+$, and the level of h_3 above its minimum, i.e., $h_3 \geq h_3^-$. We assume that we have a constant inflow q_0 and we modify only the setting of valve V_3 . We denote $V_3 = 0$ to be closed, and $V_3 = 1$ to be open, i.e., the valve can only be fully closed or fully open. Our switching control engages when either $h_3 < h_3^-$ or $h_1 > h_1^+$.

Figure 4.3 shows a typical simulation for this system. A cycle of the system (covering times $t = 0$ to $t = 5.8$ s) is as follows. We start with the valve set open, and we see that the fluid levels h_1 and h_3 are both falling. When the level $h_3 < h_3^-$, we close the valve. The levels h_1 and h_3 now rise, and we open the valve when $h_1 > h_1^+$. The cycle then repeats.

4.5.2 Fault Model

In the following we define valve (actuator) faults; other faults, e.g., leaks or sensor faults, can be defined analogously.

We assume an additive valve fault, where the actual valve position for valve i , given commanded position v_i and fault Δ_{v_i} , is

$$v_i = \begin{cases} \max\{0, v_i + \Delta_{v_i}\} & \text{if } \Delta_{v_i} \leq 0 \\ \min\{1, v_i + \Delta_{v_i}\} & \text{if } \Delta_{v_i} > 0 \end{cases} \quad (4.12)$$

where $\Delta_{v_i} \in [-1, 1]$.

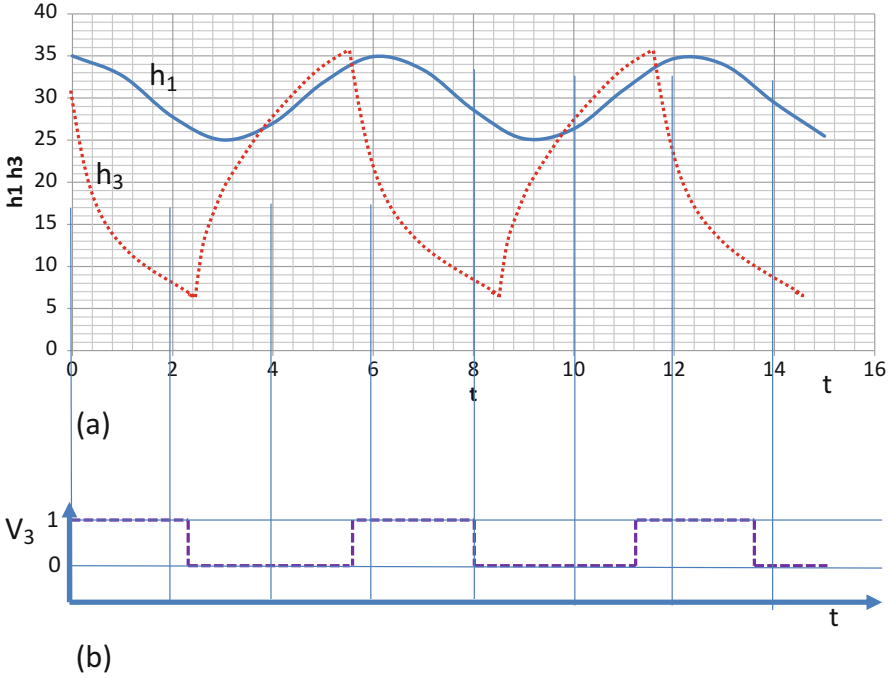


Fig. 4.3 Simulation of the three-tank system. (a) Height of liquid in tanks h_1 and h_3 . (b) Setting of valve V_3

4.5.3 Max-Plus Model

We now derive a max-plus model from the ODE model. Our aim is to create an event-based representation with timings between event. We use as events the switches of valve V_3 . We set z_2 as the time when a switch σ_2 occurs due to the height in tank 1 being exceeded ($h_1 > h_1^+$), and z_1 as the time when a switch σ_1 occurs due to the height in tank 3 going too low ($h_3 < h_3^-$). We set t_0 as our initial time. We identify two time parameters, τ_1 and τ_2 for the nominal operation of the system. τ_1 is the time interval between switch $\sigma_1(k - 1)$ and $\sigma_1(k)$. Analogously, τ_2 is the time interval between switch $\sigma_2(k - 1)$ and $\sigma_2(k)$.

As a consequence, we must have

$$z_1(k) \geq \max\{z_2(k - 1) + \tau_1\}$$

$$z_2(k) \geq \max\{z_1(k - 1) + \tau_2\}.$$

In the max-plus algebra we write this as

$$z_1(k) \geq \tau_1 \otimes z_2(k - 1)$$

$$z_2(k) \geq \tau_2 \otimes z_1(k - 1),$$

which in matrix notation is $Z(k) \geq A_0 Z(k) \oplus A_1 Z(k-1)$, or

$$\begin{bmatrix} z_1(k) \\ z_2(k) \end{bmatrix} \geq \begin{bmatrix} \epsilon & \epsilon \\ \tau_2 & \epsilon \end{bmatrix} \begin{bmatrix} z_1(k) \\ z_2(k) \end{bmatrix} \oplus \begin{bmatrix} \epsilon & \tau_1 \\ \epsilon & \epsilon \end{bmatrix} \begin{bmatrix} z_1(k-1) \\ z_2(k-1) \end{bmatrix}, \quad (4.13)$$

such that $Z(k) = [z_1(k) \ z_2(k)]^T$, $A_0 = \begin{bmatrix} \epsilon & \epsilon \\ \tau_2 & \epsilon \end{bmatrix}$ and $A_1 = \begin{bmatrix} \epsilon & \tau_1 \\ \epsilon & \epsilon \end{bmatrix}$. By writing $A = A_0^* \otimes A_1$, where we apply the Kleene matrix product to obtain A_0^* , we obtain $Z(k) = A \otimes Z(k-1)$, with $A = \begin{bmatrix} \epsilon & \tau_1 \\ \epsilon & \tau_2 \end{bmatrix}$.

4.6 Diagnosing Hybrid Systems Using SMPL Automata

This section introduces an observer framework for monitoring SMPL Automata; we then extend this to isolate faults in these automata.

4.6.1 Observers

We now extend a max-plus-linear state space model that is in mode $\gamma(k) \in 1, \dots, \eta$ for event step k into a monitored system using a residual vector $\mathbf{r}(k)$. We define our state specification with observer as follows:

Definition 5 (Observer State Space Model)

$$\begin{aligned} \hat{\mathbf{x}}(k+1) &= A^{(\gamma(k))} \otimes \hat{\mathbf{x}}(k) \oplus B^{(\gamma(k))} \otimes \mathbf{u}(k) \\ &\quad \oplus K^{(\gamma(k))} \otimes \mathbf{r}(k) \end{aligned} \quad (4.14)$$

$$\hat{\mathbf{y}}(k) = C^{(\gamma(k))} \otimes \hat{\mathbf{x}}(k) \quad (4.15)$$

$$\mathbf{r}(k) = |\hat{\mathbf{y}}(k) - \mathbf{y}(k)|, \quad (4.16)$$

where variables with a hat ($\hat{\mathbf{x}}, \hat{\mathbf{y}}$) correspond to model predictions, and variable \mathbf{y} (without a hat) corresponds to the measured output.

In this description, $K^{(\gamma(k))}$ is the observer gain matrix that we must tune. Further details of observer-based control can be found in [13].

Given an observer, we can monitor our system and identify anomalous behaviour using the residual as follows:

Definition 6 (Fault Detection) Given a non-negative threshold $\delta \in \mathbb{R}$, an anomaly (corresponding to a fault) exists if $|\mathbf{r}(k)| > \delta$.

4.6.2 Isolating Faults

This section describes a method for fault isolation given an anomaly. In general, we can isolate faults in this framework using a range of approaches, e.g., a bank of residual generators, ARR, etc. A classical approach for fault isolation is to use a bank of residual generators, one for each fault to be diagnosed [27]. This approach can be easily accommodated in our framework; however, it does not scale well to the large number of possible multiple-fault combinations.³ As a consequence, it is computationally prohibitive to search the entire space, and using a bank of residual generators typically is limited to single-fault scenarios.

In the following we address the most-likely multiple-fault scenarios. In particular, we use the fault-transition probabilities to focus inference on the most-likely fault trajectories.

We need to introduce a few definitions to clarify our fault isolation procedure. We are interested in multiple-fault diagnoses, where we allow each failure mode $\gamma \in \Gamma_f$ to take on a discrete set of values.

Definition 7 (Trajectory) A trajectory is a sequence of events and states.

Definition 8 (Observation Sequence) An observation sequence is a sequence of observable events.

For a stochastic transition, we compute the probability of a future state.

Definition 9 (Stochastic State Estimation Task) Given a sequence Y of k observable events and initial state $x(0)$, the stochastic state estimation task is to identify $P(x(k))$, the probability of state x at time k .

We can use the switching probability matrix P_S to compute $P(\gamma(k)|P(\gamma(0)))$, where $\gamma(0)$ is the initial mode. P_S^k denotes the probability distribution over arriving at any mode after k steps. More precisely, the ij th entry denotes the probability of moving from mode i to mode j after k steps. We can use this matrix to compute the probability $P(\gamma(k)|P(\gamma(0)))$ for any mode $\gamma \in \Gamma$.

We adopt as our baseline diagnostic approach the use of multiple observers, where we compute with each observer a residual tuned to a particular fault. We assume that we compute just the single-fault diagnoses with each residual. We will then compare this approach with a multiple-fault approach.

In this article we adopt an approximation-based approach for multiple-fault isolation, where we investigate the most-likely sub-space of the diagnosis space. To do this, we must compute the most-likely trajectories. Fortunately, these are easily computed using algebraic techniques. Assume that we identify an anomaly at k steps. Using a $(\max, *)$ -algebra where $*$ is standard multiplication, we can compute the probability of a fault occurring at k steps using P_S^k using the $(\max, *)$ -algebra, which computes the probability of paths of length k [20]. The entry π_{ij} in

³The diagnosis space is exponentially-growing in Γ_f , the number of fault modes.

P_S^k denotes the maximum probability of the k -step path from mode i to mode j . We use a threshold δ_π such that we consider fault occurrence for fault j only if $\pi_{ij} \geq \delta_\pi$. We consider the set Γ_f of faults.

We identify the fault that minimizes the loss function $\mathcal{J}(\hat{\mathbf{y}}, \mathbf{y})$:

$$\gamma_f^* = \arg \min_{\gamma_f \in \Gamma_f} \mathcal{J}(\hat{\mathbf{y}}, \mathbf{y}) \quad (4.17)$$

In this article, we use a probabilistic loss function, so we compute the highest-probability fault.

4.7 Computational Complexity

The complexity of diagnostic inference in a switching max-plus system (Definition 4) depends on two factors:

Fault Detection To identify an anomaly, we must solve our system to compute \mathbf{r} , which requires solving a matrix relation of the form given by Eqs. (4.6) and (4.7).

Fault Isolation This phase of inference requires us to identify the failure mode that “explains” the anomaly, i.e., that minimizes our diagnostics cost function.

We now define the complexity of each factor in turn.

4.7.1 Fault Detection

Given an observation $\mathbf{y}(k)$ at time k , computing a residual $\mathbf{r}(k) = |\mathbf{y}(k) - \hat{\mathbf{y}}(k)|$ involves estimating the output using $\hat{\mathbf{y}}(k)$, which we can calculate from Eqs. (4.6) and (4.7) as

$$\begin{aligned} \hat{\mathbf{y}}(k) &= C \otimes \hat{\mathbf{x}}(k) \quad \text{for } k = 1, 2, \dots \\ &= C \otimes \left[A^{\otimes k} \otimes \mathbf{x}(0) \oplus \bigoplus_{i=1}^k A^{\otimes k-i} \otimes B \otimes \mathbf{u}(i) \right] \end{aligned}$$

Computing the k th power of a matrix, for $k \in \mathbb{N}_0$, takes the form

$$(A^{\otimes k})_{ij} = \max_{i_1, i_2, \dots, i_{k-1}} (a_{ii_1} + a_{i_1 i_2} + \dots + a_{i_{k-1} j}) \quad \forall i, j.$$

This is clearly linear in the size of the matrix. From this, we can see that computing the residual is linear in the size of the matrices involved. This contrasts with traditional matrix operations, which are $O(n^3)$ for $n \times n$ matrices.

4.7.2 Fault Isolation

Isolating faults is the computationally taxing part of the problem. Below, we outline the worst-case complexity of this problem, and then show the approximation technique that we adopt.

We can define our diagnostic problem as follows:

Definition 10 (SMPL Diagnosis) Given an SMPL system with initial condition $\mathbf{x}(0)$ and anomalous observation \mathbf{y} , compute a switching sequence ending with a persistent fault that generates an output $\hat{\mathbf{y}}$ such that $\mathcal{J}(\hat{\mathbf{y}}, \mathbf{y})$ is minimized over all permutations of switching sequences.

We can use this problem formulation to prove a decision version of our diagnostics task.

Proposition 1 (SMPL Diagnosis Complexity) Given an integer SMPL system with initial condition $\mathbf{x}(0)$ and anomalous observation $\mathbf{y}(k)$ at time k , it is NP-complete to compute if there exists a switching sequence ending with a persistent fault that generates an output $\hat{\mathbf{y}}(k) = \mathbf{y}(k)$ at time k .

The full diagnosis problem (Definition 10) is an optimization version of the decision problem, so is NP-hard. The problem is the exponential number of switching sequences that must be analysed. It is important to note that, although the SMPL approach solves an NP-hard diagnostics inference problem, this is a more tractable task than using conventional HS diagnostics inference.

4.7.3 Approximation Algorithm

We use an approximation technique to explore a polynomial number of switching sequences (trajectories), rather than the (worst-case) exponential number of switching sequences. We use the stochastic function governing fault transitions to assign probabilities to the trajectories, and explore only the trajectories whose probability is higher than a threshold φ . By controlling the value of φ we can limit the number of trajectories to be polynomial in $|\mathbf{x}|$. This gives us a principal way to trade off inference speed with fault isolation accuracy.

Alternatively, we could solve this problem as a mixed-integer linear programming problem [9], for which a number of efficient solvers exist.

4.8 Diagnosis Scenarios

This section covers the diagnosis scenarios for our tank example. We will examine three scenarios: (1) T_3 leak; (2) V_3 blockage; (3) T_2 leak.

We compute residuals for inter-event timings: $R = z_2(k) - z_2(k - 1)$ and $R_3 = z_1(k) - z_2(k - 1) = \tau_1$.

4.8.1 Scenario 1: T_3 Leak

Our first scenario covers a large leak in T_3 at $t = 4$. This causes the minimum set-point h_3^- to be achieved must faster than normal, which in turn causes a faster switch of V_3 at $t = 6.5$, as shown in Fig. 4.4. Our approach is able to isolate this fault quickly, with the leak in T_3 having the highest probability among fault candidates.

4.8.2 Scenario 2: V_3 Blockage

Our second scenario covers a temporary blockage in V_3 at $t = 4$. This causes the minimum set-point h_3^- to be achieved slower than normal, which in turn causes a slower switch of V_3 at $t = 8.7$ rather than $t = 8$, as shown in Fig. 4.5. Our approach is able to isolate blockage in V_3 as the highest-probability fault.

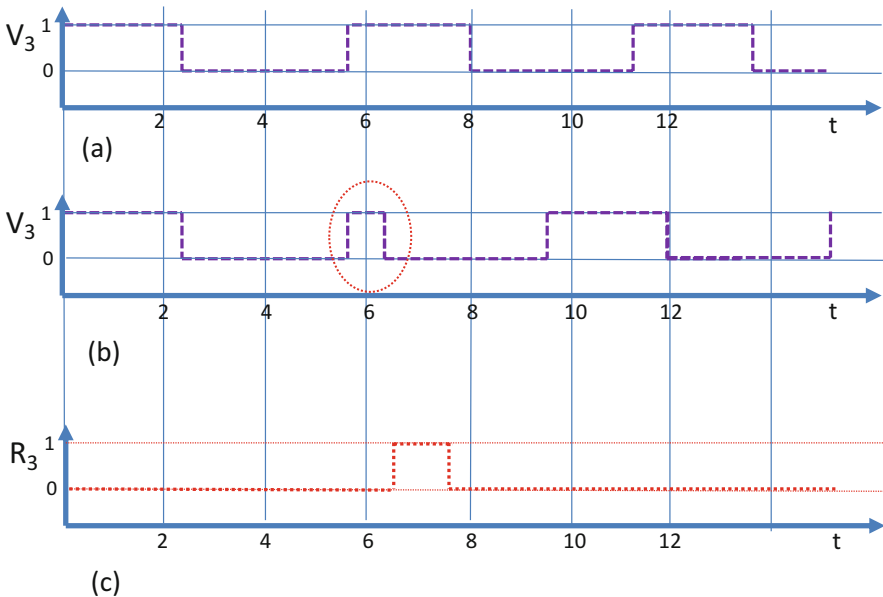


Fig. 4.4 Diagnosis scenario with leak in tank 3. (a) Normal setting of valve V_3 . (b) Faulty setting of valve V_3 . (c) Residual setting valve V_3

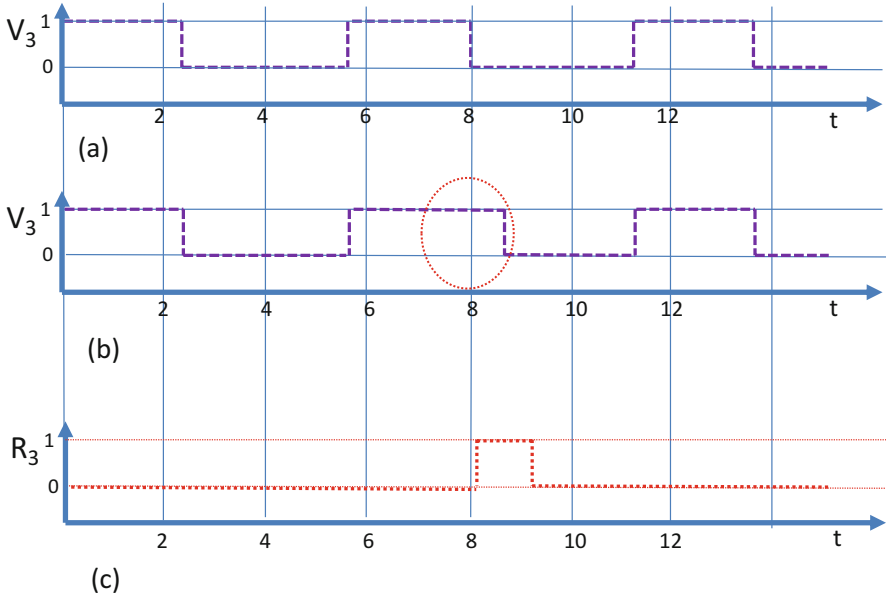


Fig. 4.5 Diagnosis scenario with blockage in valve V_3 . (a) Normal setting of valve V_3 . (b) Faulty setting of valve V_3 . (c) Residual setting valve V_3

4.8.3 Scenario 3: V_2 Blockage

Our third scenario covers a temporary leak in T_2 at $t = 4$. This causes the minimum set-point h_3^- and the maximum set-point h_1^+ to be achieved slower than normal. Both residuals R and R_3 signal a problem, but the system cannot isolate the unique cause of the problem. It computes the probabilities of a blockage in valves V_2 and V_3 as equally likely.

4.9 Types of Hybrid Systems Covered

This section discusses the types of hybrid system to which our SMPL framework is applicable. Our approach uses observers to minimize the error between the actual and predicted output times, possibly subject to additional constraints on the inputs and the outputs. In its most general sense, this approach is applicable to hybrid systems that consist of piecewise affine (PWA) systems [23]. PWA systems are well-known HS models since they capture a wide range of HS properties yet have mathematically tractable descriptions. PWA systems are mathematically tractable since they extend linear systems capable of modeling non-linear/non-smooth phenomena with an arbitrary degree of precision. Moreover, PWA systems

are expressive, in that they can represent key hybrid features such as linear-threshold events and mode switching.

van den Boom and De Schutter [26] has shown the following equivalence result:

Proposition 2 (Equivalence) *Every finite SMPL system can be written as a piecewise affine system.*

In the following, we briefly review the notation necessary to establish this result, and refer the reader to [26] for the full details.

van den Boom and De Schutter [26] use the results of [14] to show that an SMPL system can be rewritten as a piecewise affine system, a max-min-plus-scaling system, an (extended) linear complementarity (ELC/LC) system or as a mixed logic dynamical (MLD) system, all of which are used in the field of hybrid systems. We can use this equivalency to transfer properties of piecewise affine systems and max-min-plus-scaling systems to SMPL systems.

We first introduce the class of finite SMPL and piecewise affine systems that we consider. If we recall the definition of switching systems (SMPL), we have:

Definition 11 (Finite SMPL System) A finite SMPL system generates finite $\mathbf{x}(k)$, $\mathbf{y}(k)$ given as input finite $\mathbf{x}(k-1)$, $\mathbf{u}(k)$, for all $\gamma(k-1) \in \{1, \dots, \mu\}$.

Our aim is to relate a finite SMPL system to a PWA system, which is based on a notion of polyhedral partitioning:

Definition 12 (Polyhedral Partition) A polyhedral partition $\{\Lambda_i\}_{i=1, \dots, n_s}$ of the space \mathbb{R}_w^n is defined as the partitioning of the space \mathbb{R}_w^n into non-overlapping polyhedra Λ_i , $i = 1, \dots, n_s$ of the form

$$\Lambda_i = \{w(k) | S_i w(k) \preceq_i s_i\}, \text{ for } i = 1, \dots, n_s,$$

for some matrices $S_i \in \mathbb{R}^{q \times n_w}$ and vectors $s_i \in \mathbb{R}^q$, and with \preceq_i a vector operator where the entries stand for either \leq or $<$ and there holds

$$\bigcup_{i=1}^{n_s} \Lambda_i = \mathbb{R}^{n_w} \text{ and } \Lambda_i \cap \Lambda_j = \emptyset \text{ for } i \neq j.$$

We now define a PWA system as follows:

Definition 13 (Piecewise Affine System) Piecewise affine systems are described by

$$\mathbf{x}(k) = A_i \mathbf{x}(k-1) + B_i \mathbf{u}(k) + f_i \quad (4.18)$$

$$\mathbf{y}(k) = C_i \mathbf{x}(k) + D_i \mathbf{u}(k) + g_i \quad (4.19)$$

for

$$\begin{bmatrix} \mathbf{x}(k-1) \\ \mathbf{u}(k) \\ \mathbf{d}(k) \end{bmatrix} \in \Omega_i,$$

where $f_i \in \mathbb{R}^{n_x \times 1}$, $g_i \in \mathbb{R}^{n_y \times 1}$, $A_i \in \mathbb{R}^{n_x \times n_x}$, $B_i \in \mathbb{R}^{n_x \times n_u}$, $C_i \in \mathbb{R}^{n_y \times n_x}$, $D_i \in \mathbb{R}^{n_y \times n_u}$, for $i = 1, \dots, N$ where the signal $d(k) \in [0, 1]$ is a uniformly distributed stochastic scalar signal and $\Omega_{i=1, \dots, N}$ is a polyhedral partition of $\mathbb{R}^{n_x + n_u + 1}$.

To show this equivalence, [26] rewrite the original Definition 4 of SMPL systems in terms of piecewise affine polyhedral partitions:

Definition 14 (Stochastic SMPL System) Consider the system of [Definition 4] with μ possible modes and let the probability of switching to mode $\gamma(k)$ given $\gamma(k-1), \mathbf{x}(k-1), \mathbf{u}(k), \mathbf{w}(k)$ be denoted by $P[\pi(k) = \gamma(k) | \gamma(k-1), \mathbf{x}(k-1), \mathbf{u}(k), \mathbf{w}(k)]$. This qualifies as a Switching Max-Plus-Linear (SMPL) system if for any given $\gamma(k) \in 1, \dots, \mu$, $P[\pi(k) = \gamma(k) | \cdot, \cdot, \cdot, \cdot]$ is a probability function that is piecewise affine on a polyhedral partition of the space of the variables $\gamma(k-1), \mathbf{x}(k-1), \mathbf{u}(k), \mathbf{w}(k)$.

These results indicate the broad range of hybrid systems that can be modelled using the max-plus framework. The advantage of a max-plus framework is generalizability as well as efficiency with respect to other representations.

4.10 Summary

This article has proposed a max-plus algebraic approach for solving a class of PWA hybrid systems. For this class of system the max-plus algebraic approach is computationally faster than traditional methods. We have described an approximation technique that is of complexity polynomial in the problem size, even though the general diagnostic inference task is NP-hard. We have illustrated our approach on a process-control example.

This approach provides a novel computational framework for diagnosing hybrid systems. We build on a significant base of work on modeling and controlling systems using the max-plus algebra. There are many avenues for future work, including applying this approach to large systems to test scaling properties, studying the impact of switching probabilities on fault isolation accuracy, and comparing our approach to state-of-the-art methods.

We also plan to show the generalizability of this algebraic approach, namely that just by changing the underlying algebraic operations we can define a range of stochastic hybrid systems, such as Markov switching systems or even non-linear systems whose dynamics typically are defined using particle filters [17]. We show how all the above approaches use the same formulation, and differ only in the underlying algebraic operations.

Acknowledgements The paper has been supported by SFI grants 12/RC/2289 and 13/RC/2094.

References

1. Arogeti, S. A., Wang, D., & Low, C. B. (2008). Mode tracking and FDI of hybrid systems. In *10th International Conference on Control, Automation, Robotics and Vision (ICARCV)* (pp. 892–897). New York: IEEE.
2. Baccelli, F., Cohen, G., Olsder, G. J., & Quadrat, J.-P. (1992). *Synchronization and linearity: An algebra for discrete event systems*. Chichester: Wiley.
3. Basile, F. (2014). Overview of fault diagnosis methods based on Petri net models. In *2014 European Control Conference (ECC)* (pp. 2636–2642). IEEE.
4. Bayouhd, M., & Travé-Massuyès, L. (2014). Diagnosability analysis of hybrid systems cast in a discrete-event framework. *Discrete Event Dynamic Systems*, *24*(3), 309–338.
5. Bemporad, A., & Morari, M. (1999). Robust model predictive control: A survey. In *Robustness in identification and control* (pp. 207–226). London: Springer.
6. Boukra, R., Lahaye, S., & Boimond, J.-L. (2015). New representations for (max,+) automata with applications to performance evaluation and control of discrete event systems. *Discrete Event Dynamic Systems*, *25*(1–2), 295–322.
7. Butkovic, P. (2010). *Max-linear systems: Theory and algorithms*. Berlin: Springer.
8. Cuninghame-Green, R. A. (1979). *Minimax algebra. Lecture notes in economics and mathematical systems* (Vol. 166). Berlin: Springer.
9. De Schutter, B., Heemels, W. P. M. H., & Bemporad, A. (2002). On the equivalence of linear complementarity problems. *Operations Research Letters*, *30*(4), 211–222.
10. Gaubert, S. (1990). An algebraic method for optimizing resources in timed event graphs. In *Analysis and optimization of systems* (pp. 957–966). Berlin: Springer.
11. Gaubert, S. (1995). Performance evaluation of (max,+) automata. *IEEE Transactions on Automatic Control*, *40*(12), 2014–2025.
12. Gaubert, S. (1997). Methods and applications of (max,+) linear algebra. In *Annual symposium on theoretical aspects of computer science* (pp. 261–282). Berlin: Springer.
13. Hardouin, L., Shang, Y., Maia, C. A., & Cottenceau, B. (2017). Observer-based controllers for max-plus linear systems. *IEEE Transactions on Automatic Control*, *62*(5), 2153–2165.
14. Heemels, W., De Schutter, B., & Bemporad, A. (2001). Equivalence of hybrid dynamical models. *Automatica*, *37*(7), 1085–1091.
15. Hillion, H. P., & Proth, J.-M. (1989). Performance evaluation of job-shop systems using timed event-graphs. *IEEE Transactions on Automatic Control*, *34*(1), 3–9.
16. Holloway, L. E., Krogh, B. H., & Giua, A. (1997). A survey of Petri net methods for controlled discrete event systems. *Discrete Event Dynamic Systems*, *7*(2), 151–190.
17. Koutsoukos, X., Kurien, J., & Zhao, F. (2002). Monitoring and diagnosis of hybrid systems using particle filtering methods. In *International Symposium on Mathematical Theory of Networks and Systems*.
18. Leenaerts, D., & Van Bokhoven, W. M. G. (2013). *Piecewise linear modeling and analysis*. New York: Springer Science & Business Media.
19. Lerner, U., Parr, R., Koller, D., Biswas, G. (2000). Bayesian fault detection and diagnosis in dynamic systems. In *AAAI/IAAI* (pp. 531–537).
20. Manger, R. (2008). A catalogue of useful composite semirings for solving path problems in graphs. In *Proceedings of the 11th International Conference on Operational Research (KOI 2006)*.
21. Provan, G. (2016). A general characterization of model-based diagnosis. In *Proceedings of the 22nd European Conference on Artificial Intelligence*. IOS Press.
22. Sahner, R. A., Trivedi, K., & Puliafito, A. (2012). *Performance and reliability analysis of computer systems: An example-based approach using the SHARPE software package*. Berlin: Springer Science & Business Media.
23. Sontag, E. (April 1981). Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, *26*(2), 346–358.

24. Van den Boom, T. J. J., & De Schutter, B. (2004). Model predictive control for perturbed max-plus-linear systems: A stochastic approach. *International Journal of Control*, 77(3), 302–309.
25. van den Boom, T. J. J., & De Schutter, B. (2006). Modelling and control of discrete event systems using switching max-plus-linear systems. *Control Engineering Practice*, 14(10), 1199–1211.
26. van den Boom, T. J. J., & De Schutter, B. (2012). Modeling and control of switching max-plus-linear systems with random and deterministic switching. *Discrete Event Dynamic Systems*, 22(3), 293–332.
27. Witczak, M. (2007) *Modelling and estimation strategies for fault diagnosis of non-linear systems: From analytical to soft computing approaches* (Vol. 354). Berlin: Springer Science & Business Media.
28. Zaytoon, J., & Lafortune, S. (2013). Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37(2), 308–320.
29. Zhao, F., Koutsoukos, X., Haussecker, H., Reich, J., & Cheung, P. (2005). Monitoring and fault diagnosis of hybrid systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(6), 1225–1240.

Chapter 5

Monitoring of Hybrid Dynamic Systems: Application to Chemical Process



Nelly Olivier-Maget and Gilles Hetreux

5.1 Introduction

In this chapter, a methodology of a fault detection and isolation for chemical process is presented. This methodology, called SimAEM (Simulation Abnormal Event Management) is particularly designed for the monitoring of batch and semi-continuous processes. These processes are the prevalent production mode for low volume of high added value products. Such processes are composed of interconnected and shared resources, in which a continuous treatment is carried out. For this reason, they are generally considered as hybrid systems where discrete aspects mix with continuous ones. Otherwise, the recipe is more often described with state events (temperature or composition threshold, etc.) than with fixed processing times [1]. SimAEM methodology is a model-based approach. Model-based diagnosis is widely discussed in the literature and many industrial applications exploit this principle [2]. Most of the methods in this approach are designed in three stages: residual generation, residual assessment and localization. In our study, our approach is based on a hybrid dynamic simulator. This simulator provides a reference model, which is supposed to be correct [3, 26]. The general architecture of SimAEM monitoring system is shown in Fig. 5.1.

The sequence of the different steps of a failure diagnosis is highlighted. Moreover, a distinction between the on-line and off-line steps is made. Our approach is therefore divided into three steps:

N. Olivier-Maget (✉) · G. Hetreux
Laboratoire de Génie Chimique, Université de Toulouse, CNRS, Toulouse, France
e-mail: nelly.olivier@ensiacet.fr

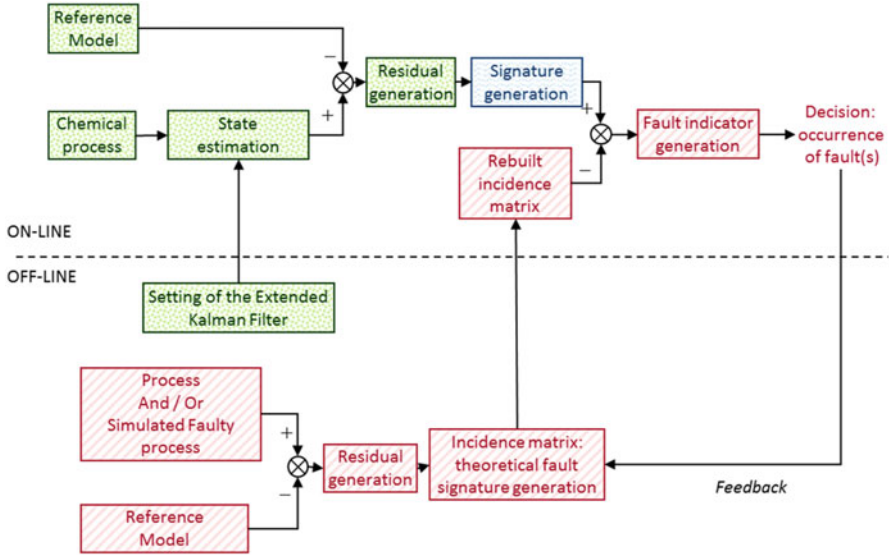


Fig. 5.1 SimAEM architecture

- The first step is the residual generation (in dotted motif in Fig. 5.1). It consists of the comparison between the predicted model obtained by simulation and the real system behaviour obtained by an observer. In our case, the extended Kalman filter is implemented. The aim is to reconstruct the system outputs from measurements.
- The second step (the signature generation) aims to analyse the residuals (in wave motif in Fig. 5.1). This is the detection step. It determines the presence or absence of a failure. The “signature” notion is introduced.
- Finally, the last step (in hatched motif in Fig. 5.1) consists of fault diagnosis. This step exploits the generated signatures in the previous step in order to determine the fault type. To do this, an inline matching process has been made. This is a pattern recognition problem. For this purpose, the instantaneous fault signature is then compared with the theoretical fault signatures by the calculation of distances in order to identify and localize the fault(s). These theoretical fault signatures are listed in the incidence table. The latter is obtained by experiment or by the off-line simulation of a faulty process.

5.2 Residual Generation by the Extended Kalman Filter

The initial step of a model-based diagnosis system generates fault indicators, called residuals. The residuals contain information on the drift or failure of the monitored system. The goal is to measure the difference between the system measurements

and the so-called “theoretical” value obtained by a reference model. The generation of residual is a critical step in the success of the diagnosis.

5.2.1 State Estimator: Extended Kalman Filter

Numerous works on Hybrid Dynamic Systems revolve around the axes of modelling, stability and controllability [4]. In recent years, more particular efforts have been made in the literature on observability. The high robustness and real-time ability of observer is well-known for industrial applications [5]. Although the theory of state observation has reached a certain degree of maturity in the domains of continuous and discrete events, the observation of dynamic hybrid systems remain a challenge.

The observation of state is particularly adapted to the studies of fault detection and diagnosis. It provides more information to make decision. Thus, the residual generation by a state estimation consists of rebuilding the state or, more generally, the process output by using observers, and then using the error estimation as residual. Clark was one of the first to use this concept [6]. If the problem of design of observers for linear systems seems well overcome, this is not the case for nonlinear systems: there is currently no satisfactory global solution.

In this study, extended Kalman filter has been chosen to rebuild the process state. Indeed, this filter is inexpensive in computation time and gives good results for moderate nonlinear systems [5, 7, 8, 26]. It should be noted that as soon as the nonlinearities become too strong or if it is badly initialized, extended Kalman filter is not efficient. In our work, this filter is based on the dynamic simulation of hybrid dynamic systems. PrODHyS simulator [9] provides models, which characterize the process behaviour, especially during the transient states. Thanks to the use of this filter, the monitoring is robust with noises and process uncertainties. It avoids thus false alarms.

The state reconstruction by the extended Kalman filter consists of making the estimation error independent of the uncertainties of the system. A description of this filter and of its implementation can be found in [9, 10].

5.2.2 Residual Generation

Next, residuals $r(t)$ are generated. They result from the comparison between the state vector reconstructed by the observer representing the estimated state $\widehat{X}(t)$, and the state vector $X(t)$ obtained with the reference model:

$$r(t) = \widehat{X}(t) - X(t) \quad (5.1)$$

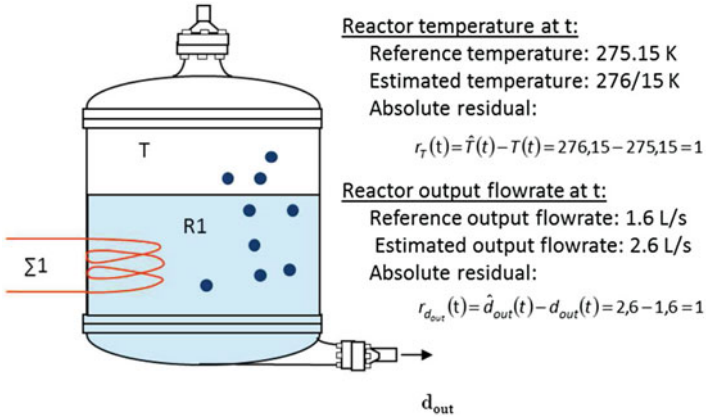


Fig. 5.2 Absolute residual

This residual is called “absolute” residual. Let us illustrate this concept through a simple example (Fig. 5.2). Consider a reactor $R1$ heated by an energy source $\Sigma 1$.

In order to determine the variables representative of the abnormal behaviour, it is necessary to compare the residual of the temperature T and the output flowrate d_{out} . However, although similar in value, these residuals are not dimensionless. To be able to compare them, they must be dimensionless: $r_T(t) = 1\text{K}$ and $r_{d_{out}}(t) = 1\text{L/s}$. For this, a relative residual is defined:

$$r^r(t) = \frac{\hat{X}(t) - X(t)}{X(t)} \quad (5.2)$$

Then, we obtain the following relative residuals.

$$r_T^r(t) = \frac{\hat{T}(t) - T(t)}{T(t)} = \frac{276.15 - 275.15}{275.15} = 0.36\%$$

$$r_{d_{out}}^r(t) = \frac{\hat{d}_{out}(t) - d_{out}(t)}{d_{out}(t)} = \frac{2.6 - 1.6}{1.6} = 62.5\%$$

It is possible to conclude the output flowrate is a variable representative of the abnormal state, while the temperature is under normal conditions.

5.3 Residual Estimation: Signature Generation

Real-time operation is an important factor in fault detection. Indeed, an early detection of a fault is an asset to avoid its consequences that can be disastrous for a chemical process [2]. In addition, past information can help understand the current behaviour. Then, the observations are collected, according to their availability. Intuitively, we suspect that when the time horizon is large $t \gg 1$, the data of the initial moment will have no influence on the residuals at the date t . So, it is not necessary to collect all the data. An observation window of size T is then defined. The system is observed during the period T . This window is representative of the system state. Its size is a parameter chosen according to system dynamic. Figure 5.3 illustrates the concept of this sliding window. Next, the detection consists of evaluating an instantaneous signature from the residual generation in the first step (Fig. 5.1). We denote this instantaneous signature S . The instantaneous signature (S) is a positive vector of dimension n (the size of the state vector). More specifically, each component of this vector is a positive real which is the result of a threshold test. An element of the signature is thus defined as follows:

$$S_i(t) = \begin{cases} 0 & \text{if } |r_i(t)| \leq \varepsilon_i(t) \\ \alpha_i > 0 & \text{if } |r_i(t)| > \varepsilon_i(t) \end{cases} \quad \text{with } i \in [1; n] \quad (5.3)$$

where α_i is the result of the threshold violation test; $r_i(t)$ is the generated residual in the first part of the diagnosis, $\varepsilon_i(t)$ is the adaptive detection threshold.

S is an instantaneous default signature. A nonzero component of this vector assumes the occurrence of a fault ($S_i(t) = \alpha_i > 0$ with $i \in [1; n]$). A null vector means a priori a normal behaviour of the monitored system ($S_i(t) = 0$ for $i = 1 \dots n$). This signature vector could be a binary vector: if the residual exceeds the threshold then the signature is equal to 1. Nevertheless, by defining the signature vector in this way, there is a loss of information on the magnitude of the failure: how

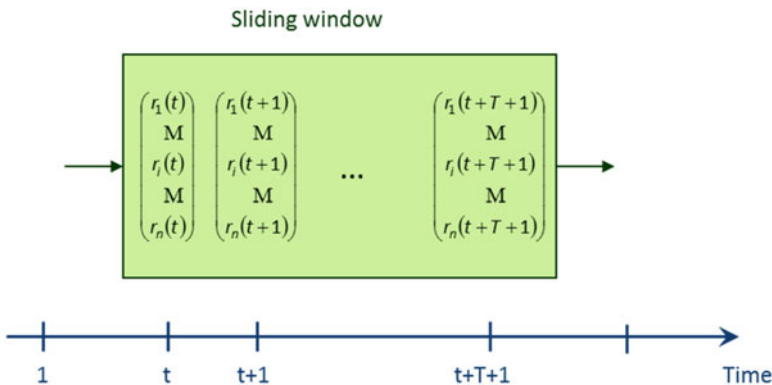


Fig. 5.3 Sliding window

much does it exceed? Is this excess not negligible? By not defining the signature vector as a Boolean, false alarm is thus avoided and the cases of a deviation and of a failure can be differentiated. Moreover, it is hard to detect drift-like fault in early stage. A survey of a drift detection and handling can be found in Sayed-Mouchaweh [11]. In this research work, with the use of a non-binary vector, we have all the necessary information to visualize the effect of a drift on the state vector. Furthermore, thanks to the use of the Kalman filter, it is possible to differentiate drift and model/measurement noises [9].

The instantaneous fault signature $S(t)$ at t is thus a vector function of the residual $r(t)$ and of the detection threshold $\varepsilon(t)$. Each component $S_i(t)$ is defined by the following equation:

$$S_i(t) = \text{Max} [(|r_i(t)| - \varepsilon_i(t)); 0] \quad \text{with } i \in [1; n] \quad (5.4)$$

In the previous point, the interest of relative residual is underlined. In the same way, an instantaneous relative fault is defined and it is a function of the relative residual $r^r(t)$, of the detection threshold $\varepsilon(t)$ and X the state vector:

$$S_i^r(t) = \text{Max} [(|r_i^r(t)| - \varepsilon_i'(t)); 0] \quad \text{with } i \in [1; n] \quad (5.5)$$

with $r_i^r(t) = \frac{\widehat{X}_i(t) - X_i(t)}{X_i(t)}$ and $\varepsilon_i'(t) = \frac{\varepsilon_i(t)}{X_i(t)}$.

Finally, it is interesting to normalize these signatures in order to see the predominant variations. Thus, the normalized relative fault signature is defined by the following equation:

$$S_i^{\text{rN}}(t) = \frac{r_i^r(t)}{\sum_{k=1}^n r_k^r(t)} = \frac{\text{Max} [(|r_i^r(t)| - \varepsilon_i'(t)); 0]}{\sum_{k=1}^n \text{Max} [(|r_k^r(t)| - \varepsilon_k'(t)); 0]} \quad \text{with } i \in [1; n] \quad (5.6)$$

Therefore, the sum of all the components of the normalized relative fault signature is 1. This translates the following heuristic: if a residual r_i^r is sensitive to a fault, then the others r_k^r (with $k \neq i$) are not.

5.4 Determination of the Incidence Matrix

Numerous works deal with the distance to the fault signatures or with the structural properties of the incidence matrix in order to have a robustness fault isolation [12–14]. A fault signature is characteristic of a particular residual and a particular fault. This signature is commonly obtained by experiment (or in our case by simulation). The approach consists of evaluating a signature by comparison between the reference model and the experiment or the simulation of the faulty process

(Fig. 5.1). More specifically, each component of this vector is the result of a threshold test (see Eq. (5.6)).

For our approach, we simulate same fault at different times. We generate the characteristic signature of this fault for the p simulations of the faulty process. The goal is then to have an only representation of this fault. Two cases are envisaged:

- The signatures characterize the same state vector. That means that the p simulations have the same importance: their occurrences are equally likely. The characteristic signature corresponds to the centre of gravity of the p signatures obtained by simulation. For complex systems, it is interesting to analyse the data and to determine their main components. Then, an approximate representation of the p simulations is a subspace of small size.
- The signatures don't characterize the same state vector (different number of state variables). It is then necessary to make a canonical analysis. Consider two sets of simulation characterizing the same fault. The first one is represented by the state vector 1 and the second one by the state vector 2. This analysis consists of examining the links existing between these sets. It is based on a Principal Component Analysis decomposition. This theory is described in [15]. Note that if both spaces are confounded, this means that only one of both sets is necessary, since they have the same power of description. Conversely, if these both sets are orthogonal, both sets do not represent the same properties. It is then necessary to consider two different fault signatures characterizing the same fault.

Let's illustrate this initial learning phase. Consider a system characterized by the state vector $[x, y, z]$. A set of simulations is performed by introducing the same fault at different occurrence dates. Let's represent the results on a graph (Fig. 5.4). We thus obtain a pattern characterizing a fault. However, the fault signature may differ according to system state. That is why we can have different theoretical signatures of a fault for different state or we can have an only one (Fig. 5.4).

Once the global incidence matrix is obtained, it is important to rebuild an incidence matrix adapted to the system state (Fig. 5.1). For this, the incidence matrix is reduced: only the present residuals are used in the instantaneous fault signature. Finally, each theoretical fault signature is normalized. Let's do this on a simple example (Fig. 5.5).

5.5 Fault Isolation

The isolation system is represented in Fig. 5.1. It consists of establishing the diagnosis from measured information of the process (instantaneous fault signature) and from information obtained by experiments or by simulation (theoretical fault signatures).

5.5.1 Principle

The columns of the incidence matrix T represent the fault signatures. The notation adopted for the columns of the incidence matrix is the following: $T_{\cdot j}$ ($j = 1 \dots m$). $T_{\cdot j}$ corresponds to the signature associated with the j th fault f_j . Similarly, each line of the incidence matrix, $T_{i \cdot}$, represents a signature of the i th residual. Figure 5.6 shows an example of theoretical fault signatures and residual signatures of an incidence matrix.

Our approach is similar to a pattern recognition problem. The form to be classified is the instantaneous normalized relative fault signature S^{rn} , generated in the previous step (Fig. 5.1). It is then necessary to assign this pattern to the existent classes. In our case, each class is represented by a theoretical fault signature $T_{\cdot j}$ ($j = 1 \dots m$).

In the case of fault detection and diagnosis, the instantaneous normalized relative fault signature S^{rn} is therefore compared with the m theoretical fault signatures $T_{\cdot j}$ ($j = 1 \dots m$). The signature S^{rn} transcribes the symptoms of the physical system. The vector $T_{\cdot j}$ represents the signature of the j th fault. The fineness of the correlation between these both signatures is directly proportional to the occurrence probability of the fault f_j ($j = 1 \dots m$). So, if it exists $j \in [1; m]$ such as $S^{rn} \cong T_{\cdot j}$, then the diagnosis concludes at the occurrence of the fault f_j .

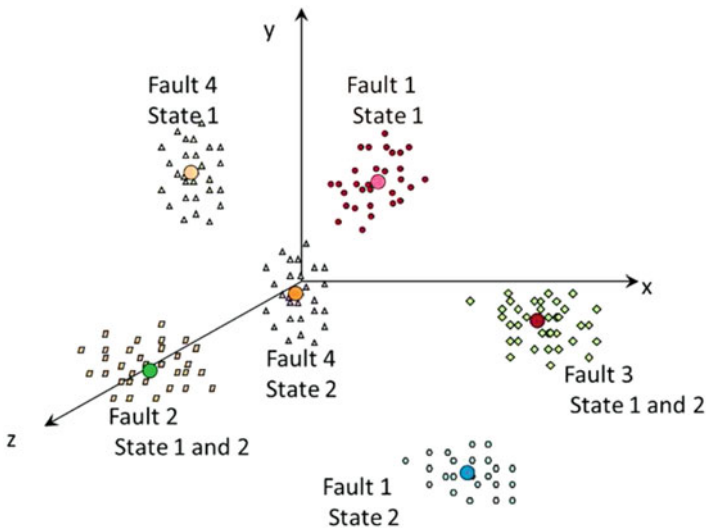


Fig. 5.4 Example of an incidence matrix

Fig. 5.5 Example of the reduction of an incidence matrix

Suppose that the signature obtained is the following:

$$S^{rN} = \begin{pmatrix} 1/2 \\ 1/2 \\ 0 \end{pmatrix}_{r_1, r_3, r_4}$$

The incidence matrix is the following:

$$T = \begin{matrix} & f_1 & f_2 & f_3 & f_4 & f_5 \\ \begin{matrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 1/3 & 1/3 \\ 0 & 1/2 & 1 & 0 & 1/3 \\ 0 & 0 & 0 & 1/3 & 1/3 \\ 0 & 1/2 & 0 & 1/3 & 0 \end{pmatrix} \end{matrix}$$

In this example, the residual r_2 doesn't appear in the instantaneous fault signature. It means that the incidence matrix is not adapted. A work on this matrix is therefore necessary. The 2nd row of the incidence matrix is deleted and the resulting fault signatures are normalized:

$$T' = \begin{matrix} & f_1 & f_2 & f_3 & f_4 & f_5 \\ \begin{matrix} r_1 \\ r_3 \\ r_4 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1/3 & 1/3 \\ 0 & 1/2 & 0 & 1/3 & 0 \end{pmatrix} \end{matrix} \Rightarrow T' = \begin{matrix} & f_1 & f_2 & f_3 & f_4 & f_5 \\ \begin{matrix} r_1 \\ r_3 \\ r_4 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 1/3 & 1/2 \\ 0 & 0 & 0 & 1/3 & 1/2 \\ 0 & 1 & 0 & 1/3 & 0 \end{pmatrix} \end{matrix}$$

Note that the fault f_3 is not isolable here.

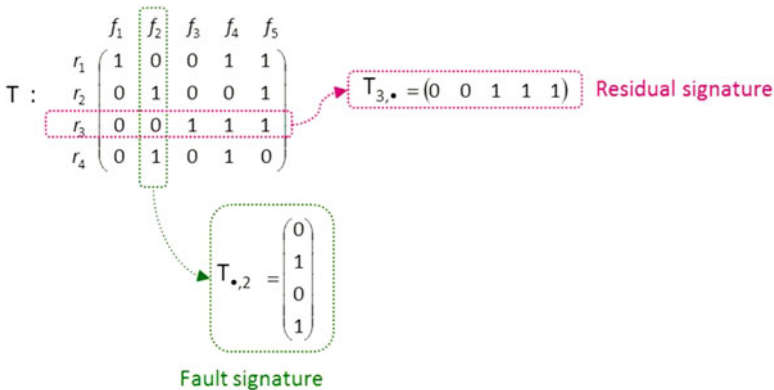


Fig. 5.6 Fault signature and residual signature

In order to compare an instantaneous signature $S^{rN}(t)$ and a particular fault signature $T_{*,j}$, a similarity function or a distance can be used. In our case, the classification is made thanks to a distance in the signature space.

Definition of a Distance Let S be the space of the instantaneous normalized relative signatures and T the bounded space of the theoretical fault signatures ($\text{card}(T) = m$, m being the number of considered faults). A distance $S^{rN}(t)$ defines the correlation

symptoms—faults. The distance between an instantaneous signature and a fault signature $T_{.j}$ is defined by the following expression:

$$D : S \times T \rightarrow [0; 1]$$

$$(S^{\text{rN}}(t); T_{.j}) \alpha D_j(t) = D(S^{\text{rN}}(t); T_{.j})$$

The distance D verifies these following properties:

For $X \in S, Y \in T$,

1. $D(X, Y) = 0 \Rightarrow X = Y$
2. $D(X, Y) = D(Y, X)$
3. For $Z \in S, D(X, Z) \leq D(X, Y) + D(Y, Z)$

Then we define a fault indicator:

Definition of a Fault Indicator A fault indicator $I_j \in [0; 1]$ is specific to the fault f_j with $j = 1 \dots m$. It represents the occurrence probability of the fault. It is defined by the following relation:

$$I_j(t) = 1 - D_j(t) = 1 - D(S^{\text{rN}}(t), T_{.j}) \quad (5.7)$$

According to the property 1 of a distance, $I_j(t) = 0$ means that the fault f_j is not occurring. On the contrary, $I_j(t) = 1$ reflects the fact that the fault f_j is detected and localized.

In general, we will not have these strict equalities, but rather the relation of order: $0 < I_j(t) < 1$.

This relationship triggers an alarm on the fault f_j . If the fault indicator I_j is close to zero, the occurrence of the fault is not proved. On the other hand, if I_j is close to one, then the occurrence of the fault f_j is demonstrated.

5.5.2 Distances

Generally, the distance used is Hamming distance [16, 17]. It is a mathematical distance. It compares two binary vectors B_1 and B_2 of the same size. This distance is equal to the sum of the absolute values of the differences, component by component of the two vectors B_1 and B_2 :

$$D^H = \sum_{i=1}^n |B_{1i} - B_{2i}| \quad (5.8)$$

Figure 5.7 illustrates the calculation of the Hamming distance between two binary vectors B_1 and B_2 . In this example, the Hamming distance is equal to 1.

$$B_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix} ; B_2 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \rightarrow D^H = \sum_{i=1}^4 |B_{1i} - B_{2i}| = |0-1| + |1-1| + |0-0| + |1-1| = 1$$

Fig. 5.7 Example of a calculation of Hamming distance

That means that only one component is different between both vectors. In order to standardize this distance for all signatures, the relative Hamming distance has been defined [18]. This distance between two binary vectors B_1 and B_2 is defined by the following expression [19]:

$$D^{\text{Hr}}(t) = \frac{\sum_{i=1}^n |B_{1i} - B_{2i}|}{n} \quad (5.9)$$

In the previous section, we underline the interest to work in the continuous space $[0;1]$. Equation (5.8) can be generalized to non-binary vectors: in this case the distance is called *Manhattan distance*. In the same way, we generalize Eq. (5.9) to the non-binary case and thus define a new distance called: relative Manhattan distance [1, 9, 10]. The demonstration of this definition can be found in [10].

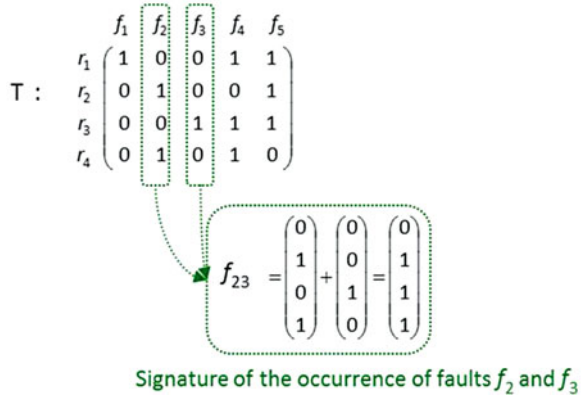
Definition of Relative Manhattan Distance Let S be the space of the instantaneous normalized relative signatures and T the bounded space of the theoretical fault signatures ($\text{card}(T) = m$, m being the number of considered faults). The relative Manhattan distance between an instantaneous signature $S^{\text{N}}(t)$ and a particular fault signature $T_{\cdot,j}$ (both of size n) is defined by the following relation:

$$D_j^{\text{Mr}}(t) = \frac{\sum_{i=1}^n |S_i^{\text{N}}(t) - T_{ij}|}{n} \quad (5.10)$$

One of the major problems of FDI systems is their ability to detect the occurrence of multiple faults and to localize them. Indeed, the theoretical signatures characterize a particular fault. However, the occurrence of multiple fault is represented by a new fault signature [20]. This signature is obtained by combining the theoretical fault signatures [19]. This is illustrated in Fig. 5.8.

Taking into account all the linear combinations of the theoretical signatures is not a satisfactory solution because of the combinatory explosion. It is therefore necessary to use a method which avoids the combination tests. Thus, Theillol et al. [18] have defined a modified Hamming indicator, which only takes into account the nonzero elements of the theoretical fault signature in the comparison:

Fig. 5.8 Signature of multiple faults



$$D_j^{\text{Ha}}(t) = \frac{\sum_{i=1}^n |S_i^{\text{rN}}(t) - T_{ij}| \cdot T_{ij}}{n'} \tag{5.11}$$

with n' the number of nonzero elements of the theoretical fault signature $T_{\bullet,j}$. We generalize this distance to the non-binary case by defining the improved Manhattan distance D^{Ma} [10]:

$$D_j^{\text{Ma}}(t) = \frac{\sum_{i=1}^n |S_i^{\text{rN}}(t) \times m' - T_{ij} \times n'| \cdot T_{ij}}{n'} \tag{5.12}$$

with n' the number of nonzero elements of the theoretical fault signature $T_{\bullet,j}$, m' the number of nonzero elements of the instantaneous fault signature S^{rN} .

Note

Improved Hamming and Manhattan distances are not mathematical distances [10]. Nevertheless, these indicators are called “distance”, since these both indicators allow to make a comparison between the instantaneous signature S^{rN} and a particular fault signature $T_{\bullet,j}$ in terms of similitude of abnormal symptoms.

Let’s apply the relative and improved Manhattan signatures to a concrete example. Consider the case where the faults f_1 and f_2 take place simultaneously. The instantaneous signature vector and the incidence matrix are shown in Fig. 5.9. These distances (Eqs. (5.10) and (5.12)) and the corresponding fault indicators (Eq. (5.7)) are calculated.

In this example, the calculation of the relative Manhattan fault indicators does not allow us to conclude. The instantaneous fault signature does not correspond to any theoretical fault signatures. The improved Manhattan distance is based on the idea of finding in the instantaneous fault signature only the significant symptoms

Consider the following instantaneous fault signature and incidence matrix:

$$S^{rN} = \begin{pmatrix} 1/3 \\ 1/3 \\ 0 \\ 1/3 \end{pmatrix} \quad T = \begin{matrix} & f_1 & f_2 & f_3 & f_4 & f_5 \\ \begin{matrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 1/3 & 1/3 \\ 0 & 1/2 & 0 & 0 & 1/3 \\ 0 & 0 & 1 & 1/3 & 1/3 \\ 0 & 1/2 & 0 & 1/3 & 0 \end{pmatrix} \end{matrix}$$

Relative Manhattan distance: $D_i^{Mr} = \frac{\sum_{k=1}^4 |S_k^{rN} - T_{ki}|}{4}$

	f_1	f_2	f_3	f_4	f_5
$D^{Mr} =$	0,33	0,17	0,5	0,17	0,17
$I^{Mr} =$	0,67	0,83	0,5	0,83	0,83

← Unable to conclude

Improved Manhattan distance: $D_i^{Ma} = \frac{\sum_{k=1}^4 |S_k^{rN} \times m' - T_{ki} \times n'| \cdot T_{ki}}{n'}$ with $m'=3$

	f_1	f_2	f_3	f_4	f_5
$D^{Ma} =$	0	0	1	0,11	0,11
$I^{Ma} =$	1	1	0	0,89	0,89

← Detection and isolation of the faults f_1 and f_2

Fig. 5.9 Example of Manhattan distances and corresponding fault indicators

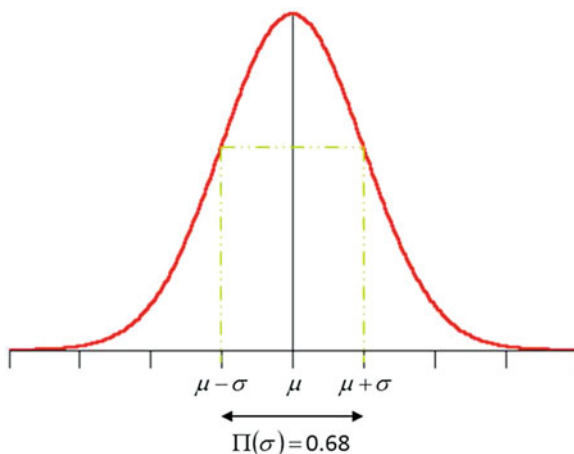
of faults (i.e. the nonzero elements). With the improved Manhattan indicator, both faults f_1 and f_2 are detected and isolated.

5.5.3 Decision Making

The generated fault indicators are then transmitted to the decision step (Fig. 5.1). This step consists of the discrimination of the most probable fault. Since both distances are defined in the space interval [0;1], the fault indicators are defined as the complement to 1 of these distances. An indicator can be viewed as the probability of the occurrence of a particular fault. These indicators follow a reduced centred normal law $\mathfrak{N}(\mu, \sigma)$. This distribution is shown in Fig. 5.10. This is confirmed by the well-known statistical test of Shapiro-Wilk [21, 27]. This test is used to verify normality. According to the test value, we can accept or reject the hypothesis that the corresponding distribution is normal. The Shapiro-Wilk W test is the most widely used normality test because it is a powerful test compared to many alternative tests [15].

The generated fault indicators are exploited to take a diagnosis of the system. To make this decision, we formulated two postulates:

Fig. 5.10 Reduced centred normal law $\mathfrak{N}(\mu, \sigma)$



- A minimum value of the indicator, for which the fault can be considered, is defined. This threshold is equal to 0.68 and corresponds to the probability at the standard deviation. This allows us to define a limit threshold corresponding to the probability at the standard deviation, i.e. less than 0.68. Thus, the presence of a fault is not valid if its indicator is less than 0.68.
- Then, in order to limit the choice of possible faults, the following hypothesis is put forward: the number of faults, which can simultaneously take place, is limited to three.

5.6 Monitoring of a Complex Chemical Process

In this example, the case study deals with a variant of a chemical process described in [22]. The process is described in Fig. 5.11. The purpose of this installation is to produce and package a product P whose molar purity must be equal to 98%. The reaction considered is an endothermic balanced reaction, whose reaction is the following:



In order to maximize the conversion rate of the reaction without penalizing the cycle time of the process, the reaction is stopped as soon as the molar composition of product P reaches the value of 0.8. Moreover, the reaction (R) speed increases with temperature T . The selected temperature for the reaction must guarantee a rapid reaction and maintain the components in the liquid state. A temperature of 383 K satisfies these two constraints. Discrete controller commands the valves (open valve/close valve).

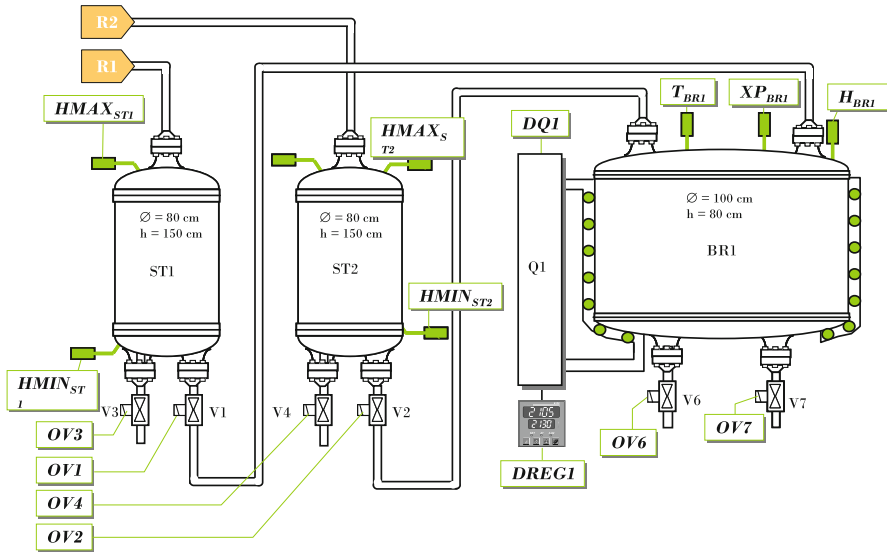


Fig. 5.11 Process flowsheet [22]

The production of P according to the reaction (5.13) in the reactor BR1 involves the following steps:

- Introduction in the reactor with $n/2$ moles of product $R1$,
- Preheating to 383 K,
- Introduction of $n/2$ moles of product $R2$ in the reactor with a temperature control with the set point 383 K,
- Reaction until the product composition P reaches the value of 0.8.

5.6.1 Simulation of the Reference Model

The models used in this simulation take into account global and partial material balances, energy balance, liquid/vapor equilibria, reaction rates, and hydraulic phenomena. Indeed, except the pipes with a pump, the transfers between tanks are carried out by gravity. This implies that the outlet flows of the tanks are a function of the hydraulic pressure and of the liquid level in the source tanks. The transfer times therefore depend on the time evolution of the system state. The simulation is made with the hybrid dynamic simulator PrODHyS. The reader can find more information about PrODHyS in [23]. Figure 5.12 illustrates the time evolution of the composition in the reactor.

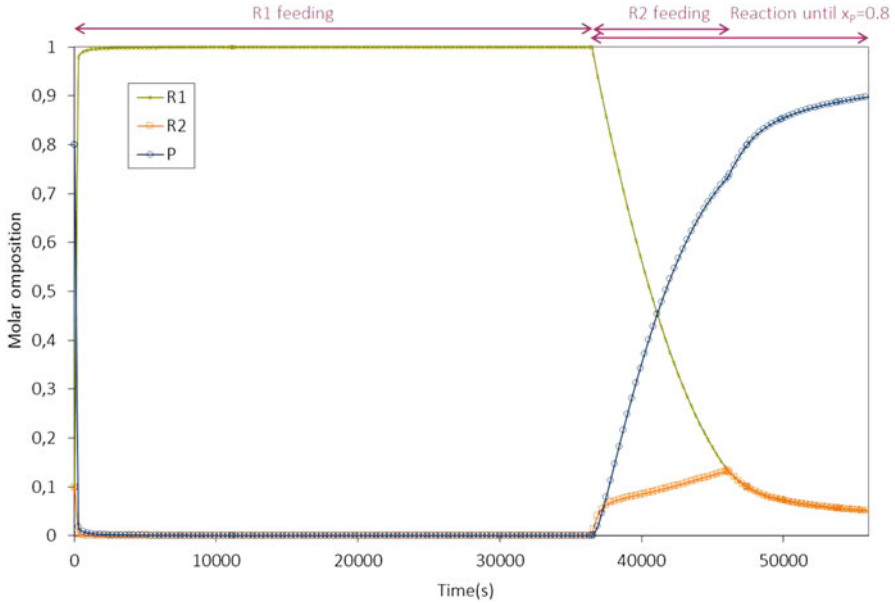


Fig. 5.12 Time evolution of the composition in the reactor

5.6.2 Detection

The proposed approach is illustrated using a chemical process. The fault studied concerns a degradation: the flow rate in valve $V2$ is degraded. That means that valve $V2$ is blocked off partially. It is very interesting to be able to detect and diagnosis a drift in order to avoid the failure [24]. We can find in literature numerous works dealing with this problem [11].

For this case study, 17 signatures related to a physical quantity are considered:

- The signature s_1 represents the flow rate in the valve $V1$,
- The signature s_2 represents the flow rate in the valve $V2$,
- The signature s_3 represents the $R1$ composition in the tank $ST1$,
- The signature s_4 represents the $R2$ composition in the tank $ST1$,
- The signature s_5 represents the P composition in the tank $ST1$,
- The signature s_6 represents the liquid retention in the tank $ST1$,
- The signature s_7 represents the $R1$ composition in the tank $ST2$,
- The signature s_8 represents the $R2$ composition in the tank $ST2$,
- The signature s_9 represents the P composition in the tank $ST2$,
- The signature s_{10} represents the liquid retention in the tank $ST2$,
- The signature s_{11} represents the liquid level in the tank $BR1$,
- The signature s_{12} represents the $R1$ composition in the tank $BR1$,
- The signature s_{13} represents the $R2$ composition in the tank $BR1$,

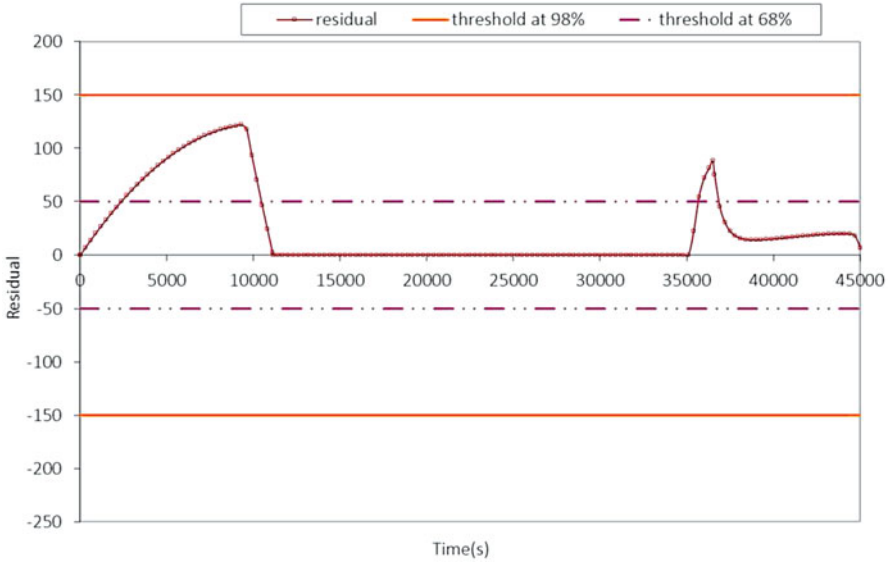


Fig. 5.13 Detection of a drift

- The signature s_{14} represents the P composition in the tank BR1,
- The signature s_{15} represents the temperature in the tank BR1,
- The signature s_{16} represents the liquid retention in the tank ST1,
- And the signature s_{17} represents the heat provided by the power supply of the tank BR1.

Figure 5.13 illustrates the detection step. The residual of the liquid retention for the reactor BR1 is presented. A statistical analysis estimates the prediction errors of the Kalman filter and determines a limit threshold of 150 moles. This threshold corresponds, according to the normal law, to a probability of 98%: there is a probability of 98% that the behaviour is normal in this interval. The obtained residual remains in this confidence interval. That means that this threshold (98%) is not adapted to detect degradation. The threshold must be changed for that and must be lower than this one. A compromise is made to avoid false alarms. For this goal, the same postulate as the fault indicators is formulated: the threshold is lower to a probability of 68% (Fig. 5.10): the new threshold is then obtained at 50 moles. From $t = 2400$ s, the residual is out of the normal operating area. The diagnosis is launched at $t = 3000$ s.

The residual vector is then evaluated and the corresponding instantaneous fault signature is obtained:

5.6.3 Diagnosis

The considered faults in this example are chosen according to a risk assessment study and lessons to learn from accident:

- Fault 1 corresponds to a fault on the power supply of the reactor BR1: the latter supplies a degraded amount of energy.
- Similarly, fault 2 represents a fault in the cooling system of the reactor BR1 which provides a degraded amount of energy.
- Fault 3 is a composition fault on the tank ST1, which normally contains the pure component $R1$. Here, traces of component $R2$ are found in this tank.
- Fault 4 characterizes the same fault but this time there are traces of the constituent P .
- The same type of fault is also considered on the tank ST2 which normally contains the pure component $R2$. Thus, fault 5 represents the fact that component $R1$ exists in tank ST1.
- Fault 6 is the same fault but with component P .
- Fault 7 represents a fault in the reactor power supply which is not at the right temperature.
- Next, actuator faults are considered with the fault 8 corresponding to the blocking of the valve $V1$ in the open position,
- And with the fault 9 corresponding to a degraded state of the valve $V1$: the flow rate of this valve is degraded.
- The fault 10 is identical to the fault 8 but for the valve $V2$.
- Similarly, fault 11 is the same fault as fault 9 but applies to valve $V2$.

The incidence matrix contains all the theoretical fault signatures. An off-line Monte Carlo simulation provides the theoretical signatures. It consists of simulating a fault with different occurrence date. The parameter of the faults change for each simulation and the noises are simulated. For example, consider the fault 9. The flow rate of valve $V1$ is degraded. The valve is blocked off partially due to fooling. The value of the rate of fooling changes. This matrix is rebuilt on-line to match the state vector. This stage has been developed in point 4.

The instantaneous fault signature (Table 5.1) is compared with the incidence matrix by calculating the relative fault indicators for the relative Manhattan distances (Eq. (5.10)) and the improved one (Eq. (5.12)). The obtained indicators are presented in Table 5.2.

Table 5.1 Instantaneous fault signature

s_1	0.21775356	s_7	0	s_{13}	0.16505573
s_2	0	s_8	0	s_{14}	0.16505354
s_3	0	s_9	0	s_{15}	0
s_4	0	s_{10}	0	s_{16}	0.21909804
s_5	0	s_{11}	0.21952267	s_{17}	0
s_6	0.01182547	s_{12}	0		

Table 5.2 Relative and improved Manhattan indicators

	Relative Manhattan indicator	Improved Manhattan indicator
Fault 1	0.92118933	0.84052179
Fault 2	0.92118933	0.86669003
Fault 3	0.90196728	0.15177246
Fault 4	0.90277453	0.19006373
Fault 5	0.88238145	0.0003779
Fault 6	0.88236707	0.00018733
Fault 7	0.92118933	0.86669003
Fault 8	0.92335219	0.67944269
Fault 9	0.98670897	0.95493963
Fault 10	0.92383701	0.84716655
Fault 11	0.99785443	0.99547681

The values 0.68 of the fault indicators do not allow us to avoid faults since all the values are greater than 0.68. On the other hand, the improved fault indicator makes it possible to eliminate faults 3, 5, 6 and 8. We therefore have 6 possible faults. We then use the second hypothesis that we formulated (see point 5.3): there can be no more than three simultaneous faults. Thus, only the indicators with the highest values are kept:

- Fault 11 with a rate of more than 99%,
- Fault 9 with a 95% rate,
- And faults 2 and 7 which have indicator values equal to 98.7%.

By combining the results of both indicators, it is found that fault 11 is in both cases with a rate of more than 99%, and in particular the fault which provides the maximum indicators. We can therefore conclude on the most probable cause of the failure: fault 11, which represents the degraded state of valve V2 (the flow rate is lower than the normal one).

The value of the residual then reveals the magnitude of the deviation, i.e. about 0.1. A parametric estimate here would be profitable in order to more precisely determine the opening coefficient of the valve. However, in view of the results, the system is in degraded mode. It may be considered to leave it in this state. In this case, it is interesting to take this degradation into account in the reference model. Finally, we can conclude that the SimAEM methodology is able to detect and diagnose degradation.

5.7 Conclusion

This chapter presents a model-based approach and this methodology is illustrated with the simulation of a complex chemical process. The feasibility of using the simulation as a tool for fault detection is described. The method developed in this study relies on the hybrid dynamic simulator (PRODHYS). The fault detection

and diagnosis approach, developed here, is a general method for the detection and isolation of occurrence of a fault. Besides, this approach allows the detection of numerous types of fault and has the ability to detect and isolate simultaneous faults [1]. The works in progress aim at defining a recovery solution following the diagnosis of fault. For this, the results of signatures will be exploited in order to generate qualitative information. As shown by the example, it is possible to distinguish a simple degradation from a failure. Finally, dynamic simulation of faulty processes is a real asset for safety studies. It makes it possible to analysis the drifts to evaluate their dynamic and their magnitude and thus to define the required safety barriers. Moreover, the simulation results provide predictive information to validate the nature and the sizing of barriers.

References

1. Olivier-Maget, N., Hétreux, G., Le Lann, J. M., & Le Lann, M. V. (2009). Model-based fault diagnosis for hybrid systems: Application on chemical processes. *Computers & Chemical Engineering*, 33(10), 1617–1630.
2. Venkatasubramanian, V., Rengaswamy, R., Yin, K., & Kavuri, S. N. (2003). A review of process fault detection and diagnosis. *Computers & Chemical Engineering*, 27, 293–346.
3. De Kleer, J. (1986). An assumption-based TMS. *Artificial Intelligence*, 28, 127–162.
4. Birouche, A. (2006). *Contribution sur la synthèse d'observateurs pour les systèmes dynamiques hybrides*. Thèse de doctorat, Institut National Polytechnique de Lorraine, Nancy, France.
5. Ding, S. X. (2014). Data-driven design of monitoring and diagnosis systems for dynamic processes: A review of subspace technique based schemes and some recent results. *Journal of Process Control*, 24(2), 431–449.
6. Clark, R. N., Fosth, D. C., & Walton, V. M. (1975). Detection instrument malfunctions in control systems. *IEEE Transactions on Aerospace and Electronic Systems*, AES-11, 465–473.
7. Jazwinski, A. H. (1970). *Stochastic processes and filtering theory, Mathematics in Science and Engineering* (Vol. 64). New York: Academic Press.
8. Reif, K., & Unbehauen, R. (1999). The extended Kalman filter as an exponential observer for nonlinear systems. *IEEE Transactions on Signal Processing*, 47(8), 2324–2328.
9. Olivier-Maget, N., Hétreux, G., Le Lann, J. M., & Le Lann, M. V. (2009). Dynamic state reconciliation and model-based fault detection for chemical processes. *Asia Pacific Journal of Chemical Engineering*, 4(6), 929–941.
10. Olivier-Maget, N. (2008). *Surveillance des systèmes dynamiques hybrides: Application aux procédés*. Thèse de doctorat, Université de Toulouse, France.
11. Sayed-Mouchaweh, M. (2016). *Learning from data streams in dynamic environments, Springer briefs in electrical and computer engineering* (p. 75). Cham: Springer. ISBN: 978-3-319-25665-8.
12. Chin, H., & Danai, K. (1991). A method of fault signature extraction for improved diagnosis. In *IEEE ACC Conference*, Boston, USA.
13. Fang, C. Z., & Ge, W. (1998). Failure isolation in linear systems. In *IMACS 12th world congress*, Paris, France, pp. 442–446.
14. Gertler, J., & Singer, D. (1990). A new structural framework for parity equation based failure detection and isolation. *Automatica*, 26(2), 381–388.
15. Saporta, G. (1990). *Probabilités, analyse des données et statistique*. Paris: Éditions Technip.
16. Cassar, J. P., Litwak, R.-G., Cocquempot, V., & Staroswiecki, M. (1994). Approche structurelle de la conception de systèmes de surveillance pour les procédés industriels. *Diagnostic et Sécurité de Fonctionnement*, 4(2), 179–202.
17. Kaufmann, A. (1977). *Introduction à la théorie des sous-ensembles flous à l'usage des ingénieurs*. Tomes I et II, Masson.

18. Theillol, D., Weber, P., Ghetie, M., & Noura, H. (1995). A hierarchical fault diagnosis method using a decision support system applied to a chemical plant. In *International Conference on Systems, Man, and Cybernetics*, Canada.
19. Ripoll, P. (1999). *Conception d'un système de diagnostic flou appliqué au moteur automobile*. Thèse de doctorat, Université de Savoie, France.
20. Koscielny, J. M. (1993). Method of fault isolation for industrial processes. *Diagnostic et Sécurité de Fonctionnement*, 3(2), 205–220.
21. Olivier-Maget, N., & Hétreux, G. (2016). Fault detection and isolation for industrial risk prevention. *Journal Européen des Systèmes Automatisés*, 49(4-5), 537–557.
22. Joglekar, G. S., & Reklaitis, G. V. (1985). A simulator for batch and semi-continuous processes. *Computers and Chemical Engineering*, 8(6), 315–327.
23. Olivier-Maget, N., Hétreux, G., Le Lann, J. M., & Le Lann, M. V. (2008). Integration of a failure monitoring within a hybrid dynamic simulation environment. *Chemical Engineering and Processing: Process Intensification*, 47(11), 1942–1952.
24. Toubakh, H., & Sayed-Mouchaweh, M. (2016). Hybrid dynamic classifier for drift-like fault diagnosis in a class of hybrid dynamic systems: Application to wind turbine converters. *Neurocomputing*, 171, 1496–1516.
25. Einicke, G. A., & White, L. B. (1999). Robust extended Kalman filtering. *IEEE Transactions on Signal Processing*, 47(9), 2596–2599.
26. De Kleer, J., & Williams, B. C. (1987). Diagnosing multiple faults. *Artificial Intelligence*, 32, 97–130.
27. Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4), 591–611.

Chapter 6

Hybrid Bond-Graph Possible Conflicts for Hybrid Systems Fault Diagnosis



Carlos J. Alonso-González, Belarmino Pulido, and Anibal Bregon

6.1 Introduction

Hybrid systems are physical systems that exhibit continuous behavior which can be modified due to changes in their configuration. Their presence in our everyday life is pervasive: the anti-lock braking system (ABS) commonly found in the automotive industry; heating, ventilation, and air conditioning (HVAC) systems that can provide heat and/or cold air; and aircraft or factories that work in different operation modes are well-known examples. Hybrid systems exhibit complex behavior, made up of a mix of continuous and discrete dynamics, being extremely challenging to track its current state, either nominal or faulty. Such tracking is an essential task in order to keep the system working in a nominal and safe state.

The focus of this work is on the kind of hybrid systems that have continuous behavior controlled by discrete events. In those systems, the main source of hybrid behavior are discrete actuators, like valves or switches in fluid or electrical systems, respectively. A fault in an actuator, named a discrete fault, affects system dynamics, usually causing a mode change, thus modifying system behavior in a different way than a parametric fault [1]. Fault detection and isolation not only must be fast, but it must be performed among continuous mode changes. As a consequence, using a unique diagnosis framework capable to cope with both types of faults would ease the task. In this work we propose a model-based diagnosis framework capable to diagnosis both parametric and discrete faults.

C. J. Alonso-González (✉) · B. Pulido · A. Bregon
Departamento de Informática, Universidad de Valladolid, Valladolid, Spain
e-mail: calonso@infor.uva.es; belar@infor.uva.es; anibal@infor.uva.es

Hybrid systems modelling and diagnosis have been approached by the Control Theory and the Artificial Intelligence communities¹ in the last 20 years. In the FDI field several approaches have been developed to diagnose hybrid [2] or quantized systems [3]. Meanwhile, in the DX field, different proposals have been made based on hybrid modelling [4, 5], hybrid state estimation [6, 7], or combination of on-line state tracking and residual evaluation [8, 9]. All these proposals have at least one of the following difficulties: either it is necessary to pre-enumerate all possible configurations or working modes in the system (and to provide models for all of them), or they need to determine somehow the actual working mode, including the actuators' configuration and the continuous behavior. To overcome some of these difficulties several authors have proposed Hybrid Bond Graphs (HBGs) [10, 11] as an alternative modelling technique, because the enumeration of the whole set of configurations is not required. HBG modelling is an extension for hybrid systems of the well-known Bond-Graph (BG) modelling approach [12, 13], that provides a graphical description of the system model. Hybrid behavior is introduced by means of idealized switching junctions that connect or disconnect parts of the system. Different kinds of numerical equations can be automatically derived from BG and HBG graphical models, and these equations can be later used to simulate or to diagnose system behavior.

Our proposal for hybrid systems diagnosis is an extension of consistency-based diagnosis using Possible Conflicts (PCs) [14]. A PC is a minimal overdetermined set of equations that can be used to check different diagnosis hypotheses. PCs were defined as sets of algebraic or differential equations, but the concept was later extended to work with BG models [15]. Initially PCs were computed from the Temporal Causal Graph (TCG) associated with the BG once causality was assigned using the Sequential Causal Assignment Process (SCAP) algorithm.² Later on, we proposed to derive PCs for hybrid systems using HBGs [16], that were named Hybrid PCs (HPCs). Main assumption in that approach was that there was a valid causal assignment (VCA)³ for any bond in the model in integral causality when every switching junction was set to ON. The causality within the HPC models could be changed using HSCAP, which is the extended version of SCAP for hybrid systems, whenever a mode change was detected [17]. In this work we remove that requirement and provide a formal characterization of HPCs in the HBG framework that will be called Hybrid Bond-Graph Possible Conflict (HBG-PC). We achieve this goal introducing the concept of Structural Hybrid Bond Graph Possible Conflict, SHBG-PC.

Additionally, we provide a common framework for Fault Detection and Isolation of discrete and parametric faults. Once a fault is detected, and it can be related to a discrete fault, they will be preferred candidates instead of any parametric fault,

¹Frequently known also as the FDI and the DX communities, respectively.

²Causality in a bond represents how the underlying equation is solved, i.e. which variable is dependent upon the rest of variables in the equation.

³A VCA provides a causal assignment for every bond in the BG/HBG model.

because they introduce highly non-linear behavior. Usually a discrete fault will change the current state. We propose to track the residual for every possible new state, and to reject those states whose residuals do not become zero. Only if discrete faults are rejected, we start the parametric fault isolation stage. To achieve this unifying solution we will use different structural and qualitative information derived directly from the HBG-PC models: the Hybrid Fault Signature Matrix (HFSM), the Reduced Qualitative Fault Signature Matrix (RQ-FSM), and the Hybrid Qualitative Fault Signature Matrix (HQFSM). Each Fault Signature Matrix (FSM) will be used at different stages to reduce the search space among potential state changes or potential fault candidates, rejecting those states or modes that are not consistent with current observations.

The organization of this work is as follows: first, we introduce concepts about BG modelling, and the PC approach for consistency-based diagnosis in the BG framework. Afterwards we extend PCs to HBG terminology, and provide the algorithms to compute SHBG-PCs. Later on we introduce the common diagnosis framework for both parametric and discrete faults, showing its performance on a four tank simulation case study. We finish by discussing about related work and drawing some conclusions.

6.2 Characterizing PCs in the BG Modelling Framework

In this section we briefly introduce the concept of hybrid bond-graphs for modeling of hybrid systems and present the Possible Conflicts diagnosis approach from a Bond-Graph perspective.

6.2.1 Hybrid Bond-Graphs for Hybrid Systems Modelling

Hybrid Bond Graphs extend Bond-Graphs by including idealized switching junctions, SW for short, to allow configuration changes in the system. If a SW is set to *ON*, it behaves as a regular junction. When it changes to *OFF*, all bonds incident on the junction are deactivated forcing 0 flow (or effort) for 1 (or 0) junctions.

Figure 6.1a, b shows the two configurations of an ideal 1-SW with three bonds. The associated bond graphs describe the casual assignment, with the usual graphical notation that we will introduce in the next section. Note that the casual assignment of the *OFF* configuration is mandatory. Transitions on an ideal SW are implemented as a finite state machine *control specification (CSPEC)*. Transitions between the CSPEC states can be triggered by endogenous or exogenous variables, called guards. CSPECs capture controlled and autonomous changes as described in [17]. Figure 6.1c shows the automata for a potential CSPEC being the SW SW_1 set to *ON*, represented as sw_1 , or *OFF*, represented as $!sw_1$.

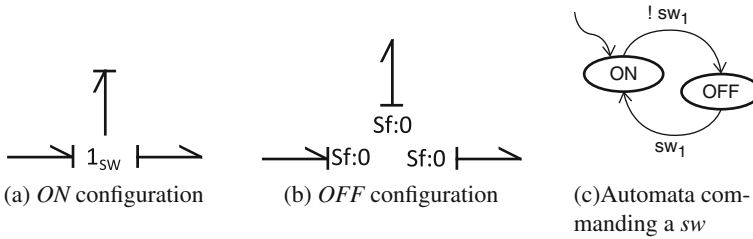


Fig. 6.1 Semantics of a 1 switching junction in (a) and (b). Automata associated with the commanded transition for SW_1 from ON to OFF and vice versa in (c)

6.2.2 Possible Conflicts from BG Models

The PC approach is a dependency-compilation technique from the DX community [14] for consistency-based diagnosis in continuous systems. PCs compile offline those minimal structurally overdetermined subsets of equations from the system model capable of generating fault hypotheses from observed measurement deviations, i.e. they are the basis to check the consistency between observed and estimated variables in the system.

The structural and causal information required to compute the set of PCs can be derived automatically from a set of equations, which can be a set of Algebraic Differential Equations or a Bond-Graph model [15].

To extend the PCs approach for hybrid systems diagnosis, we chose the HBG models, because it is not necessary to enumerate every working mode beforehand [18]. Moreover, efficient proposals exist to automatically change the causality in the model whenever a change in the system mode occurs [17].

We initially relied upon two main assumptions to compute the set of PCs: it was possible to use integral causality to solve the underlying equations, and there was a complete valid causal assignment for the system model when every SW was set to ON. These subsystems were called Hybrid PCs [16]. Now we propose to remove both assumptions and still compute the set of Hybrid PCs. Main reason is that some systems do not have a VCA under integral causality when every SW is ON, because such configuration will never be set. We will illustrate this phenomena with our motivating example, that we will introduce in Sect. 6.4.

To avoid such requirement we propose to compute the set of Hybrid PCs without initially considering causality. We will call these subsystems Structural Hybrid Bond-Graph Possible Conflicts (SHBG-PCs). In order to define SHBG-PCs we proceed first by defining PCs for BG modelling, and then extending them to HBG models. To illustrate the intuitions behind these definitions we will use the Bond Graph model example shown in Fig. 6.2.

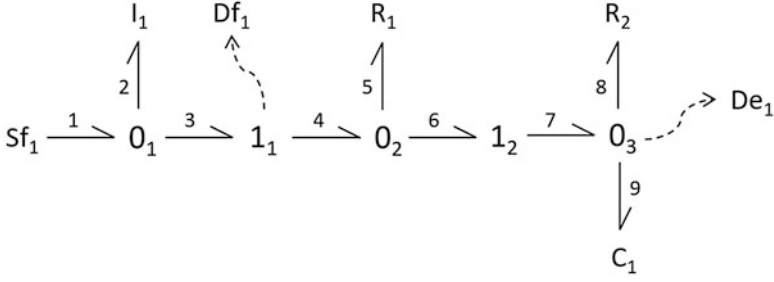


Fig. 6.2 Bond Graph model example

Definition 1 (Bond Graph (BG)) A BG is a connected graph made up of elements and bonds: $\{E, B\}$, where $E = St \cup M$. M stands for measurements or sensors, (De, Df) , and St , the set of structural elements, is made up of $St = S \cup PSV \cup J_t$. S represents effort or flow Source elements (Se, Sf). PSV contains passive elements (resistors, R , capacitors, C , or inductance elements, I). J_t is the total set of junctions: $J_t = J \cup T$, where T are transformers, TF , or gyrators, GY , and J is the set of 0- or/and 1-junctions.

In a BG elements are connected by means of bonds, defined by $B \subset E \times E$, and meaning that not every relation between elements e_i, e_j is allowed for each bond $b_k \in B$. In fact, for each $(e_i, e_j) \in B$, $e_i \in J_t$ or $e_j \in J_t$ or $\{e_i, e_j\} \subset J_t$.

Exceptionally there could be combinations of one source and one passive element that would not respect that generic rule, but we do not consider those systems as significant for fault diagnosis.

Moreover, BGs are usually extended by adding a number to each bond, in order to facilitate the enumeration of each effort and flow variable. Each bond $b_k \in B$ represents a relation or equation among system (effort and flow) variables. The elements in $S \cup PSV$ provide the behavioral model by means of the set of its constituent equations. The elements of J_t provide the structural model of the system. The set M determines which variables in the system can be observed (the observational model).

In the BG model example of Fig. 6.2 we have two sensors, $M = \{Df_1, De_1\}$, one flow source, $S = \{Sf_1\}$, four passive elements, $PSV = \{I_1, R_1, R_2, C_1\}$, five junctions $J = \{0_1, 0_2, 0_3, 1_1, 1_2\}$, connected by means of nine bonds.

Causality expresses computational dependencies between effort and flow variables in a BG [17]. A BG with a valid global causal assignment and without sensors defines a just-determined set of equations, where S elements are the exogenous variables or inputs [19]. This kind of BG is known as a Causally Enhanced BG [18] or Causal BG [19]:

Definition 2 (Causal BG) It is a $BG = \{E, B\}$, where each bond, $b_i \in B \subset E \times E$ is extended with a label *causality* = {"effort", "flow"}, that signals which variable (effort or flow) fixes the causality in the bond: $b_i = (e_i, e_j, causality)$.

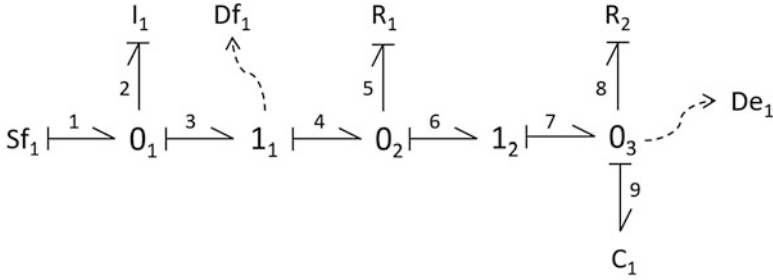


Fig. 6.3 Bond Graph model example with a valid causal assignment

Causality in each bond is graphically depicted as a stroke (that indicates the direction of the effort) in the edges. Once causality is assigned to the bonds in a BG model, we know how to use the equations for behavior simulation. The SCAP algorithm [12] has been used to assign causality automatically to the BG. Figure 6.3 shows a valid causal assignment, under integral causality assumption, for the BG example in Fig. 6.2.

There are a set of rules that govern causality assignment in a BG model [19]. Usually, those rules do not prescribe a unique causal assignment and some arbitrary choices have to be made to fix causality.

Adding sensors in a BG introduces analytical redundancy in the system model, because we can at least estimate and observe each variable related to the sensor. As it is the usual procedure in model-based fault diagnosis, sensors are the potential source of discrepancies. This is the main idea behind building Analytical Redundancy Relations (ARRs) or Diagnostic Bond-Graphs (DBGs) for FDI using BGs [19]. PCs also rely on these concepts although they were not originally defined on the BG framework. Extending the concept of PCs to BGs requires finding the set of subsystems in a BG with minimal analytical redundancy, which in turn requires introducing the three following definitions.

Definition 3 (Degenerated Junction (J_d)) A degenerated 1-j (equivalently 0-j) is a one-port element that must be obtained from a valid 1-j (equiv. 0-j) in a BG that is connected to a flow sensor Df (equiv. effort, De) or a flow source, Sf (equiv. effort, Se). Given a bond, b , and a measurement Df_1 , the 1-degenerated junction (equiv. 0-j) changes the junction behavioral model:

- $f_b := Df_1$ (instead of the set of equalities $f_a = f_c = f_b$ for a 3-port 1-junction with determining bond b). If b is linked to a source, the equation would be $f_b := Sf_1$.
- there is no restriction for the conjugated variable, e_b (instead of $e_b = e_a + e_c$).

Degenerated junctions provide the value for exactly one variable, of exactly the same type (effort/flow) of the adjacent measurement or source.

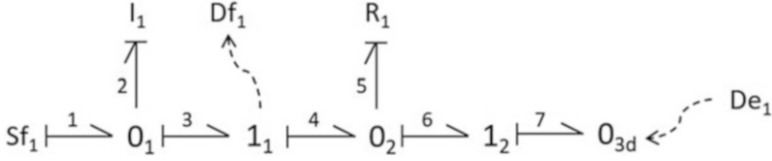
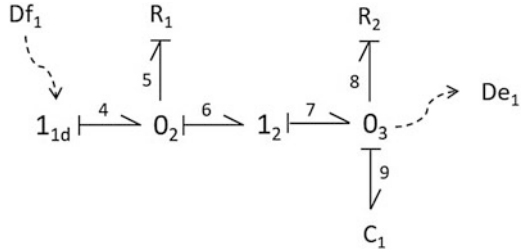


Fig. 6.4 Structural sub Bond Graph model, *st-sBG1*, of the BG model example

Fig. 6.5 Structural sub Bond Graph model, *st-sBG2*, of the BG model example



Definition 4 (Structural Sub Bond Graph (*st-sBG*)) A structural sub Bond Graph, *st-sBG*, derived from $BG = \{E, B\}$, is a connected subgraph $\{E', B'\}$ $| E' = S' \cup M'$ and $S' = S' \cup PSV' \cup J'$ with $S' \subseteq S$, $PSV' \subseteq PSV$, $M' \subseteq M$ and $B' \subseteq E' \times E' \subset B$. $J' = J'_o \cup J'_d$, $J'_o \subseteq J_t$, and J'_d is a set of zero or more degenerated junctions. Additionally if $j_d \in J'_d$ was derived from $j_o \in J_t$ then $j_o \notin J'$.

A *st-sBG* defines a subgraph obtained from a BG because it is made out of some of the constituent elements of BG together with a set of junctions J_o from the original BG. However there is also a potentially empty set of degenerated 1- and 0-junctions, J_d , not contained in the original BG, that will be used to split the BG in terms of sources or measurements and they are used to determine the value of flow/effort variables. If j_d is a degenerated junction derived from an original junction $j_o \in J$, then by definition $j_o \notin J'$.

Degenerated junctions will be a key part in computing PCs from BGs, as it will be shown below. The main idea is that degenerated junctions together with dualized sensors allow to stop the search while computing PCs because they provide the value of a measured effort/flow variable, which will act as input variables for the subgraph defined by the PC. Figure 6.4 shows *st-sBG1*, a *st-sBG* obtained from the BG model example with VCA of Fig. 6.3. *st-sBG1* is derived replacing the O_3 junction in the BG model example by the degenerated junction O_{3d} , deleting bonds 8 and 9, and dualizing sensor De_1 . In a similar way, we can obtain *st-sBG2*, shown in Fig. 6.5, by dualizing sensor Df_1 , creating a new degenerated junction 1_{1d} , and removing bond number 3 and the rest of elements to its left in the original BG.

For diagnosis task, we are only interested in *st-sBGs* with analytical redundancy:

Definition 5 (Redundant Structural Sub Bond Graph (RBG)) A *RBG* is defined as a *st-sBG* whose underlying model has analytical redundancy.

In order to obtain a parsimonious representation of conflicts and diagnosis we define the concept of *minimal RBG*:

Definition 6 (Minimal RBG) A *RBG* derived from a valid BG with a VCA is minimal if \nexists RBG' also derived from BG such that $RBG' \subset RBG$.

Now we have the necessary concepts to define a PC in the BG framework:

Definition 7 (Bond-Graph Possible Conflict (BG-PC)) Given a valid BG with a VCA, a BG-PC is a *minimal RBG* derived from the BG.

The existence of a BG-PC derived from a BG requires that the BG has analytical redundancy: there must exist $d' \in M_{pc}$ such that the VCA of BG-PC allows estimation of d' from dualized sensors and/or sources: d' is the discrepancy node of the BG-PC and it is unique (otherwise the analytical redundancy in BG-PC would not be minimal).

Regarding our BG mode example, it is easy to check that both *st-sBG1* and *st-sBG2* are minimal RBGs. Hence, both *st-sBG1* and *st-sBG2* are BG-PCs.

6.3 Characterizing HBG-PCs

HBG models have no genuine new elements w.r.t. a BG model. Figure 6.6 shows a Hybrid Bond Graph model example, obtained from our former BG example, replacing junction I_2 by a SW 1_{sw} . Only a SW changing its state can modify the underlying set of equations. Consequently, using HBG models instead of BG models makes little difference in the way PCs are defined. Hence, extending the concept for hybrid systems is rather straightforward if every SW is set to ON. All the former definitions extend naturally accepting the presence of SWs that are set to ON. Due to its predominant role, we state the definition for HBG-PC:

Definition 8 (HBG-PC) It is a hybrid *BG-PC* = $\{E', B'\} | E' = St' \cup M'$ and $St' = S' \cup PSV' \cup J'$, derived from a valid HBG, where some elements $J_{sw} \subset J'$ are SWs, and has a global VCA when every SW in J_{sw} is set to ON.

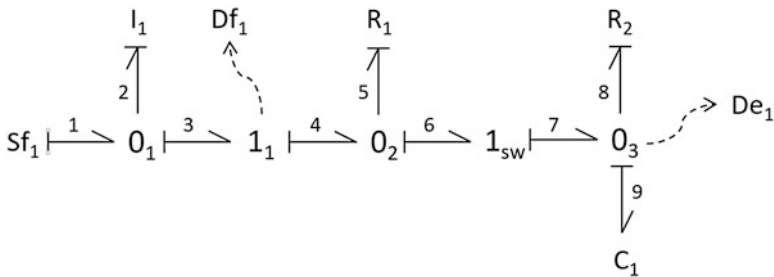


Fig. 6.6 Hybrid Bond Graph model example

In [16, 20] it is discussed how the set of PCs in a HBG when every SW is set to ON provides the smaller set of PCs, and any analytically redundant subsystem will be part of one of these PCs.⁴ The reason is that a change in one SW going from ON to OFF will neither introduce new measurements in the system nor increase the number of state variables. Hence, it cannot be the source of new redundancy. Switching from ON to OFF and vice versa will only connect or disconnect parts of the system.⁵

We now propose to remove the assumption that there is a VCA for the complete system when every SW is set to ON: such configuration might have no valid causal model. This is not surprising. Some systems have multiple structural configurations and several are not compatible among them. These configurations are usually known beforehand because they represent the *limited* set of valid operation modes in the system. But a well-designed system must have at least one VCA for some configuration of SWs set to ON and/or OFF.⁶ If we assume that a SW set to OFF disconnect a part of the system, each one of these valid configurations represent a subsystem where our assumption for BG-PCs holds: it must have a VCA when every SW in that subsystem is set to ON. If we are able to find the maximal subsets of SWs where our assumption holds, we can guarantee that there will be no new genuine and smaller BG-PCs.

Definition 9 (*maxJ_{SW}*) Given a *RBG* and its set of SW, J_{sw} , a maximal subset $maxJ_{SW} \subseteq J_{sw}$, satisfies the following properties:

- *RBG* has a VCA when all the SW in $maxJ_{SW}$ are set to ON
- $\forall sw' \in \{J_{sw} \setminus maxJ_{SW}\}$, *RBG* has no VCA with all the SW in $maxJ_{SW} \cup \{sw'\}$ set to ON

Since each BG-PC in a BG model is related to a sensor, and changing a SW from ON to OFF does not introduce new redundancy, we can search for HBG-PCs in two steps: first, we search for structural redundancy, assuming every SW is ON, without considering causality. We call these new subsystems containing analytical redundancy: Structural HBG-PCs, or SHBG-PCs for short. Second, we check for those maximal SW configurations in each SHBG-PC for VCAs. Each one of these valid configurations is a HBG-PC.

Definition 10 (Structural HBG-PC) It is a hybrid *RBG* such that $\forall maxJ_{SW} \subseteq J_{sw}$, each $RBG' \subseteq RBG$, obtained by means of the following operations, is a HBG-PC (that is, each RBG' generated is *minimal*):

- setting in *RBG* all the SW in $\{J_{sw} \setminus maxJ_{SW}\}$ to OFF
- keeping in RBG' all the SW in $maxJ_{SW}$ to ON

⁴Exceptionally some degenerated subsystems can appear, but they had no interest for diagnosis purposes.

⁵Exceptions to this behavior may happen if there are non-parametric flow or effort paths in the BG from sources to sensors.

⁶Otherwise there would be parts of the system that will be never used, or there is no need for such SW in the model.

And subject to the following conditions:

- $J_{sw} = \bigcup \max J_{sw}$
- Structural HBG-PC is minimal in the sense that \nexists RBG'' derived from the RBG such that $\text{RBG}'' \subseteq \text{RBG}'$ and RBG'' is a HBG-PC.

Each SHBG-PC defines the maximal set of elements in the HBG that can be a part of a HBG-PC related to the discrepancy node in the SHBG-PC (which is the sensor introducing the redundancy).

Searching for the sets $\max J_{sw}$ is a worst case exponential problem. However, the way we compute SHBG-PCs help us with this problem because the SHBG-PC is a subset of the complete system, hence reducing the search space.

For our HBG model example of Fig. 6.6, it is easy to check that we obtain two hybrids *st-sBGs*, essentially the same as in Figs. 6.4 and 6.5, simply replacing the junction I_2 by the SW I_{sw} . Both of them have a VCA with the I_{sw} set to ON. Hence $\max J_{sw}$ is the set $\{I_{sw}\}$ in both cases. As a consequence we obtain two SHBG-PCs.

In this simple HBG model example the concept of SHBG-PC does not pay off, because there is only one SW and there exists a VCA when the I_{sw} is set to ON. Hence, we obtain the same results than using the BG-PC formalism. Therefore, we introduce a motivating example to show the potential of SHBG-PCs.

6.4 SHBG-PCs Motivating Example

To illustrate the SHBG-PCs approach we will use a simple electric circuit that exhibits the major feature of hybrid systems: a pair of physical switches that provide the system with four potentially different working modes. This system is shown in Fig. 6.7a.

This circuit includes two batteries connected in parallel. Although this is a common configuration that poses no problems in real applications, it can create computational difficulties in simulation. When the batteries are modeled as ideal batteries with no internal resistance and both switches are set to ON, the resulting simulation equations have no valid causal assignment, because each battery imposes a different value for the voltage at the input of resistance R_1 .

Figure 6.7b shows the HBG model for our electric system. In that model there are two 1-SWs, I_{sw_1} and I_{sw_2} , both of them set to ON. That HBG model has no causal assignment to the bonds in the graph.

Figure 6.8 shows different configurations of the system when we set to OFF any of these SWs. For the sake of clarity we turn the color to grey in the HBG schematic for those SWs and their related bonds when they are switched to OFF. Each one of them is a BG if we think that a SW set to ON is a regular junction. Configurations in Fig. 6.8b–d have a VCA. Only configuration in Fig. 6.8a has no VCA, corresponding to the two SWs set to ON.

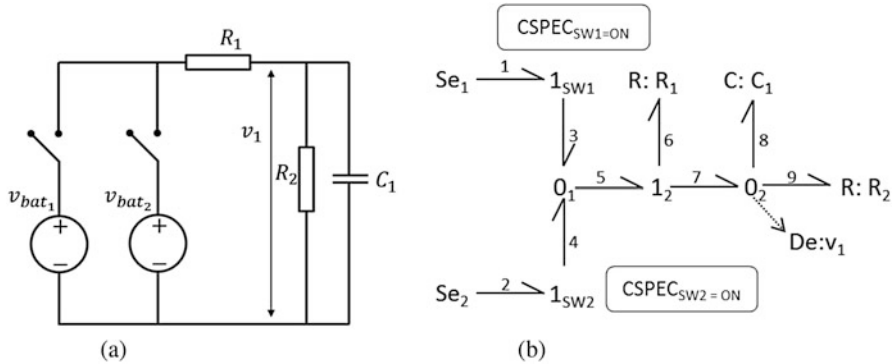


Fig. 6.7 An electric system fed with two power sources. Each power source can be selected using a switch. (a) System schematic. (b) Hybrid Bond-Graph Model

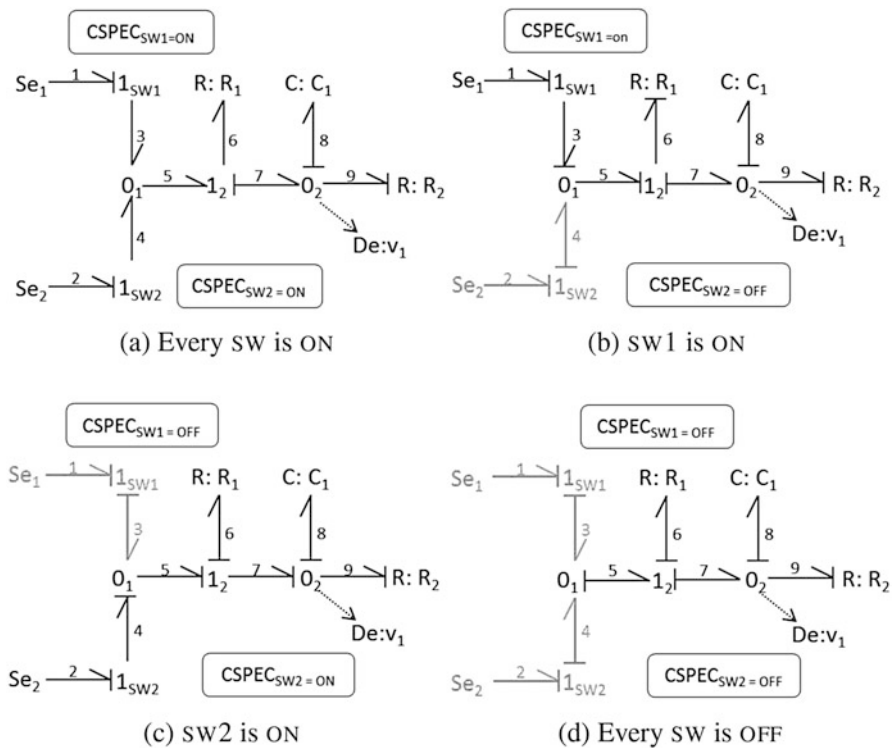


Fig. 6.8 BGs for the running example under the four possible configurations. Configuration (a) has no valid causal assignment. Configuration (d) has a valid causal assignment, but no redundancy

6.5 Computing Structural HBG-PCs

In this section we present the new algorithms to compute the set of SHBG-PCs and we will illustrate their performance on the electrical circuit of Fig. 6.7.

6.5.1 The Algorithms

Algorithm 1 assumes the original HBG model is correct, and that effort/flow sensors will be only connected to 0/1 junctions, respectively. The HBG is modelled as a set of nodes (elements) connected through bonds.

Computing SHBG-PCs from HBGs is straightforward: we must traverse the HBG from sensors to sources and/or other sensors, collecting all its components, except those of the degenerated junctions linked to dualized sensors. There is, however, one exception to this rule, due to the presence of *non-parametric* paths: paths of the HBG that allow propagating flow or effort from a source or sensor independently of passive elements. These paths hence contain only sources, junction elements, and/or sensors. Given that we are looking for minimal redundancy, the passive elements of non-parametric paths should not be included in a SHBG-PC. For that reason, the algorithm first will look for *non-parametric* paths in the HBG. This can be done straightforward, using depth-first search from each source/sensor through paths that contain only non-parametric or degenerated junctions.

It should be noticed that some non-parametric paths could be BG-PCs because they can estimate the value of sensor from other sources or other sensors. These paths can be regarded as degenerated BG-PCs because they can only detect faults in sensors, but they cannot be used to diagnose parametric faults. For that reason, they are not considered in this framework as regular BG-PCs.

Once the algorithm identifies nodes in non-parametric paths in the HBG, then it performs depth-first search for minimal paths from every *sensor* available to source or sensor elements. Each node in the so-called *TentativePath* is required to compute the effort/flow variable in the *sensor*. Once the node is analyzed, it is included in the *TentativeSBGPC*, which would be a SHBG-PC if the search succeeds.

Algorithm 2 extracts in each step a *node* from the *TentativePath* until this path is empty. In that case the search stops and the set of nodes in *TentativeSBGPC* is a new SHBG-PC.

Each element in *TentativePath* is analyzed by means of Algorithm 3. Depending on the type of *node*, different actions are performed. If *node* is a junction, and there is a source or sensor adjacent to *node*, the search will stop. Same happens if the junction belongs to a non-parametric path linked to a sensor or source different from the original discrepancy sensor. Otherwise, we collect any element—including junctions—not previously visited, adjacent to the junction, and store them in the current *TentativeSBGPC*.

Algorithm 1: SHBG-PC ALGORITHM**Input:** Set of nodes in Bond Graph: *nodeSet***Output:** Set of SHBG-PCs

```

1 Mark nodes belonging to Non-Parametric Paths to sources/sensors;
2 for each sensor  $\{De \text{ or } Df\}$  in nodeSet do
3    $TentativeSBGPC := \{sensor\}; TentativeSBGPC.discrepancyNode := sensor;$ 
4    $TentativePath := \{\}; nodeSet := nodeSet - \{sensor\};$ 
5   for any node in nodeSet adjacent to sensor do
6      $TentativePath := TentativePath \cup \{node\};$ 
7    $[TentativeSBGPC.ok] := buildRBG(nodeSet, TentativeSBGPC, TentativePath)$ 
8   if  $ok == 1$  AND TentativeSBGPC is a non parametric BG-PC then
9      $\lfloor$  Insert TentativeSBGPC in SHBG-PCs;

```

Algorithm 2: BUILD RBG**Input:** Set of nodes, *nodeSet*, current RBG, *TentativeSBGPC*, and current path, *TentativePath***Output:** Updated *TentativeSBGPC* and *TentativePath*, and error code, *ok*

```

1 while TentativePath is not empty AND  $ok \neq -1$  do
2   Extract node from TentativePath;
3    $[TentativeSBGPC, TentativePath, ok] :=$ 
4      $analyzeEl(node, TentativeSBGPC, TentativePath)$ 

```

Algorithm 3: ANALYZEEL**Input:** Current *node*, current *TentativeSBGPC*, and current search path *TentativePath***Output:** Updated *TentativeSBGPC* and *TentativePath*, and error code *ok*

```

1  $ok := 0$ 
2 if node is not in TentativeSBGPC then
3   Add node to TentativeSBGPC;
4   if node is a junction linked to sensor/source s AND  $s \notin TentativePath$  then
5      $\lfloor ok := 1; \text{Add } s \text{ to } TentativeSBGPC;$ 
6   else if node is a junction in a Non-Parametric Path to s' AND
7      $s' \neq TentativeSBGPC.DiscrepancyNode$  then
8      $\lfloor ok := 1; \text{Add nodes in } Non\text{-Parametric Path to } s' \text{ to } TentativeSBGPC;$ 
9   else if there is a non-empty subset of nodes adjacent to node then
10     $\lfloor$  Add every 1-Port element, E in subset to TentativeSBGPC
11     $\lfloor$  Add every junction in subset to TentativePath
12  else
13     $\lfloor ok := -1$ 

```

Once we have found a SHBG-PC it is necessary to find the sets $maxJ_{SW}$ to determine if it defines a HBG-PC. As mentioned in the previous section this problem has a worst case exponential complexity. However, finding first the SHBG-PCs using structural information we reduce the complexity of the search, turning a global search into a local search (within the SHBG-PC). Moreover, determining the presence of a HBG-PC only needs to find a maximal set with at least one VCA.

6.5.2 SHBG-PCs Found in the Motivating Example

We will illustrate the performance of these algorithms in our motivating example. The HBG model in Fig. 6.7 is correct and has redundancy (given by effort sensor v_1), but if we run HSCAP for “every SW set to ON” (see the configuration shown in Fig. 6.8a) there is no VCA.

Algorithm 1 is run for the only available sensor v_1 , which will be the only possible discrepancy node. The algorithm searches backwards from the adjacent junction 0_2 , adding adjacent elements C_1 and R_2 to the *TentativePath*. Each element in *TentativePath* is analyzed and included in *TentativeSBGPC*.

Algorithms 2 and 3 proceed analyzing junction 1_2 . This junction contains element R_1 which is added to *TentativePath*. The algorithm stops searching when it reaches 0_1 . Both paths arriving to 0_1 are non-parametric: $\{Se_1, 1_{sw_1}, 0_1\}$, and $\{Se_2, 1_{sw_2}, 0_1\}$. Every element is added to *TentativeSBGPC*. The HBG is correctly defined and both paths finish at two sources, each one of them capable to set a value for the effort in junction 0_1 .

Algorithm 1 has found a potential SHBG-PC that is made up of the whole system. Since the whole structural model has no VCA, we start the search for maximal subsets of SWs set to ON in a top-down manner: we alternatively switch one SW to OFF: first 1_{sw_2} , then 1_{sw_1} . Both configurations have a VCA, hence it is not necessary to search further. Consequently the *TentativeSBGPC* is a SHBG-PC that has two sets of $maxJ_{SW}$: $maxJ_{SW}^1$ and $maxJ_{SW}^2$, with $maxJ_{SW}^1 = \{1_{sw_1}\}$ and $maxJ_{SW}^2 = \{1_{sw_2}\}$. When we configure the SHBG-PC according to those $maxJ_{SW}$, we obtain two minimal hybrid RBGs: RBG_1 and RBG_2 . Both are minimal HBG-PCs too: namely $HBG-PC_1$ and $HBG-PC_2$. They only differ on the effort source (ideal battery) and the SW that is set to ON.

These results are summarized in Table 6.1.

Table 6.1 HBG-PCs found in the electric circuit for the SHBG-PC containing the whole system model

HBG-PC	Sensor	$maxJ_{SW}$	Elements
$HBG-PC_1$	v_1	$\{1_{sw_1}\}$	$\{Se_1, 0_1, 1_1, R_1, 0_2, C_1, R_2\}$
$HBG-PC_2$	v_1	$\{1_{sw_2}\}$	$\{Se_2, 0_1, 1_1, R_1, 0_2, C_1, R_2\}$

These two configurations have VCA and redundancy

Although it was not necessary to try the configuration where both SW were set to OFF because it would not be maximal, looking at Fig. 6.8d it is clear that such configuration has a VCA, but it is not a RBG. Hence, it does not affect our results on the previous paragraph.

In Fig. 6.8 we could see the four configurations for the non-causal model in Fig. 6.7b. $HBG-PC_1$ and $HBG-PC_2$ correspond to Fig. 6.8b and c respectively. The HBG in Fig. 6.8a has no VCA and the HBG in Fig. 6.8d is not a RBG.

Summarizing, the SHBG-PC is the union of $HBG-PC_1$ and $HBG-PC_2$ and it contains the whole system given its two valid $maxJ_{SW}$ sets.

The SHBG-PC framework works searching for *minimal* HBG-PCs related to every measured variable, independent of causality. Those HBG-PCs related to the same discrepancy node made up the SHBG-PC. For each SHBG-PC we must try causality assignments and find out every $maxJ_{SW}$.

We have tested the algorithms in larger systems with more complex HBG models (such as a Reverse Osmosis System), finding the complete set of HBG-PCs that can be found using previous approaches. Due to lack of space we can neither include the description of the systems nor the results. The reader can find a complete description of these tests in [21].

The next section shows how SHBG-PCs can be used to perform fault detection, isolation and identification of both parametric and discrete faults for hybrid systems.

6.6 Common Framework for Discrete and Parametric Faults

In this section we extend the concepts of fault signature matrix and qualitative fault signature matrix to the hybrid case. These extensions allow us to process in a similar way discrete and parametric faults. Afterwards, we illustrate the diagnosis process with a hypothetical scenario, and finally we discuss complexity issues.

6.6.1 Assumptions

Before we describe the architecture, we need to clarify several issues:

- The set of available measurements is fixed (otherwise we need to re-compute the set of redundant subsystems).
- There are no structural faults: System models do not include faults capable of changing the system structure.
- Model parameters can be used to model parametric faults behavior.
- Regarding faults profile, our current proposal works with single fault and abrupt fault assumptions for parametric faults. Abrupt faults appear instantaneously and their magnitudes do not change afterwards (can be modeled as a step function).

- Discrete faults are related to faults in discrete actuators, i.e. commanded or autonomous mode switchings associated to an actuator that does not perform the correct action. We consider four different faulty situations for a SW_i (where SW_i refers to switching junction i), depending on the current state of SW_i —the state can be ON, i.e. 1, or OFF, i.e. 0—and its response to a new command:

1. $SW_i(11)$: SW_i is ON (1) and remains stuck to ON, despite a command OFF (0).
2. $SW_i(00)$: SW_i is OFF (0) and remains stuck to OFF, despite a command ON (1).
3. $SW_i(01)$: Non-commanded switch transition for SW_i from OFF (0) to ON (1).
4. $SW_i(10)$: Non-commanded switch transition for SW_i from ON (1) to OFF (0).

By non-commanded transition we mean either a commanded switch changes its state without a proper command (for instance, a commanded valve suddenly closes) or an autonomous switch changes its state without the required condition (for instance, a complete block of a pipe between two connecting tanks).

- This proposal assumes that the current state of the system is known, and it is characterized by a given configuration of the set of SW in each SHBG-PC: $J' \subset \max J_{SW} \subset J$. A change in the current mode will be related to a commanded or non-commanded/autonomous transition in a SW or due to a discrete fault.

Hence, the continuous system state can be tracked, although full system observability is not required, except for the commanded switches, that must be observable.

- For every SHBG-PC, any of its $\max J_{SW_i}$ defines a $HBG-PC_i$, that is, a minimal RBG_i that includes all the junctions of $\max J_{SW_i}$ set to ON and any other SW set to OFF. Each $HBG-PC_i$ describes a set of equations that can be used for consistency check or residual evaluation. If there is neither state change nor faults, that residual “ideally” should be zero.

The main idea of the integration proposal is to consider always discrete faults as preferred candidates for FDI, because of their potential catastrophic effects. Additionally, since the current state is known, we know the current position for the SWs in the SHBG-PC, thus limiting the search space of potential new faulty states.

6.6.2 Fault Signature Matrices for Fault Isolation

In our framework we use the structural and behavioral information for each SHBG-PC to build three different Fault Signature Matrices. Looking at our electric system example, the structural information is located in Table 6.1.

First, sws in a SHBG-PC will be related to discrete faults. This information is gathered in the Hybrid Fault Signature Matrix, HFSM [16]. In our example, $\{1_{SW_1}, 1_{SW_2}\}$ will be related to the only $SHBG-PC$.

Second, we can characterize the current state in terms of the value of SWs to ON/OFF, representing each configuration by a binary number: SW=ON is translated to 1, while SW=OFF is translated to 0. Hence, $SHBG-PC < 1, 0 >$ and $SHBG-$

Table 6.2 Fault Signature Matrices obtained for *SHBG-PC* in the electric circuit example

sw	<i>SHBG-PC</i>
1_{sw_1}	1
1_{sw_2}	1

(a) HFSM for *SHBG-PC*

$\theta_i \in \Theta$	<i>HBG-PC</i> ₁
R_1^+	0-
C_1^+	-+
R_2^+	0+

(b) RQ-FSM for *SHBG-PC* < 1, 0 >

sw	<i>HBG-PC</i> ₁
$1_{sw_1}(11)$	+
$1_{sw_1}(10)$	-
$1_{sw_2}(00)$	Forbidden configuration
$1_{sw_2}(01)$	No VCA

(c) HQFSM for *SHBG-PC* < 1, 0 >

PC < 0, 1 > are the two valid configurations leading to *HBG-PC*₁ and *HBG-PC*₂, respectively, in Table 6.1. This information will be used to determine potential state transitions due to discrete faults.

Third, *Elements* $\subseteq S \cup PSV \cup J_t$, i.e. subset *Elements* in the SHBG-PC belong to the original HBG. But we are only interested in parametric faults, hence neither sources, sensors nor junctions will be considered as fault candidates. For the electric system example $\Theta = \{R_1, C_1, R_2\}$ is the set of potential parametric faults. This information will be used to build the Reduced Qualitative Fault Signature Matrix, RQ-FSM [16].

We can build three new Fault Signature Matrices using these pieces of information:

1. The *Hybrid Fault Signature Matrix (HFSM)* [16] describes the relation between each SHBG-PC and its sws. If entry $H_{i,j}$ is 1 means that sw_i is in *SHBG-PC*_j, being a potential discrete fault candidate.

For instance, the HFSM for the electric circuit can be seen in Table 6.2a. Since *SHBG-PC* contains the whole electric circuit, it contains every SW. In general, different SHBG-PCs will contain different subsets of SWs.

2. For each state, a subset $sw_j \subseteq \max J_{sw}$ will be set to ON, defining a *HBG-PC*_j, which is a minimal RBG. Hence, the qualitative influence of each fault in the discrepancy node (only available output measurement for *HBG-PC*_j) can be computed from its associated Temporal Causal Graph [4] as described in the TRANSCEND approach, except that using the PCs approach, the information is minimal [15]. That qualitative information is the expected deviation in the residual for each HBG-PC, obtained by comparing the real measurement against the predicted value by its discrepancy node.

Such qualitative information is stored in the *Reduced Qualitative Fault Signature Matrix (RQ-FSM)* [16] for each state. Entries $RQ_{i,j}$ in the RQ-FSM represents the qualitative influence of parametric fault $\theta_i \in \Theta$ for *HBG-PC*_j, if any.

In our electric circuit example, there is only one sensor v_1 , and the RQ-FSM for *SHBG-PC* < 1, 0 >, named *HBG-PC*₁ in Table 6.1, can be seen in Table 6.2b.

3. Discrete faults will usually introduce significant deviations in residual signals. Hence, using a similar approach as in the RQ-FSM, we can build the *Hybrid*

Qualitative Fault Signature Matrix (HQFSM) [16] for the current mode, that represents the expected qualitative deviation in the HBG-PC residual for the current mode for each potential discrete fault configuration.

The reader should notice that under single-fault and known mode assumptions, there is a limited set of discrete faults configurations, that must comply with the definition of discrete faults at the beginning of this section.

For instance, in the electric system example, if we are in the state where $1_{SW_1} = ON$ and $1_{SW_2} = OFF$, that is we have $SHBG-PC < 1, 0 >$ and we are tracking the system with $HBG-PC_1$, then we have the following potential discrete faults: $1_{SW_1}(10)$, $1_{SW_1}(11)$, $1_{SW_2}(01)$, and $1_{SW_2}(00)$. The HQFSM for $HBG-PC_1$ can be seen in Table 6.2c, where the qualitative signatures represent the expected influence of each potential discrete fault in $HBG-PC_1$ residual. In the next subsection we explain how they affect the diagnosis computation.

It seems rather obvious that we need to obtain correct qualitative information from the analysis of residual evolution, because qualitative signatures play a significant role in our reasoning process. Summarizing our methodology, robust methods based on the Z-test are used for symbol generation [4]. The first symbol is derived from the result of fault detection. The second symbol is calculated for the direction of the slope of the residual, also using the Z-test, thus having a robust approach to compute that symbol when the signals are noisy. No implicit computation of the numerical value of the derivative is needed. The window used to calculate the slope is increased until the symbol is successfully generated, or the difference between the current time and the detection time becomes larger than a user-specified limit, at which point the slope is reported as 0, implying that the true slope is either zero or unknown, but very small.

Now we have all the elements required to describe our diagnosis framework.

6.6.3 The Diagnosis Framework

Our model-based diagnosis for both discrete and parametric faults is consistency-based. Hence, we will always use the information in the previously defined fault signature matrices to discard faults whose signatures are not consistent with the hypothesized fault. Only when no fault candidate can be discarded we will start a fault identification process, as described in [22, 23].

We will use Fig. 6.9 to explain the diagnosis process of our proposal. We are representing a hypothetic system with three SHBG-PCs that share several parameters and actuators, graphically depicted by the overlapping HBG-PCs.

Tracking of hybrid systems can be performed using SHBG-PCs. Each new state (due to a new SW configuration) defines one HBG-PC for each SHBG-PC if there is a VCA for the new mode. Such test requires running the HSCAP algorithm on the SHBG-PC in the new mode. If there is a valid HBG-PC it can be used to track the system and to compute a residual [16]. This is also true for the initial state.

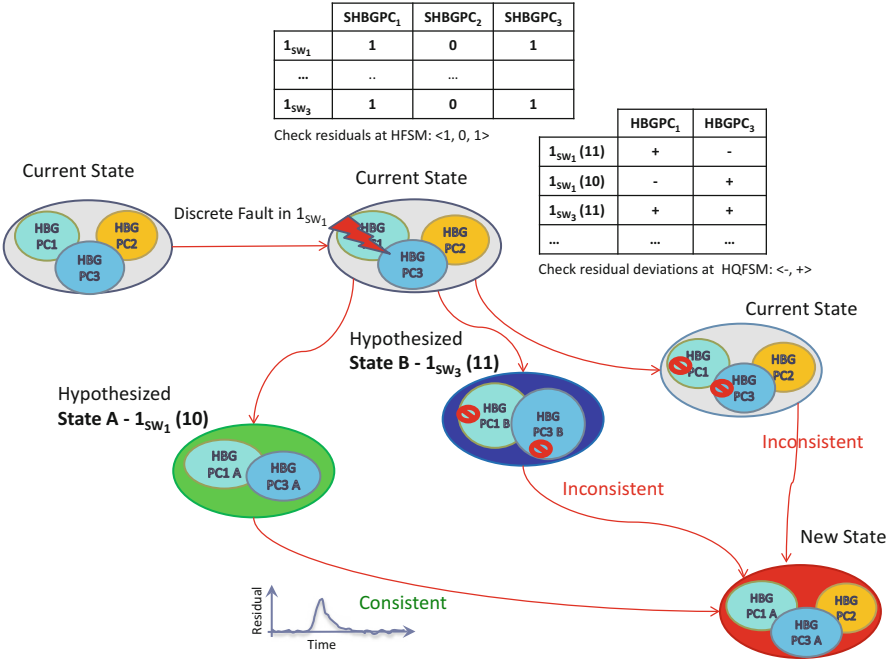


Fig. 6.9 State-tracking for a hypothetical diagnosis scenario

In case of fault appearance, residuals for HBG-PCs sensitive to the fault should be activated. In our diagnosis scenario in Fig. 6.9, the system is in *Current State* and the presence of a discrete fault produces two residual activations for *HBG-PC1* and *HBG-PC3*, while *HBG-PC2* residual remains statistically close to zero.

Since discrete faults will be always preferred candidates, we first look at the HFSM. If there is any SW related to the SHBG-PCs we then look for potential discrete faults in the current state under single-fault assumption. These potential discrete faults define a set of achievable states that are considered *Hypothesized States*, for instance, **State A** and **State B** in Fig. 6.9.

These discrete faults must introduce qualitative deviations in the residuals according to its HQFSM. We check current residual signatures against expected fault signatures in the HQFSM and reject those that are inconsistent (in our example in Fig. 6.9 this corresponds to **State B**). Actually, we do not even generate these states, as they are inconsistent with current observations.

We then create a new *Hypothesized State* for each potential state that is consistent with the current observations. In our example in Fig. 6.9, for the sake of clarity, we have included only one new consistent *Hypothesized State*: **State A**.

Afterwards, we start tracking their residuals during a period σ_t . During that σ_t period, the residuals of one of the consistent *Hypothesized States* should eventually converge to zero, and the discrete fault would be identified.

While that process happened, in parallel, the original HBG-PCs for the *Current State* were still tracking the system to update the set of candidates in case new activations were found.

Finally, we create a *New State* that includes the set of HBG-PCs whose residuals are statistically null. This last *New State* is consistent with current observations and describes the current set of discrete faults under the single fault hypothesis. In our example in Fig. 6.9, the *New State* is made of *HBG-PC2*, not affected by the discrete fault, plus *HBG-PC1 A* and *HBG-PC3 A*. This is the standard scenario for a single discrete fault that is correctly isolated.

Nevertheless, it is possible that none of the residuals of the initially consistent *Hypothesized States* converges to zero. Then all the *Hypothesized States* are tagged as inconsistent, their set of HBG-PCs are deactivated, and the fault is assumed to be parametric for the *Current State*. We then start the parametric fault detection and isolation procedure using the RQ-FSM to obtain an isolation as accurate as possible as described in [22], i.e. those qualitative signatures that do not match observed signatures can be rejected. Only those fault candidates whose qualitative signatures are consistent with the RQ-FSM will be used in the fault identification stage, as described in [22, 23].

In our electric circuit example, there will be only one active HBG-PC for each current or hypothesized state. Hence, the fault isolation will be straightforward. For instance, if the current state is $SHBG-PC < 1, 0 >$, $HBG-PC_1$ is used to track the system. If the residual for $HBG-PC_1$ is activated, we have four potential discrete faults as mentioned above: $1_{SW_1}(10)$, $1_{SW_1}(11)$, $1_{SW_2}(01)$, and $1_{SW_2}(00)$.

- Discrete fault $1_{SW_1}(11)$ has an associated HBG-PC which we can use to track the system. Depending on the qualitative signature of the residual it can be rejected or confirmed.
- Fault $1_{SW_1}(10)$ generates a subsystem where the effort source is no longer connected. The subsystem has no input, but we know the value of the state variable before the fault occurrence and we still can track the evolution of its residual.
- Fault $1_{SW_2}(00)$ should be forbidden due to safety considerations, because it would introduce a non-valid configuration (the original command would try to connect both sources at the same time).
- Finally fault $1_{SW_2}(01)$ will introduce a potential catastrophic configuration too, $SHBG-PC < 1, 1 >$, that has no VCA and cannot be tracked. Model-based diagnosis cannot cope directly with it because system behavior cannot be predicted under this configuration. Hence, some kind of *ad hoc* solution should be called for.

6.6.4 Complexity of the Approach

It should be clear by now that computing HBG-PCs is a system decomposition method that splits the system model into smaller subsystems, thus allowing to reason independently about each subsystem and lowering the inherent complexity

of tracking hybrid systems states. Under multiple fault assumption, the problem is clearly worst case exponential, but in our proposal with single-fault assumption plus the reasoning at the HBG-PC level mitigates this problem. In fact, knowing the current state before fault occurrence, the number of potential new states grows linear with the number of components in each HBG-PC.

Additionally, for each HBG-PC subsystem we must consider different scenarios:

- Mode changes due to nominal transitions, either commanded or autonomous ones, will introduce changes in the model, possibly needing to run a HSCAP-like algorithm, but will not introduce new states to be tracked. That happens just if residuals activate a fault detection.
- In presence of discrete or parametric faults, residual activation will create new state configurations; but due to the high dynamics introduced by discrete faults in the residuals, these kind of faults should be confirmed or rejected very fast. So, the only source for tracking several states in parallel will be due to parametric faults, that will require the qualitative fault isolation and quantitative fault parameter identification stages. Even in those cases we will be working with reduced order fault signature matrices and parameter identification tasks [22]. In case of correct fault identification we will be dealing again with tracking one mode per HBG-PC. Only in the case of new mode changes during the fault isolation and identification stages we could have branching to new states.

6.7 Case Study

In order to fully illustrate our proposal, we will use the hybrid four-tank system shown in Fig. 6.10a. The system has an input flow that can be redirected either to tank 1 or to tank 3 (or both) by using the on/off valves $SW1$ and $SW3$, respectively. Once the liquid in tank 1 reaches given height h , tank 2 starts to fill. A symmetric configuration occurs for tanks 3 and 4.

Figure 6.10b shows the HBG model of the four-tank system in Fig. 6.10a. The system has four 1-SWs: 1_{SW_1} , 1_{SW_2} , 1_{SW_3} and 1_{SW_4} . 1_{SW_1} and 1_{SW_3} are controlled *ON/OFF* transitions, while 1_{SW_2} and 1_{SW_4} are autonomous transitions. Both kind of transitions are represented using a finite state machine. Figure 6.11 shows the automata associated with both kind of transitions for 1_{SW_1} and 1_{SW_2} . Automata for symmetric switches 1_{SW_3} and 1_{SW_4} are equivalent.

Using the algorithms presented in Sect. 6.5 we computed the SHBG-PCs for the HBG model of the four-tank system. Figure 6.12 shows the four SHBG-PCs found, where each one of them estimates one of the measured effort variables (p_1, p_2, p_3, p_4). Each one of these SHBG-PCs defines an equivalent HBG-PC that can be used for diagnosis purposes, because it has a VCA when every SW is ON.

For the set of SHBG-PCs we can compute the HFSM, shown in Table 6.3. Note that faults in 1_{SW_2} and 1_{SW_4} are not considered as potential discrete faults because profiles $1_{SW}(01)$ in both SWs are not physically possible—corresponding to a pipe

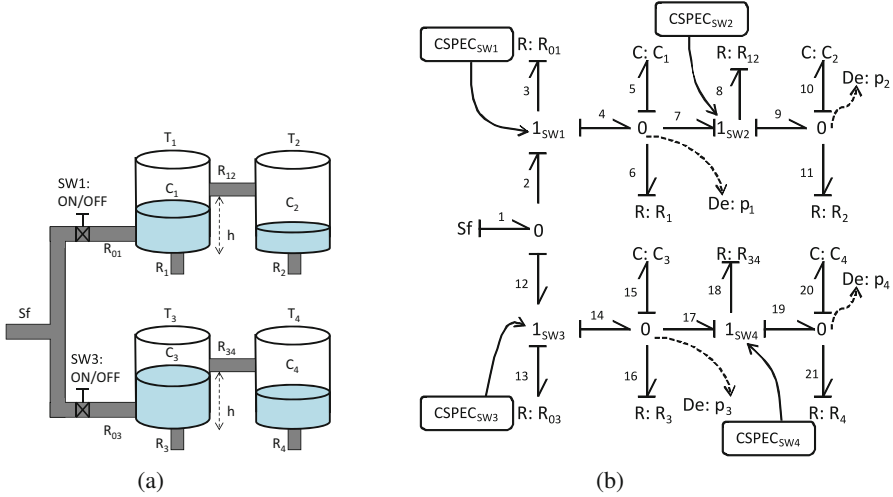
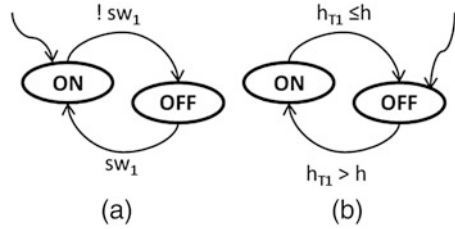


Fig. 6.10 The four-tank hybrid system: schematic and associated Hybrid Bond-Graph model. (a) Schematic. (b) HBG model

Fig. 6.11 Automata associated with the
a) commanded transition for SW_1 , and for the
b) autonomous transition in SW_2 , respectively



suddenly producing flow without any input—, while profiles $1_{SW}(10)$ are equivalent to blockages in the pipes and hence modelled as parametric faults.

For the mode where each SW is ON we can compute the RQ-FSM shown in Table 6.4 and the HQFSM, which is only built for actuators modelled by 1_{SW1} and 1_{SW3} , shown in Table 6.5.

6.7.1 Results for the Case Study

Several scenarios have been tested in simulation to validate this approach: commanded and non-commanded transitions that must not be detected, and fault injections (both discrete and parametric) that must be detected and isolated. We have run several simulations with different mode configurations and faults—varying the size, time of fault occurrence, even introducing faults immediately after the mode change—obtaining satisfactory results in all of them. Due to space limitations, we explain here the results on two of those scenarios. Both simulation experiments were run during 700 s using a sampling period of 1 s; the noise level is set to 5%.

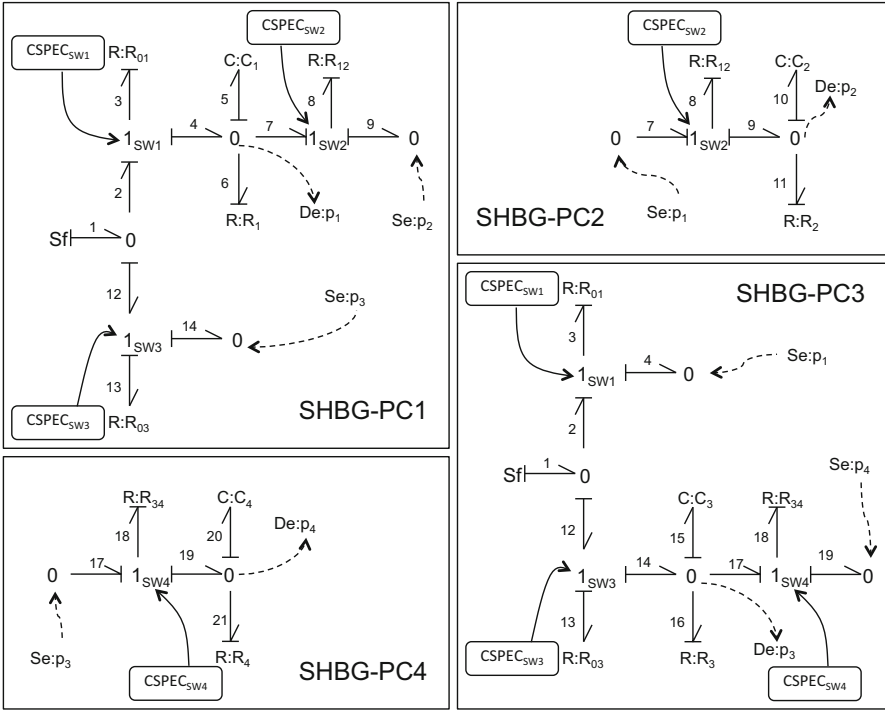


Fig. 6.12 SHBG-PCs found for the four tank system

Table 6.3 Hybrid Fault Signature Matrix (HFSM) showing the relations between switching junctions and each SHBG-PC for the four tank system

Sw-j	SHBG-PC ₁ <1110>	SHBG-PC ₂ <0100>	SHBG-PC ₃ <1011>	SHBG-PC ₄ <0001>
1 _{SW1}	1	0	1	0
1 _{SW3}	1	0	1	0

6.7.1.1 Discrete Fault in SW₁

In this first experiment, the tanks, initially empty, start to fill in. Values for both the commanded switches, SW₁ and SW₃, are set to ON. After 500 s, we introduce a discrete fault in SW₁: the switch moves to OFF without receiving a command to do so (i.e. fault 1_{sw1}(10)). Figure 6.13a shows the evolution of pressure in tanks 1 and 3, and their residuals for HBG-PC₁, and HBG-PC₃ in the time window close to the fault occurrence. Outside of that time window, the residuals are zero. Residuals for tanks 2 and 4 are not affected by this fault. Fault detection occurs at t = 502 s, and looking at Table 6.3, we see that every discrete fault is a potential candidate.

At t = 507 s, we compute the fault signatures: a 0- signature is derived for HBG-PC₁ residual and a 0+ signature is derived for HBG-PC₃ residual. Looking at Table 6.5 and comparing with the actual fault signatures, we conclude that only four discrete faults are consistent with current observations:

Table 6.4 Reduced Qualitative Fault Signature Matrix for mode $1_{SW_1} = On$, $1_{SW_2} = On$, $1_{SW_3} = On$, $1_{SW_4} = On$

Θ	HBG-PC1	HBG-PC2	HBG-PC3	HBG-PC4
C_1^+	-+			
C_2^+		-+		
C_3^+			-+	
C_4^+				-+
R_{01}^+	0-		0+	
R_{03}^+	0+		0-	
R_1^+	0+			
R_2^+		0+		
R_3^+			0+	
R_4^+				0+
R_{12}^+	0-	0-		
R_{34}^+			0+	0-

Table 6.5 Hybrid Qualitative Fault Signature Matrix for the four tank system

Sw-j	HBG-PC1	HBG-PC3
$1_{SW_1}(11)$	+	-
$1_{SW_1}(00)$	-	+
$1_{SW_1}(01)$	+	-
$1_{SW_1}(10)$	-	+
$1_{SW_3}(11)$	-	+
$1_{SW_3}(00)$	+	-
$1_{SW_3}(01)$	-	+
$1_{SW_3}(10)$	+	-

$1_{SW_1}(00)$, $1_{SW_1}(10)$, $1_{SW_3}(11)$, and $1_{SW_3}(01)$. Since we know that in the current state both switches are commanded set to ON, only two discrete faults are possible (discrete faults with the form $1_{SW_i}(1X)$, where X is either 0 or 1):

- A non-commanded transition to OFF in SW_1 : $1_{SW_1}(10)$
- A stuck ON in SW_3 : $1_{SW_3}(11)$

In the next step, our hybrid diagnosis framework creates two different instances of the HBG-PCs in the system, one for each fault candidate. It quickly reassigns causality by running Hybrid SCAP for the mode transitions, and tracks the system for an empirically determined time interval σ_r (in this work, since the system dynamics are quite fast we used $\sigma_r = 20$ s) to isolate the fault. This tracking can be seen in Fig. 6.13b, c, for SW_1 and SW_3 fault candidates, respectively. As can be seen, only the HBG-PC estimation for hypothesized non-commanded transition to OFF in SW_1 is able to track the current behavior (the residuals go to zero). Since the other hypothesized fault cannot recover their residuals, it is rejected as a valid candidate.

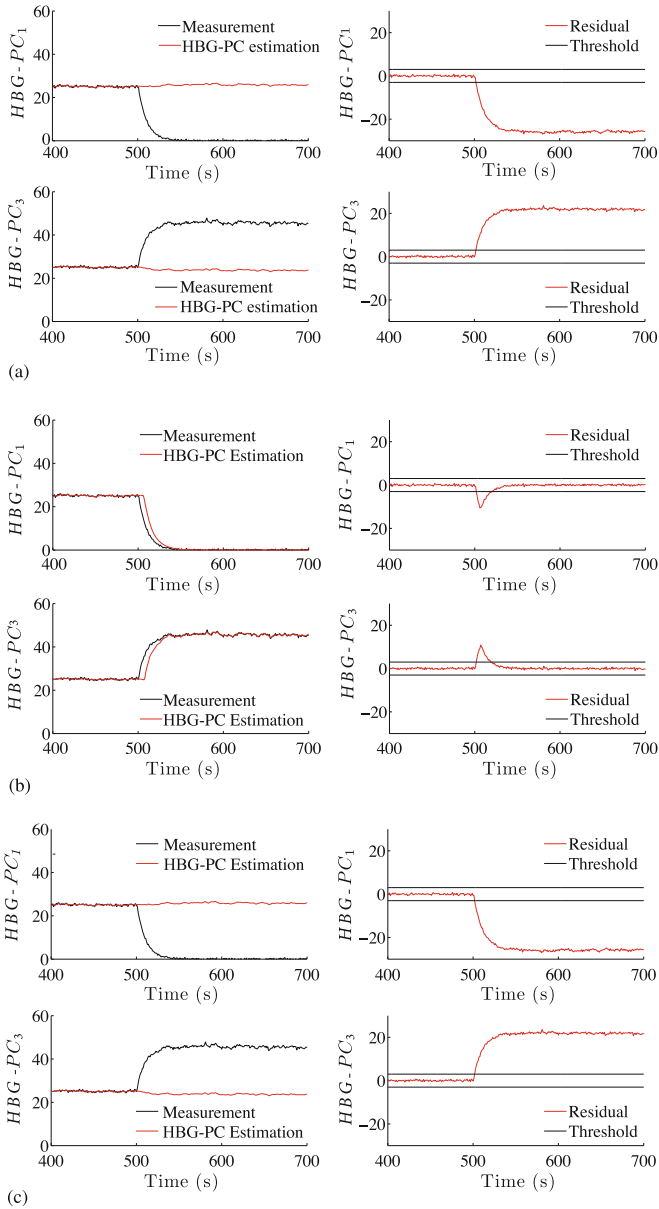


Fig. 6.13 Measurements and estimations of $HBG-PC_1$ and $HBG-PC_3$, and their corresponding residuals, for a non-commanded transition in SW_1 . (a) Non-commanded transition to OFF in SW_1 in the system. HBG-PCs estimation and corresponding residuals. (b) Non-commanded transition to OFF in SW_1 in the system and such change is correctly hypothesized in a new state. (c) Non-commanded transition to OFF in SW_1 in the system, and SW_3 stuck ON is hypothesized in a new state

6.7.1.2 Parametric Fault in R_{01}

The second experiment corresponds to a 20% blockage fault in R_{01} for the same initial configuration. The fault is introduced at time $t = 500$ s, leading to fault detection by $HBG-PC_1$ and $HBG-PC_3$ residuals at $t = 505$ s. Looking at Table 6.3, we see again that there are discrete fault candidates that we have to consider first.

At time $t = 511$ s a 0− signature is derived for $HBG-PC_1$ residual, and a 0+ signature is derived for $HBG-PC_3$ residual. These are the same signatures as in the previous experiment, then the set of candidates is the same.

Similarly to the previous experiment, the hybrid diagnosis framework creates two different instances of the HBG-PCs in the system. Hybrid SCAP reassigns causality for the mode transitions, and tracks the system for the empirically determined time interval of 20 s. However, for this scenario, none of the discrete fault candidates can be confirmed as the true fault in the system (none of the residuals of the hypothesized discrete fault scenarios converged to 0); as a result, the isolation algorithm discards a discrete fault in the system. The next step in the algorithm is to hypothesize a parametric fault. Looking at Table 6.4, we see that the fault signatures obtained for $HBG-PC_1$ and $HBG-PC_3$ only match the fault in R_{01} , thus confirming R_{01} as the true fault in the system, without further calculations.

6.8 Conclusions

The main contribution of this work is the introduction of the Structural HBG-PC, because it allows to find HBG-PCs for hybrid systems without imposing any condition on the global causality of the system model [17], which is a main difference with other approaches relying upon HBG models [18, 19].

This work provides an alternative characterization for HBG-PCs, based on maximal sets of SWs set to ON with a VCA. This definition generalizes the former one [16, 20] when the maximal set is the whole set of SWs. However, if the system does not fulfill this requirement, the concept of SHBG-PC still helps us to deal locally with invalid SW configurations, while containing the whole set of HBG-PCs for every feasible configuration.

A SHBG-PC covers a set of maximal switching junctions. The actual HBG-PCs must be obtained searching within the SHBG-PC, without further search in the global model. Finding maximal sets of SWs has a worst case exponential cost. However, it is not strictly needed to compute these sets, unless we want to characterize offline the families of HBG-PCs for each valid configuration. It is enough to check out that there is one VCA for anyone of the valid configurations.

As in previous works, structural HBG-PCs allow to track hybrid system behavior with only local changes in the model, because a change in a SW is local to the structural model. When a change in the system mode requires a change in a SW from ON to OFF or vice versa, we only need to run a HSCAP-like algorithm within the SHBG-PC to determine a new diagnosis model. This local way of reasoning is

the big difference with respect to HyDe [5], which also allows different diagnosis approaches, while we just use model-based diagnosis.

Recent works working with HBG models also have adopted the need for integration of structural or behavioral information similar to HFSM and HRQSM concepts [24, 25] for both model-based diagnosis and prognostics.

Thanks to HBGs and the concept of switching-junction it is not necessary to enumerate the complete set of modes or to provide different models for all the potential states, compared to ARR-based approaches for hybrid systems [2, 8]. Moreover, SHBG-PCs describe subsystems with minimal redundancy, capable to track current and potentially new system states in parallel, similar to [9], but different to DBGs [11, 19] or previous versions using HBGs and qualitative information [4, 18].

Work required to find and build subsystem models and fault signature matrices can be done off-line, thus reducing complexity and focusing automatically the search for potential new state tracking by analyzing different dynamics in discrete and parametric faults, instead of combining state estimation and tracking [6, 9].

Another difference, with these and similar proposals based on state estimation or pure Discrete Event Systems approaches [26], is that before a fault happens, we assume the current state is known for each SHBG-PC. On one hand, this assumption does not require to specify every potential system mode, due to the presence of HBGs, which allow us to generate dynamically different modes just acting on the values of switching junctions. On the other hand, we require observability of commanded switches. For autonomous switches, they might not be directly measured but they must depend on continuous variables within the HBG-PC, because Possible Conflicts define a strictly overdetermined set of equations, corresponding to an observable subsystem when tracking a continuous process [27]. This fact allows us to compute any unknown variable in the subsystem required for an autonomous transition to be triggered. In those systems that these constraints do not hold, the approach could not be used. We don't require the system to be structurally observable. However, the non-observable part of the system won't be described by any SHBG-PC. Note, also, that under single-fault assumption, if the system is capable to identify the precise discrete/parametric fault, this new configuration can be immediately used as the new current state which will continue tracking the system continuous behavior. We have recently shown that our approach can perform fault identification under discrete mode changes [23].

The main difference between our approach and other works for hybrid systems fault diagnosis using structural model decomposition and qualitative fault isolation [28, 29] is that their models are composed from sets of user-defined components using a compositional modeling approach, instead of using Bond-Graphs. There is also a recent work using Possible Conflicts for hybrid systems diagnosis, but they use different types of mode estimation techniques for mode tracking [30].

This work proposes a centralized approach for model-based diagnosis, but our guess is that can be extended rather straightforward to a centralized version for distributed systems following a similar approach to that proposed in [31] for extending system decomposition methods for distributed systems diagnosis [32].

The main difference with other proposals for hybrid systems diagnosis using HBGs and qualitative fault signature matrices [24, 25] is that SHBG-PCs define subsystems based on minimality alone, while their proposal is based on diagnosability properties and residual sensitivity to faults.

While this work is focused on HBG-PC characterization and computation, together with defining a diagnosis strategy for both discrete and parametric faults, additional research is needed to analyze different notions of diagnosability [33–35] at the SHBG-PC design stage. There would be additional ways to compute PCs, such as merging minimal PCs to obtain non-minimal PCs, looking to improve diagnosability results [23].

Further research is needed to integrate *HBG-PCs* that are completely non-parametric with all their SWs set to ON. In this proposal we cover the case where there are non-parametric paths close to a source. Additional research is needed to extend this approach for distributed systems and multiple faults.

Acknowledgements This work has been supported by Spanish MINECO under DPI2013-45414-R grant. The authors would like to thank Noemi Moya Alonso for her contribution on the preliminary stages of this work, particularly the early HPCs characterization, and Alberto Hernández for the implementation of the SHBG-PCs algorithms. First author wants to thank E. Martinez for the colors and the drive.

References

1. Daigle, M. J. (2008, May). *A qualitative event-based approach to fault diagnosis of hybrid systems*. Ph.D. thesis, Graduate School of Vanderbilt University, Nashville, TN.
2. Cocquempot, V., El Mezayani, T., & Staroswiecki, M. (2004). Fault detection and isolation for hybrid systems using structured parity residuals. In *5th Asian Control Conference*, July 2004 (Vol. 2, pp. 1204–1212).
3. Lunze, J. (2000). Diagnosis of quantised systems by means of timed discrete-event representations. In *Proceedings of the 3rd International Workshop on Hybrid Systems: Computation and Control*, HSCC '00, London, UK (pp. 258–271). Berlin: Springer.
4. Mosterman, P., & Biswas, G. (1999). Diagnosis of continuous valued systems in transient operating regions. *IEEE Transactions on Systems, Man, and Cybernetics—Part A*, 29(6), 554–565.
5. Narasimhan, S., & Brownston, L. (2007). Hyde - A general framework for stochastic and hybrid model-based diagnosis. In *Proceedings of the 18th International Workshop on Principles of Diagnosis, DX07*, Nashville, TN, May 29–31, 2007 (pp. 186–193).
6. Hofbaur, M. W., & Williams, B. C. (2004). Hybrid estimation of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(5), 2178–2191.
7. Benazera, E., & Travé-Massuyès, L. (2009). Set-theoretic estimation of hybrid system configurations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 39, 1277–1291.
8. Bayouhd, M., Travé-Massuyès, L., & Olive, X. (2008). Towards active diagnosis of hybrid systems. In *Proceedings of the 19th International Workshop on Principles of Diagnosis, DX08*, Sept 2008, Blue Mountains, Australia.
9. Rienmuller, T., Hofbaur, M., Travé-Massuyès, L., & Bayouhd, M. (2013). Mode set focused hybrid estimation. *International Journal of Applied Mathematics and Computer Science*, 23(1), 131.

10. Mosterman, P. J., & Biswas, G. (1994). Behavior generation using model switching - A hybrid bond graph modeling technique. In *Society for Computer Simulation* (pp. 177–182). New York: SCS Publishing.
11. Ould Bouamama, B., Biswas, G., Loureiro, R., & Merzouki, R. (2014). Graphical methods for diagnosis of dynamic systems: Review. *Annual Reviews in Control*, 38(2), 199–219.
12. Broenink, J. F. (1999). Introduction to physical systems modelling with Bond Graphs. In *SiE whitebook on simulation methodologies*. University of Twente, Enschede, Netherlands, 1999. Comment Available online at: <http://www.ce.utwente.nl/bnk/papers/BondGraphsV2.pdf>.
13. Karnopp, D. C., Margolis, D. L., & Rosenberg, R. C. (2006). *System Dynamics: Modeling and Simulation of Mechatronic Systems*. New York: John Wiley & Sons, Inc.
14. Pulido, B., & Alonso-González, C. (2004). Possible conflicts: A compilation technique for consistency-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(5), 2192–2206.
15. Bregon, A., Biswas, G., Pulido, B., Alonso-González, C., & Khorasgani, H. (2014). A common framework for compilation techniques applied to diagnosis of linear dynamic systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 44(7), 863–876.
16. Bregon, A., Alonso-Gonzalez, C., Biswas, G., Pulido, B., & Moya, N. (2012). Fault diagnosis in hybrid systems using possible conflicts. In *Proceedings of the IFAC SAFEPROCESS'12, Mexico D.F., Mexico*.
17. Roychoudhury, I., Daigle, M. J., Biswas, G., & Koutsoukos, X. (2010). Efficient simulation of hybrid systems: A hybrid bond graph approach. *SIMULATION: Transactions of the Society for Modeling and Simulation International*, 87, 467–498.
18. Narasimhan, S., & Biswas, G. (2007, May). Model-based diagnosis of hybrid systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 37(3), 348–361.
19. Samantaray, A. K., & Ould Bouamama, B. (2008). *Model-based process supervision: A bond graph approach*. London: Springer.
20. Moya, N. (2013). *Fault Diagnosis of Hybrid Systems with Dynamic Bayesian Networks and Hybrid Possible Conflicts*. PhD thesis, ETSI. Informatica. Universidad de Valladolid.
21. Pulido, B., Alonso-González, C., Bregon, A., & Hernández, A. (2015). Characterizing and computing HBG-PCs for hybrid systems fault diagnosis. In *Conference of the Spanish Association for Artificial Intelligence* (pp. 116–127). Berlin: Springer.
22. Bregon, A., Biswas, G., & Pulido, B. (2012). A decomposition method for nonlinear parameter estimation in TRANSCEND. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 42(3), 751–763.
23. Bregon, A., Alonso, C., & Pulido, B. (2015). Improving fault isolation and identification for hybrid systems with hybrid possible conflicts. In *Proceedings of the XXVI International Workshop on Principles of Diagnosis, DX'15, Paris, France* (pp. 59–66).
24. Prakash, O., & Samantaray, A. K. (2017). *Model-based diagnosis and prognosis of hybrid dynamical systems with dynamically updated parameters*. In *Bond Graphs for Modelling, Control and Fault Diagnosis of Engineering Systems* (pp. 195–232). Switzerland: Springer.
25. Prakash, O., Samantaray, A. K., Bhattacharyya, R. (2017). Model-based diagnosis of multiple faults in hybrid dynamical systems with dynamically updated parameters. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, PP*, 1–20.
26. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., & Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9), 1555–1575.
27. Bregon, A., Alonso-González, C. J., & Pulido, B. (2014). Integration of simulation and state observers for online fault detection of nonlinear continuous systems. *IEEE Transactions on Systems Man and Cybernetics: Systems*, 44(12), 1553–1568.
28. Daigle, M., Bregon, A., & Roychoudhury, I. (2015). A structural model decomposition framework for hybrid systems diagnosis. In *Proceedings of the 26th International Workshop on Principles of Diagnosis, DX'15, Sept 2015, Paris, France*.

29. Bregon, A., Daigle, M., & Roychoudhury, I. (2016). Qualitative fault isolation of hybrid systems: A structural model decomposition-based approach. In *Third European Conference of the PHM Society*, July 2016.
30. Feng, W., Qin, R., Zhang, W., & Zhao, Q. (2016). A possible conflicts based distributed diagnosis method for hybrid system. In *2016 Prognostics and System Health Management Conference (PHM-Chengdu)*, Oct 2016 (pp. 1–6).
31. Bregon, A., Daigle, M., Roychoudhury, I., Biswas, G., Koutsoukos, X., & Pulido, B. (2014). An event-based distributed diagnosis framework using structural model decomposition. *Artificial Intelligence*, *210*, 1–35.
32. Sayed-Mouchaweh, M., & Lughofer, E. (2015). Decentralized fault diagnosis approach without a global model for fault diagnosis of discrete event systems. *International Journal of Control*, *88*(11), 2228–2241.
33. Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M., & Schröder, J. (2006). *Diagnosis and fault-tolerant control* (Vol. 691). Berlin: Springer.
34. Travé-Massuyes, L., Escobet, T., & Olive, X. (2006). Diagnosability analysis based on component-supported analytical redundancy relations. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, *36*(6), 1146–1160.
35. Sayed-Mouchaweh, M. (2014). *Discrete event systems: Diagnosis and diagnosability*. Berlin: Springer Science & Business Media.

Chapter 7

Hybrid System Model Based Fault Diagnosis of Automotive Engines



E. P. Nadeer, S. Mukhopadhyay, and A. Patra

7.1 Introduction

Model based fault diagnosis typically involves three stages: Modelling, residual generation and residual evaluation. Traditionally, due to the ease of development and implementation, model based diagnosis schemes for automotive engines have used Mean Value Models (MVMs) [1–4], where only the mean values of engine variables over each engine cycle are considered. However, the automotive engine is best modeled as a hybrid system with nonlinear continuous dynamics in its various discrete modes, arising due to the forward and backward motions of fluids, the different strokes of the piston cycle, fuel injection, ignition, combustion, etc. We shall refer to such a system as Hybrid Nonlinear System (HNS). The system dynamics or ‘mode’ changes depending on the inputs and/or continuous states themselves. When the system is in one discrete mode, the states of the system follow a continuous dynamics, represented by a set of differential equations describing the state evolution within that particular mode. For improved fault detectability, isolability and identifiability, the continuous dynamic model in each discrete mode captures the within-cycle details of engine variables, including those under faults, much more vividly than the so-called Mean Value Models which average signals over a cycle and suppress fault signatures. Although the use of such Within-

E. P. Nadeer (✉)

KPIT Technologies, Hinjawadi Phase 1 Rd, Phase 1, Hinjewadi Rajiv Gandhi Infotech Park, Hinjawadi, Pimpri-Chinchwad, Pune - 411057, Maharashtra, India

Department of Electrical Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, India

e-mail: epnadeer@gmail.com

S. Mukhopadhyay · A. Patra

Department of Electrical Engineering, Indian Institute of Technology Kharagpur, Kharagpur, West Bengal, India

Cycle Models (WCMs) greatly enhances detection sensitivity and isolability, it also increases the computational costs particularly for on-board applications. As a result, hybrid model-based fault diagnosis schemes have not yet found acceptance in the industry for on-board implementations.

The next stage in diagnosis, namely the residual generation, involves the use of state estimators and observers. Estimation of the engine state is highly challenging for hybrid nonlinear systems (HNS). Extensions of linear system techniques for state estimation are often used in practice. Some examples are nonlinear observers and the nonlinear extensions of the Kalman filter. The most common techniques in use are the Extended Kalman Filter (EKF), the Unscented Kalman Filter (UKF) and their variants. For HNS, the optimal estimation of discrete modes and continuous states would mean that the number of modes to be estimated grows exponentially with each time step. To reduce the complexity, in some techniques, N modes with largest probability are kept and the rest are discarded, and probabilities are renormalized to sum up to unity. The Generalized Pseudo-Bayesian (GPB) approaches and Interacting Multiple Model (IMM) method [5] fall in this category. The diagnosis schemes for an engine typically employ a bank of nonlinear state estimators or observers which require the sets of engine dynamic continuous variable models, one element of the set for each discrete mode, and one set each corresponding either to the nominal or to a faulty system for the various faults under consideration. Note that the number of estimators grows combinatorially with the number of faults and the number of modes of the model.

Particle filters (PF) [6] belong to a class of numerical methods for the solution of the optimal estimation problem in nonlinear non-Gaussian scenarios and come under the generic category of Sequential Monte Carlo algorithms. The Rao-Blackwell Particle Filter (RBPF) [7] is a special kind that reduces the variance in estimates by partitioning the state vector into two sub-vectors so that one sub-vector is updated with sampling whereas the other sub-vector is updated analytically using a suboptimal estimator such as KF, EKF or UKF, that uses prior knowledge of distributions.

Nonlinear observers have been used extensively for the state estimation of one or more subsystems of the engine system. For instance, a nonlinear observer for air mass flow to the cylinder on a turbocharged SI engine was proposed in [8]. Direct redundancy and nonlinear diagnostic observers were used for the diagnosis of the air intake system of an SI engine in [9]. In [10], the cylinder pressure and combustion heat release were estimated for SI engine diagnostic purposes from the engine speed measurement using a nonlinear sliding mode observer. Discrete nonlinear observers with convergence guarantee based on Lyapunov analysis and Linear Matrix Inequality techniques were developed in [11, 12] for cycle-by-cycle estimation of the in-cylinder air fraction in diesel engines, which can be used for combustion control. A reduced order observer was employed in [13] for estimation of exhaust manifold pressure and turbocharger speed in turbocharged gasoline engines. An adaptive extended state observer was used in air-fuel ratio control of gasoline engines in [14]. Sliding mode estimators were used in engine parameter estimation in [15, 16]. In [17, 18], within cycle estimation of engine variables based

on a hybrid automata model of the engine system from air intake to exhaust was carried out and its effectiveness in fault diagnosis was demonstrated. However, note that none of these works have reported a solution for estimating the state of the full SI Engine, but rather only that of subsystem.

As for residual evaluation, techniques such as hypothesis testing [1, 2, 4], fixed [3] and adaptive thresholding [19], Generalized Likelihood Ratio Test (GLRT) [20], Dempster-Shafer Theory (DST) [21], Bayesian Networks [22], and Artificial Neural Networks [23, 24] have been used in conjunction with one or more of the modeling and estimation techniques. However, typically these evaluation techniques only demonstrate a two-class classification problem of fault detection rather than a multiple class problem of fault isolation.

In this chapter, we present a fault diagnosis scheme for a complete SI Gasoline Engine, which requires only one instance of a nominal EKF estimator, followed by a residual prediction stage, and a fault detection and isolation stage. The special features are:

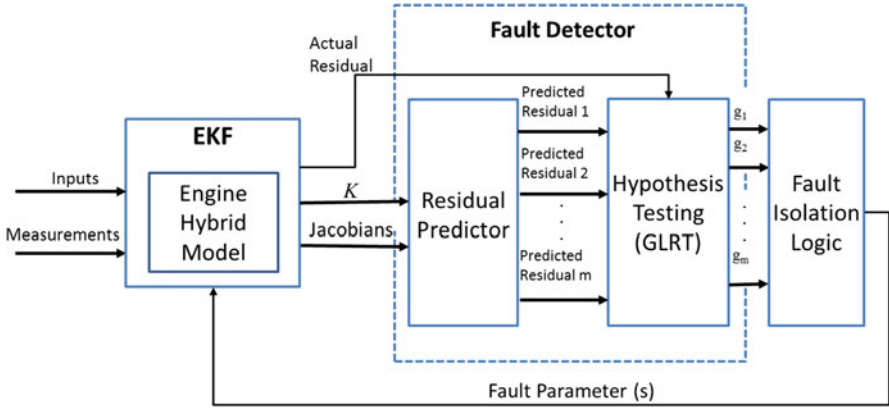
1. Use of a WCM as opposed to MVM for modeling. This enhances fault sensitivity.
2. Development of a hybrid state space model of the entire engine system from air intake to exhaust, including EGR (Exhaust Gas Recirculation), as contrasted with individual engine component models dealt with in most works in the literature.
3. Use of a single EKF estimator for fault detection, that uses an HNS corresponding to the normal system, with approximate adaptive estimation of process and measurement noise covariance matrices, as opposed to a bank of estimators, with accuracy comparable to computationally more expensive nonlinear estimators such as PF. This achieves significant reduction of complexity for real time implementation.
4. A fault isolation scheme involving EKF residual prediction under hypothesis for various faults. The residual prediction process reuses the Jacobian matrices derived for EKF, thereby saving computational costs. This is followed by Generalized Likelihood Ratio Test based isolation.

Fig. 7.1 shows the fault diagnosis scheme.

The rest of the chapter is organized as follows: In Sect. 7.2, the HNS model is developed. Section 7.3 presents the adaptive EKF. In Sect. 7.4, the residual prediction based fault detection, isolation and identification strategies are described. Section 7.5 presents the simulation results for the modeling, estimation and fault diagnosis schemes. Finally, conclusions are drawn and future directions are laid out in Sect. 7.6.

7.2 HNS Modeling of an SI Engine

A naturally aspirated SI gasoline engine has the basic subsystems and components shown in Fig. 7.2. The equations we develop here are from standard models available in duly mentioned references, the novelty being only in the state space formulation.



K - Kalman gain matrix; g_i - Detection function for i^{th} fault

Fig. 7.1 Hybrid model based fault diagnosis scheme using residual prediction

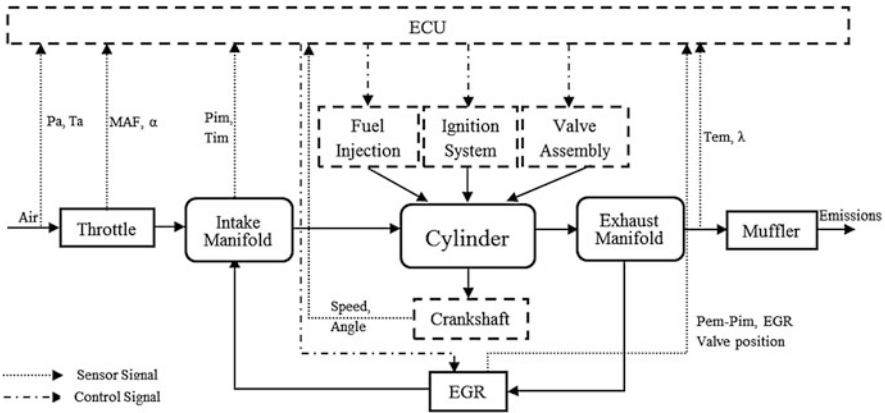


Fig. 7.2 Block diagram of the SI engine system

A full 4-stroke engine cycle corresponds to a crank shaft rotation of $0-720^\circ$, approximately 180° each for intake, compression, expansion and exhaust strokes. The WCM equations essentially describe two things: the continuous time dynamics of the system state variables of interest—such as pressure, temperature and mass flow rates—and the jump conditions, or events, which cause transition between different modes of the hybrid system. These transitions between modes could be triggered either by control actions or by the instantaneous state variable values themselves (e.g., sub-sonic or sonic, positive or negative flow conditions), or by the different strokes of the piston (upwards or downwards), etc.

Table 7.1 Variable and subscript notations

Variables		Subscripts	
P	Pressure	in	Variable in to control volume/flow element
V	Volume	out	Variable out of control volume/flow element
m	Mass	th	Throttle
R	Specific gas constant	im	Intake manifold
T	Temperature	cyl	Cylinder
\dot{m}	Mass flow rate	em	Exhaust manifold
C_p	Specific heat at constant pressure	egr	Exhaust gas recirculation
C_v	Specific heat at constant volume	muf	Muffler
γ	Ratio of specific heats	$i2c$	Intake manifold to cylinder
ω	Engine speed	$c2e$	Cylinder to exhaust manifold
θ	Crank angle	a	Air
H	Enthalpy	b	Burnt gases
Q	Internal energy	f	Fuel
U	Heat energy	cool	Cooling (temperature)
W	Work	amb	Ambient

A summary of main engine variable notations and subscripts used is given in Table 7.1. Each variable on the left column appears with one or more of the subscripts from the right column in the model equations.

Let all the gases in the engine obey the ideal gas law, i.e.,

$$PV = mRT \quad (7.1)$$

R in above expression is the specific gas constant and mass m is in kg . Note that all variables are functions of time t , although it is omitted for notational convenience.

Ignoring the changes in kinetic and potential energy in the flow, the transient mass and energy balance equations involving the state variables can be written in terms of the rates of mass m , the enthalpy H , the internal energy U , the heat energy added to the system Q , and shaft work done on the system W , as [25]:

$$\begin{aligned} \dot{m} &= \sum_i \dot{m}_{i,in} - \sum_i \dot{m}_{i,out} \\ \dot{U} &= \dot{H}_{in} - \dot{H}_{out} + \dot{Q} + \dot{W} \end{aligned} \quad (7.2)$$

where the subscripts in and out denote the variables moving in and out, respectively, of the control volume, often termed as reservoir. The index i stands for different gas species.

The flow equation for the valves can be written as [26]:

$$\dot{m} = C_d A \frac{P_{in}}{\sqrt{R_{in} T_{in}}} \psi \left(\frac{P_{in}}{P_{out}} \right) \quad (7.3)$$

where

$$\psi \left(\frac{P_{in}}{P_{out}} \right) = \begin{cases} \sqrt{\gamma_{in} \left[\frac{2}{\gamma_{in} + 1} \right]^{\frac{\gamma_{in} + 1}{\gamma_{in} - 1}}} & , P_{out} < P_{cr} \\ \left(\frac{P_{out}}{P_{in}} \right)^{\frac{1}{\gamma_{in}}} \sqrt{\frac{2\gamma_{in}}{\gamma_{in} - 1} \left[1 - \left(\frac{P_{out}}{P_{in}} \right)^{\frac{\gamma_{in} - 1}{\gamma_{in}}} \right]} & , P_{out} \geq P_{cr} \end{cases}$$

and $P_{cr} = 2/(\gamma_{in} + 1)^{\frac{\gamma_{in}}{\gamma_{in} - 1}} P_{in}$.

Note that in the above expression the subscripts *in* and *out* have been used assuming a forward flow, with P_{in} being upstream and P_{out} downstream pressures. They will be interchanged for reverse flow with a negative sign attached to the equation. Also, γ_{in} is the ratio of specific heats for the upstream gas species.

At each reservoir, the specific gas constant and specific heats can be expressed as:

$$R = \frac{\sum m_i R_i}{m}, C_p = \frac{\sum m_i C_{p,i}}{m}, C_v = \frac{\sum m_i C_{v,i}}{m}, \quad \gamma = C_p / C_v \quad (7.4)$$

where $i = a, b, f$ (for air, burnt gases and fuel), and,

$$m = m_a + m_b + m_f \quad (7.5)$$

The time derivative of internal energy can be expressed in terms of masses, specific heats and temperature at the reservoir as:

$$\dot{U} = m C_v \dot{T} + (m \dot{C}_v + \dot{m} C_v) T = m C_v \dot{T} + \left(\sum_i \dot{m}_i C_{v,i} \right) T \quad (7.6)$$

Substituting Eq. (7.6) in Eq. (7.2), the time derivative of temperature at each reservoir could be obtained as:

$$\dot{T} = \frac{1}{m C_v} \left(\sum_i \dot{m}_{i,in} C_{p_i,in} T_{in} - \sum_i \dot{m}_{i,out} C_{p_i,out} T_{out} + \dot{Q} + \dot{W} - \sum_i \dot{m}_i C_{v,i} T \right) \quad (7.7)$$

The net flow rates of enthalpy into and out of a reservoir are captured by the first two terms in parenthesis above. Index i stands for individual species of air, burnt gases, and fuel entering and leaving the reservoirs.

7.2.1 Model Equations

We choose a minimal state vector as:

$$x = [m_{im,i}, T_{im}, m_{cyl,i}, T_{cyl,i}, m_{em,i}, T_{em}]^T \quad (7.8)$$

where $i = a, b, f$, corresponding to air, burnt gases and fuel, respectively. $m_{cyl,i}$ and $T_{cyl,i}$ are vectors with elements denoting the mass and temperature for individual cylinders. By this choice of state, it is possible to express the nonlinear engine dynamical equations in the standard form $\dot{x} = f(x, u, t)$ where u is the input vector.

The model inputs are: Throttle position, Fuel control signal, EGR control signal, speed and crank angle. The engine speed and crank-angle position signals, which our model assumes as inputs, are usually not available as direct measurements, and hence need to be extracted from the crank-shaft position sensor signal.

The mass and energy balance in Eq. (7.2) can now be applied at each reservoir, the intake manifold, the cylinder and the exhaust manifold to obtain the state variables. The mass (of air, burnt gases and fuel) variables are obtained from mass balance and the temperature variables are obtained from energy balance in Eq. (7.7).

The mass balance at the intake manifold (IM) is given by:

$$\dot{m}_{im} = \dot{m}_{th} + \dot{m}_{egr} - \dot{m}_{i2c} \quad (7.9)$$

where \dot{m}_{th} and \dot{m}_{egr} stand for the mass flow rates at throttle and EGR, respectively, and \dot{m}_{i2c} is the flow rate of gas entering the cylinder from IM. These can be obtained at every time instant by using Eq. (7.3). Replacing the pressures P_{in} and P_{out} in Eq. (7.3) by Eq. (7.1), and the ratio of specific heats γ_{in} from Eq. (7.4), it can be seen that all the terms in Eq. (7.10) are expressed purely in terms of the states in Eq. (7.9). The individual mass flow rates \dot{m}_i for air, burnt gases and fuel can be obtained from the above relation by multiplying each of these flow rates by the individual fractions from the previous time instant.

Substituting for terms in Eq. (7.7), the enthalpy balance gives

$$\dot{T}_{im} = \frac{1}{m_{im} C_{v,im}} \left(\begin{array}{l} \dot{m}_{th} (\sigma_{\dot{m}_{th}} C_{p_a} T_a + (1 - \sigma_{\dot{m}_{th}}) C_{p_{im}} T_{im}) \\ + \dot{m}_{egr} (\sigma_{\dot{m}_{egr}} C_{p_{em}} T_{em} + (1 - \sigma_{\dot{m}_{egr}}) C_{p_{im}} T_{im}) \\ - \sum_{i=1}^N (\dot{m}_{i2c,i} (\sigma_{\dot{m}_{i2c,i}} C_{p_{im}} T_{im} + (1 - \sigma_{\dot{m}_{i2c,i}}) C_{p_{cyl,i}} T_{cyl,i})) \\ - h_{c,im} A_{c,im} (T_{im} - T_{cool,im}) - (\sum \dot{m}_{im,i} C_{v_{im,i}}) T_{im} \end{array} \right) \quad (7.10)$$

where $\sigma_{\dot{m}} = \left(\frac{1 + \text{sgn}[\dot{m}]}{2} \right)$, $\text{sgn}(\cdot)$ denoting the signum function and N is the number of cylinders. The heat energy lost due to convection is also included in the above equation with $h_{c,im}$ being the heat transfer coefficient, $A_{c,im}$ the effective area of heat transfer and $T_{cool,im}$ the cooling temperature of the IM. Similar expressions hold for the exhaust manifold:

$$\dot{m}_{em} = \dot{m}_{c2e} - \dot{m}_{egr} - \dot{m}_{muf} \quad (7.11)$$

where \dot{m}_{c2e} is the cylinder-to-exhaust flow rate and \dot{m}_{muf} is the muffler flow rate.

$$\dot{T}_{em} = \frac{1}{m_{em} C_{v,em}} \begin{pmatrix} \sum_{i=1}^N (\dot{m}_{c2e,i} (\sigma_{\dot{m}_{c2e,i}} C_{p_{cyl,i}} T_{cyl,i} + (1 - \sigma_{\dot{m}_{c2e,i}}) C_{p_{em}} T_{em})) \\ - \dot{m}_{muf} (\sigma_{\dot{m}_{muf}} C_{p_{em}} T_{em} + (1 - \sigma_{\dot{m}_{muf}}) C_{p_a} T_a) \\ - \dot{m}_{egr} (\sigma_{\dot{m}_{egr}} C_{p_{em}} T_{em} + (1 - \sigma_{\dot{m}_{egr}}) C_{p_{im}} T_{im}) \\ - h_{c,em} A_{c,em} (T_{em} - T_{cool,em}) - (\sum \dot{m}_{em,i} C_{v_{em,i}}) T_{em} \end{pmatrix} \quad (7.12)$$

The mass balance equation for individual cylinders could be written as

$$\dot{m}_{cyl} = \dot{m}_{i2c} - \dot{m}_{c2e} + \dot{m}_f, \quad (7.13)$$

where \dot{m}_f is the fuel input flow rate. During combustion, the total mass flow rate is zero, but the air and fuel get converted to burnt gases, changing individual component rates. These rates can be calculated for the i th cylinder as

$$\begin{aligned} \dot{m}_{cyl,a}^{(i)} &= \dot{m}_{i2c}^{(i)} \left(\sigma_{\dot{m}_{i2c}}^{(i)} \frac{m_{im,a}}{m_{im}} + \left(1 - \sigma_{\dot{m}_{i2c}}^{(i)} \right) \frac{m^{(i)}_{cyl,a}}{m^{(i)}_{cyl}} \right) - \\ &\quad \dot{m}_{c2e}^{(i)} \left(\sigma_{\dot{m}_{c2e}}^{(i)} \frac{m^{(i)}_{cyl,a}}{m^{(i)}_{cyl}} + \left(1 - \sigma_{\dot{m}_{c2e}}^{(i)} \right) \frac{m_{em,a}}{m_{em}} \right) - \frac{m_{fb}^{(i)} \lambda_{af} \omega}{\Delta \theta} \\ \dot{m}_{cyl,f}^{(i)} &= \dot{m}_{i2c}^{(i)} \left(\sigma_{\dot{m}_{i2c}}^{(i)} \frac{m_{im,f}}{m_{im}} + \left(1 - \sigma_{\dot{m}_{i2c}}^{(i)} \right) \frac{m^{(i)}_{cyl,f}}{m^{(i)}_{cyl}} \right) - \\ &\quad \dot{m}_{c2e}^{(i)} \left(\sigma_{\dot{m}_{c2e}}^{(i)} \frac{m^{(i)}_{cyl,f}}{m^{(i)}_{cyl}} + \left(1 - \sigma_{\dot{m}_{c2e}}^{(i)} \right) \frac{m_{em,f}}{m_{em}} \right) - \frac{m_{fb}^{(i)} \omega}{\Delta \theta} + \dot{m}_f^{(i)} \\ \dot{m}_{cyl,b}^{(i)} &= \dot{m}_{cyl}^{(i)} - \dot{m}_{cyl,a}^{(i)} - \dot{m}_{cyl,f}^{(i)} \end{aligned} \quad (7.14)$$

where $m_{fb}^{(i)}$ is the mass of fuel accumulated in cylinder i before combustion, ω is the engine angular speed, λ_{af} is the stoichiometric air-fuel ratio and $\Delta \theta$ is the combustion duration angle. The fuel flow rate into the cylinder, $\dot{m}_f^{(i)}$, could be calculated from the fuel injector pulse width signal coming from the ECU. The terms $\frac{m_{fb}^{(i)} \lambda_{af} \omega}{\Delta \theta}$ and $\frac{m_{fb}^{(i)} \omega}{\Delta \theta}$ in the above expression appear only for the combustion duration.

In addition to the terms in enthalpy balance for IM and EM, the enthalpy balance for cylinder i should include the rate of heat energy added during combustion (\dot{Q}_{comb}), which is approximated by the Wiebe function [27], and lost due to convection and radiation ($\dot{Q}_{heatloss}$) [28], given by,

$$\begin{aligned} \dot{Q}_{comb}^{(i)} &= \eta_c m_{fb}^{(i)} Q_{lhv} \omega \frac{dx_b^{(i)}}{d\theta} \\ &= \frac{\eta_c m_{fb}^{(i)} Q_{lhv} \omega n a}{\Delta \theta} \left(\frac{\theta - \theta_{soc}^{(i)}}{\Delta \theta} \right)^{n-1} \exp \left[-a \left(\frac{\theta - \theta_{soc}^{(i)}}{\Delta \theta} \right)^n \right] \\ \dot{Q}_{heatloss}^{(i)} &= h_c A_c^{(i)} (T_{cyl}^{(i)} - T_{cool}^{(i)}) + \varepsilon \sigma \left((T_{cyl}^{(i)})^4 - (T_{cool}^{(i)})^4 \right) \end{aligned} \quad (7.15)$$

where η_c is the combustion efficiency, Q_{lhv} the lower heating value of fuel, a , n the Wiebe parameters, x_b the burnt mass fraction, θ_{soc} the start of combustion angle, h_c the coefficient of convective heat transfer, A_c the effective area of convective heat transfer, ε is the emissivity of cylinder block material and σ the Stefan-Boltzmann constant.

At the cylinder, the enthalpy balance should also include the piston work, which for the i th cylinder is given by $P_{cyl}^{(i)} \dot{V}_{cyl}^{(i)}$. Thus for the i th cylinder, the expression for cylinder temperature derivative is

$$\dot{T}_{cyl}^{(i)} = \frac{1}{m_{cyl}^{(i)} C v_{cyl}^{(i)}} \left(\begin{aligned} &\dot{Q}_{comb}^{(i)} - \dot{Q}_{heatloss}^{(i)} + \dot{H}_{in}^{(i)} - \dot{H}_{out}^{(i)} \\ &- T_{cyl}^{(i)} \times \left(\sum \dot{m}_{cyl,j}^{(i)} C v_{cyl,j}^{(i)} \right) - P_{cyl}^{(i)} \dot{V}_{cyl}^{(i)} \end{aligned} \right), \quad (7.16)$$

where the index j has been used for individual gas species. As before, all the terms on the RHS above should be expressed in terms of inputs and state. To express the time derivative of the cylinder volume $\dot{V}_{cyl}^{(i)}$ in terms of crank angle and speed input, we consider the expression for cylinder volume [25], namely,

$$V_{cyl} = \frac{V_d}{r_c - 1} + \frac{V_d}{2} \left(\frac{l}{r} + 1 - \cos \theta + \sqrt{\frac{l^2}{r^2} - \sin^2 \theta} \right) \quad (7.17)$$

where V_d is the displacement volume, r_c is the compression ratio, l is the connecting rod length and r is the crank radius. Differentiating the above expression w.r.t. time, we get

$$\dot{V}_{cyl} = \frac{dV_{cyl}}{d\theta} \frac{d\theta}{dt} = \frac{V_d}{2} \left(\sin \theta + \frac{\sin 2\theta}{2\sqrt{\frac{l^2}{r^2} - \sin^2 \theta}} \right) \omega \quad (7.18)$$

where ω is the angular speed.

Equations (7.10)–(7.18) constitute the state space model, considering the speed ω and crank-angle θ as inputs. If, however, the speed and crank-angle are considered as states, and load torque as an input, an additional torque modelling is necessary.

As mentioned earlier, the developed model is a hybrid one, with many switching transitions happening depending on the input and state conditions.

7.2.2 Model Parameters and Tuning

The parameters used in model development can be classified into three categories: general parameters, environmental parameters and engine-specific parameters. General parameters include properties of gasoline, which was assumed to be the fuel for the developed model, such as the lower heating value, and properties of gases such as specific heats of air, burnt gases and fuel. These could be assumed to be the

same over all gasoline engines for which the model is to be used. Environmental parameters, which include the atmospheric temperature and pressure, may need to be changed according to the outdoor conditions either based on prior knowledge of environment or sensor measurements. Engine-specific parameters include all parameters related to engine geometry, fuel injection timings, valve lifts, valve timings and additional engine specific parameters such as the bypass valve area.

The engine specific parameters could either be plugged in by the user depending on the particular vehicle model and type or be chosen from a list of parameter sets available for an array of engines. If some of the engine parameters are not known beforehand, they could either be found by experimentation or could be learnt by online estimation during the model initialization phase assuming a non-faulty engine. Minor deviations in parameter values are acceptable provided they do not manifest as faults, since the estimator in which the model would be used could correct these deviations.

7.2.3 Fault Modelling

To achieve fault detection, isolation and identification, additional fault modelling needs to be carried out, thereby increasing the effective number of parameters in the model. The fault parameters should be able to capture a wide range of faults that could happen in the engine system. Table 7.2 shows how some of the engine faults could be captured in the present modelling scheme.

The intake and exhaust manifold leaks could be modelled by additional areas in respective flow elements. The valve faults in cylinder could also be modelled in the same manner. Note that the valve faults could also manifest as misfire. For the case of manifold fuel injection, the misfire and fuel injector faults could be modelled separately, since the former affects only a particular cylinder while the latter affects all.

7.3 Engine State Estimation

Noting that the engine model is time-invariant, the general form of state and measurement equations for all the estimators that we are about to implement could be formulated as:

$$\begin{aligned}
 \dot{x} &= f(x, u, w) \\
 y_k &= h(x_k, u_k, v_k) \\
 w(t) &\sim (0, Q) \\
 v_k &\sim (0, R_k)
 \end{aligned}
 \tag{7.19}$$

where x is the state vector, y is the measurement vector, u is the input, w is the process noise, v is the measurement noise, Q is the process noise covariance, R is the

Table 7.2 Fault modelling for various faults in the engine system

	Faults	Modelling
Process and actuator faults	Intake manifold leak	Additional unknown area in the throttle
	Exhaust manifold leak	Additional unknown area in the muffler
	Misfire, fuel injector fault	Multiplicative factor on fuel injection rate to individual cylinders
	Intake valve faults	Multiplicative/additive factor on intake valve area
	Exhaust valve faults	Multiplicative/additive factor on exhaust valve area
Sensor faults	Mass air flow (MAF) sensor bias	Additive term in measurement equation
	MAF sensor calibration fault	Multiplicative term in measurement equation
	Manifold pressure and temperature (TMAP) sensor bias	Additive term in measurement equation
	TMAP sensor calibration fault	Multiplicative term in measurement equation
	Exhaust pressure sensor bias	Additive term in measurement equation
	Exhaust pressure sensor calibration fault	Multiplicative term in measurement equation

measurement noise covariance and k is the sampling index. $f(\cdot)$ and $h(\cdot)$ indicate the state transition and measurement functions, respectively. For simplicity, we further assume that the noises w and v are additive and their elements are independent of each other. The sampling rates for the sensors may be different for each sensor and the integration time step for the model could be different from the sampling rates for sensors.

Not all the measurements from the engine system qualify as measurements in the measurement model, since the engine model assumes certain measurements as inputs. Thus the speed and the crank angle signals, extracted from the crank position sensor, are considered as inputs to the model. The throttle position measurement, from which the throttle area is calculated, is also an input. For simulation, the measurements that are assumed to be available to the estimator are: the mass air flow rate (MAF), the intake manifold temperature (T_{im}), the intake manifold pressure (P_{im}), the exhaust manifold temperature (T_{em}) and the exhaust manifold pressure (P_{em}).

It should be noted that, of the various switching modes that the engine system undergoes in one switching cycle, in some modes there may be partial loss of observability of some state elements from the measured variables. For example, during the combustion phase of the engine, since both the engine valves are

closed, some engine variables are isolated from sensors, and therefore would not be observable during this phase. An analysis of observability for such a nonlinear switching system is complex and is not attempted here.

Despite the above caveats, due to the fact that the system visits observable modes during every cycle, during observable modes the estimator corrects the error built-up during unobservable modes. This is demonstrated by simulations. The nonlinear estimators employed were the EKF, UKF and the RBPF. The latter two were used only for comparison of EKF results. Only the equations for EKF are presented below for brevity.

7.3.1 The Extended Kalman Filter

The Extended Kalman Filter (EKF) is based on linearizing the nonlinear system about the nominal state trajectory. The EKF algorithm in continuous-discrete form, adapted from [29, 30], is given below.

1. *Initialization*: Initialize the State x and the State Error Covariance P as

$$\begin{aligned}\widehat{x}_0^+ &= E[x_0] \\ P_0^+ &= E\left[(x_0 - \widehat{x}_0^+)(x_0 - \widehat{x}_0^+)^T\right]\end{aligned}\quad (7.20)$$

2. *Prediction*: For $k = 1, 2, \dots$, Integrate the Noise Free Model from $(k - 1)^+$ to k^- to Get x_k^- and P_k^- from x_{k-1}^+ and P_{k-1}^+ .

$$\begin{aligned}\dot{x} &= f(x, u, 0) \\ \dot{P} &= JP + PJ^T + Q\end{aligned}\quad (7.21)$$

where J is the Jacobian of f w.r.t. the state vector x . Q is the process noise covariance. The integration of state equation required at every time step was carried out using the Runge-Kutta fourth order method. The state error covariance matrix is found out using matrix exponentiation.

3. *Update*: At each k , using measurement y_k , update the state and state covariance estimates as:

$$\begin{aligned}v_k &= y_k - h(x_k^-, u_k, 0) \\ S_k &= H_k P_k^- H_k^T + R_k \\ K_k &= P_k^- H_k^T S_k^{-1} \\ x_k^+ &= x_k^- + K_k v_k \\ P_k^+ &= P_k^- - K_k S_k K_k^T\end{aligned}\quad (7.22)$$

7.3.2 Estimators with Adaptive Q and/or R

EKF and UKF require the knowledge of process and measurement noise covariance matrices, Q and R respectively, which need to be tuned for good results. A good choice of R can often be determined from the sensor characteristics. Determining Q is even more difficult since it represents the effects of unknown process disturbances and unmodeled dynamics.

Over a window of length M , unbiased estimates of mean and covariance of the measurement vector could be obtained as:

$$\begin{aligned}\bar{y} &= \frac{1}{M} \sum_{i=k-M+1}^k y_i \\ R_k &= \frac{1}{M-1} \sum_{i=k-M+1}^k (y_i - \bar{y})(y_i - \bar{y})^T\end{aligned}\quad (7.23)$$

Note that in the above expressions, the window lengths for mean and variance have been assumed to be the same. However, for large engine speeds, the local mean will have to be evaluated with small window length so that the extracted noise value is distinguished from genuine transitions in the engine variables. This, however, might leave very few samples for variance calculation. If the noise is assumed to be stationary, we can use a larger window size for variance calculation. Further, to reduce the online computational requirements, the moving average filter could be changed to an exponentially weighted one. Then:

$$\begin{aligned}\bar{y}_k &= \alpha \cdot y_k + (1 - \alpha) \cdot \bar{y}_{k-1} \\ R_k &= \beta \cdot (y_k - \bar{y}_k)(y_k - \bar{y}_k)^T + (1 - \beta) \cdot R_{k-1}, \quad 0 < \alpha, \beta \leq 1\end{aligned}\quad (7.24)$$

The coefficients α and β can now be independently tuned instead of the window length M . A small value for α and β (close to 0) would be equivalent to a large window, and vice versa. By choosing a large value for α (say 0.1) we could ensure that only the noise component, which is assumed to be having higher frequency components than the actual measurement, would contribute to the calculation of R_k . This also ensures that genuine transitions in measurements due to input changes are not misclassified as noise. The coefficient β , on the other hand, is chosen to be much smaller than α , with the assumption that the noise would be stationary over a larger window and its covariance would not change significantly.

To estimate the process noise covariance matrix Q , we modify the technique used in [31], which used the estimated covariance C_{v_k} of the innovation sequence Δv_k and Kalman gain K_k , with the exponentially weighted averaging:

$$\begin{aligned}C_{v_k} &= \beta \cdot (\Delta v_k - \Delta \bar{v}_k)(\Delta v_k - \Delta \bar{v}_k)^T + (1 - \beta) \cdot C_{v_{k-1}} \\ Q_k &= K_k C_{v_k} K_k^T \\ \Delta \bar{v}_k &= \alpha \cdot \Delta v_k + (1 - \alpha) \cdot \Delta \bar{v}_{k-1}, \quad 0 < \alpha, \beta \leq 1\end{aligned}\quad (7.25)$$

The values of α and β are typically different from the R -estimation case since the process disturbances have different characteristics than the measurement noise. A smaller value of α than that used in the R -estimation case was used here since process disturbances are usually of lower frequency than measurement noises. For faster convergence of the estimator, β was chosen to be larger than the R -estimation case.

7.4 Residual Prediction and Joint Estimation

Having developed the nominal estimator for engine state estimation, the next step in fault diagnosis is the residual generation under different fault hypotheses. This is typically done using a bank of estimators each one using a process model corresponding to a fault. In contrast, in this section, we develop a residual prediction scheme based on the nominal state estimate and Jacobian expressions from the EKF using a normal process model referred to hereafter as the “normal EKF”. The residual evaluation is then performed using the actual residuals obtained from the normal EKF and the predicted residuals from the EKF estimators using a process model for a given type of fault. The residual prediction based fault diagnosis scheme (Fig. 7.1) involves the following steps:

1. Estimation of nominal state from inputs and measurements using the EKF: From the nominal EKF, the instantaneous Kalman gain, actual residuals and the Jacobian matrix of state transition and measurement transition functions used for the state estimation are obtained.
2. Residual prediction stage: Using the Kalman gain and Jacobian matrix, the residual prediction stage predicts the residuals from the nominal estimator under each fault hypothesis. From this stage a residual vector corresponding to each hypothesized fault is obtained for unit magnitude of the fault.
3. Hypothesis testing stage: This stage generates fault detection functions for each fault using the predicted and actual residuals. The detection functions are generated by the Generalized Likelihood Ratio Test (GLRT) on the residuals. Each detection function should be compared to their respective thresholds, which are manually decided based on simulations in this chapter.
4. Fault isolation stage: In this stage, based on the detection functions from the hypothesis testing stage and on the indication whether they have exceeded their respective thresholds or not, the fault is isolated using predicate logic and knowledge of the process, under the assumption that at most a single fault could occur.
5. Fault parameter identification stage: Once a fault is detected and isolated by the previous steps, the parameter(s) associated with the particular fault could be found out either by a joint estimation [32] in the nominal EKF itself, or dual estimation, or some other separate estimators like particle filters.

The first step has already been discussed in the previous section. The last step involves well-known techniques as in [32]. The rest of the steps are explained next.

7.4.1 Residual Prediction

The residual prediction stage predicts the would-be residual from the nominal estimator (EKF) under the single fault assumption. Each fault is captured by a corresponding parameter. For example, the intake manifold leak can be captured by an additional area in the throttle. By defining this area to be a parameter, it is possible to predict the fault residuals in presence of IM leak. A similar method could be adopted for EM leak with the muffler area. If w^i is the parameter associated with fault i , and Δw^i a small change in w^i , indicating a fault, then from EKF equations in Sect. 7.3, we can predict the sign-reversed residual for fault i as:

$$\Delta v_k^i = \left(\frac{\partial h}{\partial x_k^-} \frac{dx_k^-}{dw^i} + \frac{\partial h}{\partial w^i} \right) \Delta w^i \quad (7.26)$$

Note that, $\frac{\partial h}{\partial x_k^-} = H_k$, which is already available from the nominal EKF routine. Using the linearized process model, the term $\frac{dx_k^-}{dw^i}$ in the above expression can be expressed as:

$$\begin{aligned} \frac{dx_{k+1}^-}{dw^i} &= \left(I + \frac{\partial f}{\partial x_k} \Delta T \right) \frac{dx_k^-}{dw^i} + \frac{\partial f}{\partial w^i} \Delta T \\ \frac{dx_k^-}{dw^i} &= \frac{dx_{k-1}^-}{dw^i} - K \left(\frac{\partial h}{\partial x_k^-} \frac{dx_{k-1}^-}{dw^i} + \frac{\partial h}{\partial w^i} \right) \end{aligned} \quad (7.27)$$

K is the Kalman gain and ΔT is the sampling interval. Note that, $\frac{\partial f}{\partial x_k} = J$, which is already available from the EKF routine. The effect of fault on Kalman gain K is ignored. The above residual prediction is valid for both process and sensor faults; however, it is possible to get simpler expressions for the latter category of faults. The sensor faults could be modelled by additive or multiplicative terms in the measurement equation and the partial derivative of the process map w.r.t. to the fault parameter is zero. Eq. (7.27) then reduces to:

$$\Delta v_k^i = \left(\frac{\partial h}{\partial w^i} \right) \Delta w_i. \quad (7.28)$$

The fault detection process is described next.

7.4.2 Fault Detection Based on Generalized Likelihood Ratio Test (GLRT) on Predicted Residuals

In the nominal case, the innovation residual from EKF contains noise from measurements, in addition to modelling and discretization errors. With the single fault assumption, the problem of fault detection from EKF residual at k th time instant could be posed as testing n binary hypotheses:

$$\begin{aligned} H_0 : \Delta v_k &= e_k \\ H_i : \Delta v_k &= \Delta v_k^i + e_k, \quad i = 1, 2, \dots, n \end{aligned} \quad (7.29)$$

where e_k is the error vector that captures measurement noise and model errors. As the estimator converges, we could also assume that e_k would be dominated by the measurement noise. In general, for a nonlinear system, the elements of e_k do not necessarily have the same variance and could also be correlated. If Δv_k^i is known from the model, for samples over a window of length M , assuming Gaussian e_k , the Neyman–Pearson (NP) test for these hypotheses is the likelihood ratio:

$$\frac{p(\Delta v_k/H_i)}{p(\Delta v_k/H_0)} = \frac{\frac{1}{\sqrt{(2\pi)^m |C_{e_k}|}} \exp\left(-\frac{1}{2} \sum_{k-M+1}^k (\Delta v_k - \Delta v_k^i)^T C_{e_k}^{-1} (\Delta v_k - \Delta v_k^i)\right)}{\frac{1}{\sqrt{(2\pi)^m |C_{e_k}|}} \exp\left(\sum_{k-M+1}^k \Delta v_k^T C_{e_k}^{-1} \Delta v_k\right)} > \delta_i \quad (7.30)$$

where C_{e_k} is the covariance of e_k and δ_i is the test threshold for fault i . m is the dimension of the residual vector. Taking logarithms, this transforms to the test statistic [33]:

$$T(\Delta v_k; \Delta v_k^i) = \sum_{k-M+1}^k \Delta v_k^T C_{e_k}^{-1} \Delta v_k^i > \delta_i \quad (7.31)$$

Note that the threshold here is different from Eq. (7.30), although we have used the same notation. This test is suboptimal when the parameters C_{e_k} and Δv_k^i are not known and their estimates are used instead, and is known as the Generalized Likelihood Ratio Test (GLRT) [33]. The magnitude of the fault will not change the test threshold, but changes the probability of detection. The threshold could be decided based on the minimum fault magnitude that is needed to be detected. However, this might introduce false alarms in case such a threshold is exceeded even during the nominal case because of model errors. Note that in this test, the absolute value of threshold need not be less than 1. To sum up, instead of the inner product of actual residual signal with predicted residual we correlate it with the residual modified by error covariance. The multiplication by $C_{e_k}^{-1}$ is essentially a normalization process equivalent to adaptive thresholding.

From the residual prediction process, Δv_k^i is known only up to a constant. Consequently, the test threshold has to be found out for some minimum fault parameter magnitude to be detected for each fault. If Δv_k^i has more than one elements, a threshold between 0 and 1 could be obtained by normalizing the residuals:

$$T(\Delta v_k; \Delta v_k^i) = \frac{1}{M} \sum_{k=M+1}^k \frac{(\Delta v_k^i)^T C_{e_k}^{-1} \Delta v_k}{\left\| \sqrt{C_{e_k}^{-1}} \Delta v_k^i \right\| \left\| \sqrt{C_{e_k}^{-1}} \Delta v_k \right\|} > \delta_i \quad (7.32)$$

The covariance C_{e_k} could be estimated online the same manner in which R matrix was estimated during EKF. The thresholds are determined by simulation under different fault scenarios. When multiple fault thresholds are exceeded, an isolation procedure is needed, that is described next.

7.4.3 Fault Isolation

Let the binary variable F_i denote the i th fault and G_i the associated binary (logistic) detection function. The binary variable G_i is obtained from the continuous variable g_i , the fault detection function for i th fault, by comparing with the fault detection threshold for the i th fault. If g_i crosses its threshold, G_i is 1, otherwise it is 0. The fault isolation could now proceed with the following assumptions.

1. No fault detection functions G_i are active (high) during normal operation, i.e.,

$$(\forall i \neg F_i) \Rightarrow (\forall j \neg G_j). \quad (7.33)$$

2. Missed detections are possible; however, when the i th fault is present, if j th detection function $G_j (j \neq i)$ is active, G_i is also active, i.e.,

$$\forall i (F_i \wedge (G_j | j \neq i) \Rightarrow G_i). \quad (7.34)$$

The fault detection thresholds are chosen from simulations in such a way that the above conditions are met. When multiple G_i are active, these conditions can be used to resolve the conflict, whenever possible. This can be performed using a Fault Incidence Matrix (FIM). This process is illustrated in the results section.

Certain ad-hoc methods might be employed when the above conditions cannot isolate the fault. Firstly, because of our single fault assumption, for fault detection functions associated with cylinder faults, if only one out of all cylinder fault functions is triggered for a particular fault, most likely the fault is with the particular cylinder. This eliminates the possibility of both faults in other cylinders and faults in a non-cylinder component. For example, suppose for some unknown fault, the injector 1 fault function for cylinder 1 is triggered and the EM leak function is also

triggered, but the injector fault functions for other cylinders are not triggered. This means that the fault is most likely the injector 1 fault, and not the EM leak fault. Further, again from the single fault assumption, if all of the cylinder fault detection functions are triggered for some unknown fault, and in addition a non-cylinder fault function is triggered, the fault is most likely the non-cylinder fault.

The proposed fault diagnosis scheme obviates the need for multiple estimators. It reuses the Jacobians from EKF estimators. Hence, overall computational cost is reduced very much in comparison to a bank of estimators typically used in diagnosis. In addition to the EKF terms, the terms and operations required for fault diagnosis are the partial derivatives of state transition function and measurement transition functions w.r.t. the fault parameters, and the multiplications of matrices with vectors. The complexities of these operations are polynomial in both the number of states and number of faults. When the system order and/or the number of faults considered are large, it may be advisable to model each sub-system separately, so that the order and number of faults considered in the individual sub-systems is relatively small. The proposed scheme can then be employed in local diagnosis as part of a decentralized global diagnosis structure, such as the one proposed in [34].

7.5 Results

The results are presented for the estimation and the fault diagnosis schemes.

7.5.1 Estimation Results

To validate the modelling and estimation, the estimation results are compared against values generated from an AMESim model of the 4-cylinder engine (considered as the ‘True’ values). The simulation duration was 4 s in all cases. To compare the nonlinear estimators for the engine, we use the normalized mean square error (MSE) as the measure.

The estimation results with adaptive computation of Q and R matrices are shown in Fig. 7.3. It was found that the UKF performance with adaptive Q matrix was slightly poorer than the constant matrix case and hence only R was adapted. The UKF estimation was carried out with filter parameters $\alpha = 1$, $\beta = 2$, and $\kappa = 3$. The state dimension is 24, and hence the number of UKF weights is 49. Due to the marginalization using EKF inside the PF, the RBPF does not require as many particles as weights in UKF. It was found that increasing beyond a few particles does not have much bearing on the estimator performance, and hence, only 4 particles were used in the implementation to save the computational time. The EKF and consequently RBPF performances were greatly enhanced by the adaptation of Q and R , as indicated by the normalized MSEs for both the constant and adaptive Q/R cases plotted in Fig. 7.4 and their sums tabulated in Table 7.3. The simulation

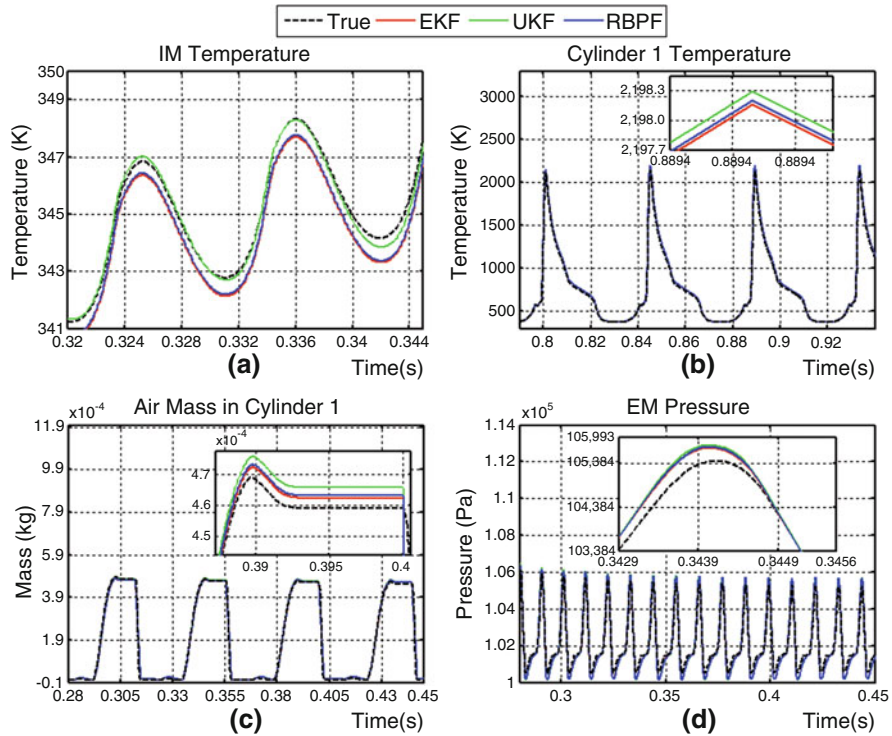


Fig. 7.3 Engine state estimation results for different estimators with adaptive Q and R matrices: **a)** Intake Manifold (IM) Temperature, **b)** Cylinder 1 Temperature, **c)** Air Mass in Cylinder 1, and **d)** Exhaust Manifold (EM) Pressure

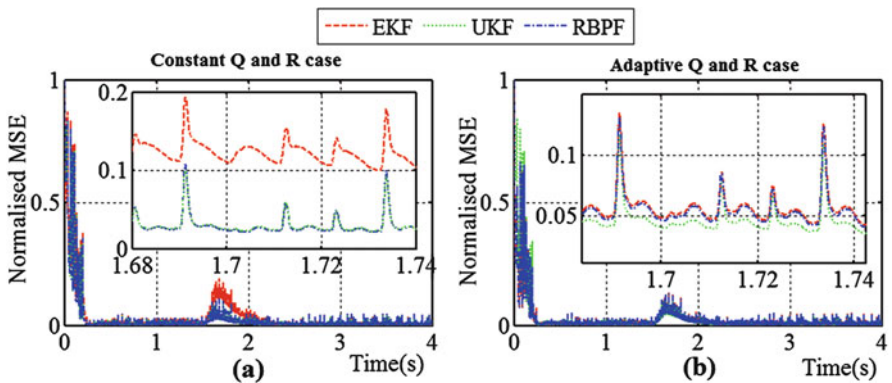


Fig. 7.4 Normalized MSE for estimators under **(a)** constant, and **(b)** adaptive Q/R matrices

Table 7.3 Comparison of estimators under constant and adaptive Q and R

Estimator	Constant Q and R case		Adaptive Q and R case	
	Simulation time (s)	Normalized MSE	Simulation time (s)	Normalized MSE
EKF	42	0.1058	43	0.0757
UKF	1167	0.0735	1182	0.0798
RBPF	244	0.0752	325	0.0750

Table 7.4 Fault magnitudes and detection thresholds for the simulation case

Fault no.	Fault	Fault magnitude	GLRT- normalized inner product
F1	IM leak	50 mm ²	0.6
F2	EM leak	50 mm ²	0.06
F3	Injector 1 fault	0.5 (injector pulse width halved)	0.15
F4	Injector 2 fault	0.5	0.15
F5	P_{im} sensor bias	1000 Pa	0.5
F6	T_{im} sensor bias	10 K	0.5
F7	P_{em} sensor bias	1000 Pa	0.5
F8	IV not closing (cylinder 1)	0.04 mm (lift)	0.5
F9	IV not closing (cylinder 2)	0.04 mm (lift)	0.5
F10	EV not closing (cylinder 1)	0.04 mm (lift)	0.5
F11	EV not closing (cylinder 2)	0.04 mm (lift)	0.5

times are also shown. It is seen that EKF with analytically evaluated Jacobian matrices performs very well by adaptation of Q and R while being computationally the least expensive, and hence could be a better choice for online implementation. The simulation time for UKF is prohibitively high for online implementation.

7.5.2 Fault Diagnosis Results

The various faults that were inserted in the simulation model are shown in Table 7.4 along with their magnitudes.

The Fault Incidence Matrix (FIM), showing the response of detection functions to different faults, is given in Table 7.5.

The fault detection process is explained with one of the faults, namely the intake manifold leak. A leak fault of 50 mm² area was introduced at 3 s in the AMESimTM model. The nominal estimator was run with data from AMESimTM simulation. Since the intake manifold is usually at a pressure slightly less than atmospheric to let the air in during the intake stroke, the leak causes an additional flow into the manifold. This causes an increase in IM pressure. Further, the leak causes the ratio of EGR flow to atmospheric air flow to be relatively lower than the case for no leak, because now there is an additional air flow through the leak. Since the atmospheric air

Table 7.5 Fault incidence matrix for GLRT based fault detection

	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	Fault isolation
G1		✓									✓	✓	{F1}
G2		✓	✓			✓		✓	✓	✓	✓	✓	{F2}
G3				✓					✓		✓	✓	{F3}
G4					✓					✓	✓	✓	{F4}
G5		✓				✓					✓	✓	{F5}
G6							✓						{F6}
G7								✓					{F7}
G8		✓				✓			✓		✓	✓	{F8}
G9		✓				✓				✓	✓	✓	{F9}
G10											✓		{F10}
G11												✓	{F11}

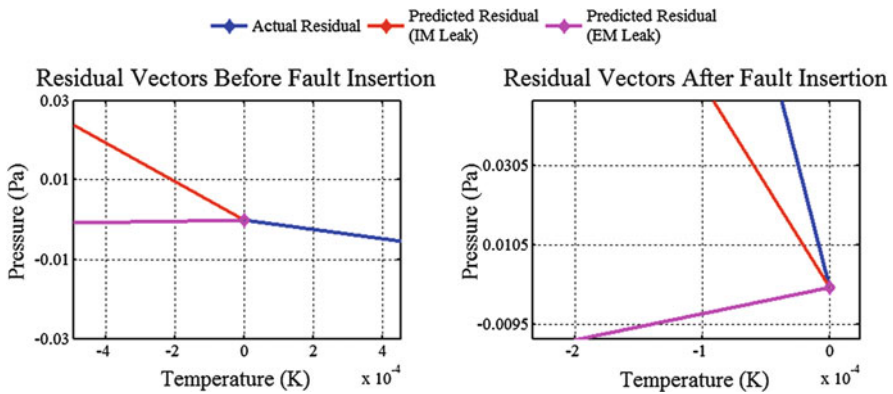


Fig. 7.5 Predicted and actual residuals before and after IM leak fault insertion

temperature is lower than EGR air temperature, the IM temperature reduces during the leak. Therefore, IM pressure and temperature are two mainly affected variables by the leak. Since the residuals are results of differences between the actual and the expected (under no fault) values, we expect that, under the IM leak fault, the actual and predicted pressure residuals will be higher than zero, whereas the same for temperature will be lower than zero.

The vectors for predicted and actual residuals for the inner product case before and after the insertion of the fault are plotted in Fig. 7.5 and their values are plotted in Fig. 7.6. Only the IM temperature and pressure residuals (mean values during 2–3 s for nominal case and 5–6 s for faulty case) were plotted here. It is clearly seen that after the occurrence of the IM leak fault, the angle between predicted residual for IM leak and actual residuals has decreased considerably, whereas the predicted residual for EM leak is still at a large angle from the actual residual. It could be seen from Fig. 7.6 that after the incidence of the fault, the residual patterns are very similar, but their magnitudes differ because the residual prediction is carried out

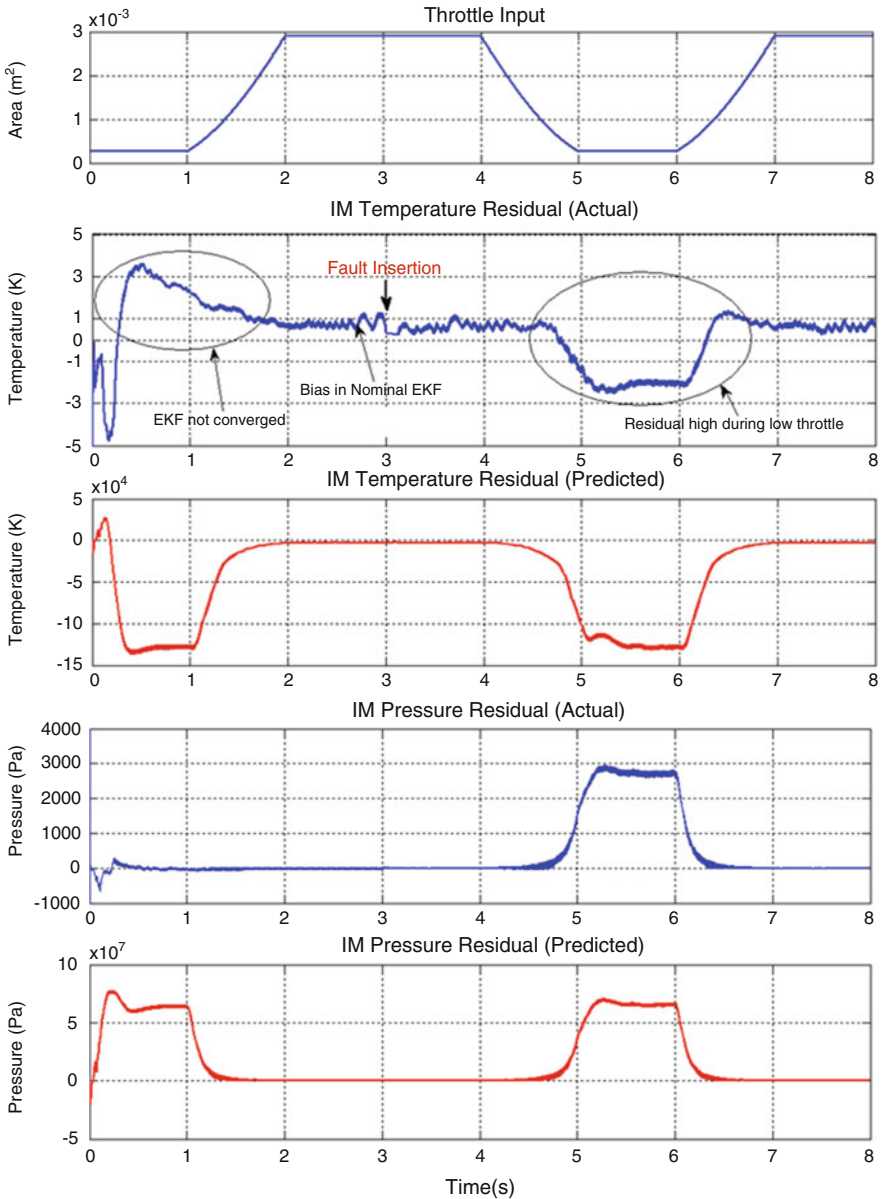


Fig. 7.6 Predicted and actual residuals for intake manifold leak fault. The detection process is turned off until the estimator converges

for unit magnitude of the fault. The nominal estimator takes a while to converge as seen from the temperature residual. Only after the trace of error covariance matrix is below a threshold the fault detection process is initiated. It could also be seen

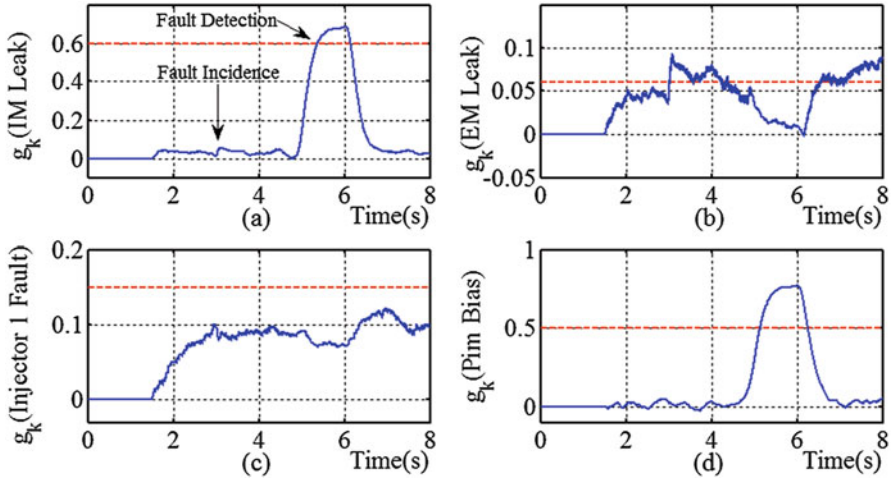


Fig. 7.7 Fault detection functions g_k for **a)** IM Leak, **b)** EM Leak, **c)** Injector 1 Fault, and **d)** P_{im} Bias, with only the IM leak fault inserted, using the GLRT (normalized) on residuals

that the effect of leak is prominent only during low throttle values. During high throttle, the leak flow rate is very small compared to the throttle flow and hence not detectable. A small bias is also observed for the nominal estimation. These biases could masquerade as faults unless the detection thresholds are appropriately chosen.

Some of the fault detection functions under IM leak fault are shown in Fig. 7.7 along with their thresholds (in red). From the FIM (Table 7.5), it is seen that G1, G2, G5, G8 and G9 have been activated. Hence the fault could be from the set {F1, F2, F5, F8, F9}. From the FIM, it is seen that only G2 is triggered for fault F2. Hence the fault is not F2. Further, G1 is not triggered for F5, indicating that the fault is not F5. The remaining {F8, F9} represent the IV faults for cylinders 1 and 2. Since both are triggered, we infer that the fault is a common fault not particular to any cylinder. Hence {F8, F9} is eliminated. We are left with F1, and hence the fault is isolated. Similar procedure is followed for other faults.

7.6 Conclusions

For accurate fault detection, isolation and identification of faults in hybrid systems such as engine, detailed physics based modelling is necessary. The computational complexity of such a model in real time implementation could be compensated by using a computationally less expensive EKF estimator with adaptive estimation of process and noise covariance matrices. Further, by employing a residual prediction based fault diagnosis scheme which reuses the intermediate terms generated during EKF, despite the model complexity, the overall time complexity of the fault

diagnosis scheme is reduced. Such a scheme could be employed for other hybrid systems, provided they have no state resets. In addition to simulations, the results of the fault diagnosis scheme have been verified on an experimental engine having no EGR. The experimental results were omitted for brevity.

The sensitivity and certainty of fault detection is dictated by the detection thresholds. In this chapter, the thresholds were manually chosen for the minimum magnitude of faults to be detected. Ideally, the fault detection thresholds have to be chosen by minimizing some risk functions associated with misclassification. However, when there are many faults, analytical calculation and minimization of such risk functions is cumbersome. A more practical approach to threshold selection would be to run Monte Carlo simulations for many different thresholds and plot the Receiver Operating Characteristics (ROC) graph [35] under different fault magnitudes and noise conditions, and chose the thresholds which provide the maximum *true positive rate* and minimum *false positive rate*.

Once a fault is detected, the fault parameter could be identified using well-known parameter estimation techniques such as dual or joint estimation [32]. With the newly estimated parameter value, the diagnoser is now ready for detecting further faults. To handle cases where the drift pattern of fault parameter may be complex, either a physical parameter drift model should be used, or some machine learning techniques need to be applied on data streams [36]. Since the engine is a hybrid dynamical system, the drift monitoring shall be performed only in those discrete modes where the continuous dynamics is impacted by the fault parameter, as proposed in [37].

References

1. Nyberg, M. (2002). Model-based diagnosis of an automotive engine using several types of fault models. *IEEE Transactions on Control Systems Technology*, 10(5), 679–689.
2. Nyberg, M., & Stutte, T. (2004). Model based diagnosis of the air path of an automotive diesel engine. *Control Engineering Practice*, 12(5), 513–525.
3. Kim, Y. W., Rizzoni, G., & Utkin, V. (1998). Automotive engine diagnosis and control via nonlinear estimation. *IEEE Control Systems*, 18(5), 84–99.
4. Andersson, P., & Eriksson, L. (2002). *Detection of exhaust manifold leaks on a turbocharged SI-engine with wastegate* (no. 2002-01-0844). SAE Technical Paper.
5. Bar-Shalom, Y., Li, X. R., & Kirubarajan, T. (2004). *Estimation with applications to tracking and navigation: Theory algorithms and software*. New York: Wiley.
6. Doucet, A., de Freitas, N., & Gordon, N. (2001). An introduction to sequential Monte Carlo methods. In A. Doucet, N. de Freitas, & N. Gordon (Eds.), *Sequential Monte Carlo methods in practice, Statistics for engineering and information science*. New York, NY: Springer.
7. Doucet, A., De Freitas, N., Murphy, K., & Russell, S. (2000, June). Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the sixteenth conference on uncertainty in artificial intelligence* (pp. 176–183). San Francisco: Morgan Kaufmann Publishers Inc.
8. Andersson, P., & Eriksson, L. (2001). *Air-to-cylinder observer on a turbocharged SI-engine with wastegate* (no. 2001-01-0262). SAE Technical Paper.

9. Nyberg, M., & Nielsen, L. (1997). *Model based diagnosis for the air intake system of the SI-engine* (no. 970209). SAE Technical Paper.
10. Shiao, Y., & Moskwa, J. J. (1995). Cylinder pressure and combustion heat release estimation for SI engine diagnostics using nonlinear sliding observers. *IEEE Transactions on Control Systems Technology*, 3(1), 70–78.
11. Yan, F., & Wang, J. (2012). Design and robustness analysis of discrete observers for diesel engine in-cylinder oxygen mass fraction cycle-by-cycle estimation. *IEEE Transactions on Control Systems Technology*, 20(1), 72–83.
12. Chen, P., & Wang, J. (2013). Observer-based estimation of air-fractions for a diesel engine coupled with aftertreatment systems. *IEEE Transactions on Control Systems Technology*, 21(6), 2239–2250.
13. Buckland, J. H., Freudenberg, J., Grizzle, J. W., & Jankovic, M. (2009, June). Practical observers for unmeasured states in turbocharged gasoline engines. In *American control conference. ACC'09* (pp. 2714–2719). IEEE.
14. Xue, W., Bai, W., Yang, S., Song, K., Huang, Y., & Xie, H. (2015). ADRC with adaptive extended state observer and its application to air–fuel ratio control in gasoline engines. *IEEE Transactions on Industrial Electronics*, 62(9), 5847–5857.
15. Butt, Q. R., & Bhatti, A. I. (2008). Estimation of gasoline-engine parameters using higher order sliding mode. *IEEE Transactions on Industrial Electronics*, 55(11), 3891–3898.
16. Iqbal, M., Bhatti, A. I., Ayubi, S. I., & Khan, Q. (2011). Robust parameter estimation of nonlinear systems using sliding-mode differentiator observer. *IEEE Transactions on Industrial Electronics*, 58(2), 680–689.
17. Sengupta, S., Mukhopadhyay, S., Deb, A., Pattada, K., & De, S. (2011). Hybrid automata modeling of SI gasoline engines towards state estimation for fault diagnosis. *SAE International Journal of Engines*, 5(3), 759–781.
18. Nadeer, E. P., Patra, A., & Mukhopadhyay, S. (2015). Model based online fault diagnosis of automotive engines using joint state and parameter estimation. In *2015 annual conference of the prognostics and health management society*, Coronado, CA, USA.
19. Schilling, A., Amstutz, A., & Guzzella, L. (2008). Model-based detection and isolation of faults due to ageing in the air and fuel paths of common-rail direct injection diesel engines equipped with a λ and a nitrogen oxides sensor. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 222(1), 101–117.
20. Riggins, R. N., & Rizzoni, G. (1990, May). The distinction between a special class of multiplicative events and additive events: Theory and application to automotive failure diagnosis. In *American control conference* (pp. 2906–2911). IEEE.
21. Vasu, J. Z., Deb, A. K., & Mukhopadhyay, S. (2015). MVEM-based fault diagnosis of automotive engines using Dempster–Shafer theory and multiple hypotheses testing. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(7), 977–989.
22. Pernestål, A. (2009). *Probabilistic fault diagnosis with automotive applications*. Doctoral dissertation, Linköping University Electronic Press.
23. Pattipati, K., Kodali, A., Luo, J., Choi, K., Singh, S., Sankavaram, C., & Qiao, L. (2008). An integrated diagnostic process for automotive systems. In *Computational intelligence in automotive applications* (pp. 191–218). Berlin: Springer.
24. Sangha, M. S., Yu, D. L., & Gomm, J. B. (2006). On-board monitoring and diagnosis for spark ignition engine air path via adaptive neural networks. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 220(11), 1641–1655.
25. Felder, R. M., & Rousseau, R. W. (2008). *Elementary principles of chemical processes*. New York: Wiley.
26. Guzzella, L., & Onder, C. H. (2010). *Introduction to modeling and control of internal combustion engine systems*. Berlin: Springer.
27. Heywood, J. B. (1988). *Internal combustion engine fundamentals* (Vol. 930). New York: McGraw-Hill.
28. Annand, W. J. D. (1963). Heat transfer in the cylinders of reciprocating internal combustion engines. *Proceedings of the Institution of Mechanical Engineers*, 177(1), 973–996.

29. Simon, D. (2006). *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. Hoboken, NJ: Wiley.
30. Särkkä, S. (2006). *Recursive Bayesian inference on stochastic differential equations*. Espoo, Finland: Helsinki University of Technology.
31. Mohamed, A. H., & Schwarz, K. P. (1999). Adaptive Kalman filtering for INS/GPS. *Journal of Geodesy*, 73(4), 193–203.
32. Haykin, S. S. (Ed.). (2001). *Kalman filtering and neural networks* (p. 304). New York: Wiley.
33. Kay, S. M. (1998). *Fundamentals of statistical signal processing: Detection theory* (Vol. 2). Upper Saddle River, NJ: Prentice Hall.
34. Sayed-Mouchaweh, M., & Lughofer, E. (2015). Decentralized fault diagnosis approach without a global model for fault diagnosis of discrete event systems. *International Journal of Control*, 88(11), 2228–2241.
35. Van Trees, H. L. (2001). *Detection, estimation, and modulation theory, part I: Detection, estimation, and linear modulation theory*. New York: Wiley.
36. Sayed-Mouchaweh, M. (2016). *Learning from data streams in dynamic environments, Springer briefs in electrical and computer engineering*. Cham: Springer.
37. Toubakh, H., & Sayed-Mouchaweh, M. (2016). Hybrid dynamic classifier for drift-like fault diagnosis in a class of hybrid dynamic systems: Application to wind turbine converters. *Neurocomputing*, 171, 1496–1516.

Chapter 8

Diagnosis of Hybrid Systems Using Structural Model Decomposition



Matthew J. Daigle, Anibal Bregon, and Indranil Roychoudhury

8.1 Introduction

Automated fault diagnosis is critical for complete system autonomy. In order for engineering systems to function in the real world, including extreme environments, the ability to self-diagnose faults and failures and then mitigate them by control or repair actions is crucial. Many engineering systems are *hybrid* in nature, i.e., they exhibit both continuous and discrete behaviors. The combination of continuous and discrete dynamics makes the problem of robust and efficient fault diagnosis significantly more challenging.

In a hybrid system, the system behavior is defined by a set of discrete *modes*. In each of these modes, a different set of continuous dynamics governs the system behavior. Discrete dynamics define how the system transitions from one mode to another. For example, consider an electric circuit with ten switching elements. If each switch can be in one of two states (the *on* state or the *off* state), then such a system has 2^{10} possible system-level modes. Therefore, a diagnosis algorithm, in general, must consider all possible modes of such a system.

M. J. Daigle (✉)
NIO USA, Inc., San Jose, CA, USA
e-mail: matthew.daigle@nio.io

A. Bregon
Departamento de Informática, Universidad de Valladolid, Valladolid, Spain
e-mail: anibal@infor.uva.es

I. Roychoudhury
Stinger Ghaffarian Technologies, Inc., NASA Ames Research Center, Moffett Field,
Mountain View, CA, USA
e-mail: indranil.roychoudhury@nasa.gov

Further, faults may manifest as direct changes in system parameters, called *parametric faults*, or as changes in the system mode, called *discrete faults*. So a diagnosis system must reason over different types of faults and possible mode transitions during the fault isolation process. The effects of a fault are also mode-dependent, and observation delays (e.g., due to delays in signal filtering within a fault detection algorithm, or communication delays) may cause the observed effects to be inconsistent with the current mode of the system, but consistent with a previous mode. All of these complications can significantly complicate the reasoning process [1].

Researchers have been intrigued by the fault diagnosis problem for hybrid systems for many years, and many different proposals for hybrid systems diagnosis exist in the literature. During the last decade or so, modeling and diagnosis for hybrid systems has been an important topic of research from both Systems Dynamics and Control Engineering (FDI) and the Artificial Intelligence Diagnosis (DX) communities. In the FDI community, several hybrid system diagnosis approaches have been developed, where parameterized ARR_s are used [2, 3]. However, such approaches are not suitable for systems with high nonlinearities or a large set of modes. In the DX community, some approaches have used hybrid automata to model the complete set of modes and transitions between them. In those cases, diagnosis is viewed as a hybrid system state estimation problem, and approached through probabilistic [4, 5] or set-theoretic approaches [6]. Another solution has been to use an automaton to track the system mode, and then use a different technique to diagnose the continuous behavior (for example, using a set of ARR_s for each mode [7], or parameterized ARR_s for the complete set of modes [8]). Nevertheless, one of the main difficulties regarding state estimation using these techniques is the need to pre-enumerate the set of *all* possible system-level modes and mode transitions, which is difficult for complex systems. Another approach to fault diagnosis, as shown in [1, 9, 10], qualitatively abstracts the transients in residual deviations and compares them with predicted fault transients. The prediction of fault transients in different modes of the system can also be computationally very expensive.

In order to address the challenges mentioned above, the techniques of *compositional modeling* and *structural model decomposition* have been developed. Building and representing hybrid system models in a *compositional* manner solves the mode pre-enumeration problem. In compositional modeling, the discrete modes are defined at a local level (e.g., at the component level) such that the system-level mode is defined implicitly by the local component-level modes. Since this allows the modeler to focus on the discrete behavior only at the component level, the pre-enumeration of all the system-level modes can be avoided [11, 12]. Additionally, building models in a compositional way facilitates reusability and maintenance, and allows the validation of the components individually before they are composed to create the global hybrid system model.

Structural model decomposition [13] offers another means to decrease the complexity of the hybrid system diagnosis problem [14–16]. In continuous systems diagnosis, structural model decomposition is a popular approach because it allows a global model to be decomposed into local submodels, with each submodel dependent on only a subset of the system faults [13]. As a result, the diagnosis problem becomes much simpler. In hybrid systems, structural model decomposition can significantly decrease the complexity of the problem as well. In addition to minimizing the set of fault effects, each submodel has only a limited number of modes. So instead of reasoning over the exponential number of system-level modes, reasoning need only be performed over the significantly smaller set of submodel modes. Further, structural model decomposition results in *computationally independent* submodels, which lends itself naturally to a *distributed* implementation.

One solution for qualitative fault isolation using structural model decomposition is presented in [17]. Within this approach, however, observation delays are not taken into account and it is applicable only to systems that are modeled using hybrid bond graphs (HBGs). A more efficient model-based methodology for diagnosis, which integrates structural model decomposition within the Hybrid Diagnosis Engine (HyDE), and uses a compositional modeling approach [11], is developed in [18]. The approach demonstrated how the integration of structural model decomposition reduces the computational complexity associated with the fault diagnosis of hybrid systems. The approach presented in this chapter is related to that in [19, 20], but differs in two major ways. First, the former work was based on modeling using HBGs, whereas the modeling framework used here is more general (in which HBGs are a special case). Second, that work was based on a global system model, while in this work, the approach is based on local submodels computed through structural model decomposition.

This chapter presents a model-based, qualitative fault diagnosis framework for hybrid systems, which can diagnose both parametric and discrete faults, and can handle observation delays. The underlying system model is built using a compositional modeling methodology, and structural model decomposition is used to decompose the model into independent submodels, and thus decompose the diagnosis problem and significantly reduce the associated computational complexity. The Advanced Diagnostics and Prognostics Testbed (ADAPT) [21], an electrical power distribution system developed at NASA Ames Research Center, is used as a case study to demonstrate that the approach can correctly isolate faults in hybrid systems even if the system transitions among different mode changes and presents observation delays during the isolation process.

The chapter is organized as follows. Section 8.2 presents the approach to hybrid systems modeling. The diagnosis problem for hybrid systems is formulated in Sect. 8.3 and the qualitative fault isolation approach is presented in Sect. 8.4. Section 8.5 describes the ADAPT case study and presents the experimental results of applying our hybrid fault diagnosis algorithm. Finally, Sect. 8.6 concludes the chapter.

8.2 Hybrid Systems Modeling

Most practical systems demonstrate mixed discrete and continuous dynamics, termed *hybrid dynamics*, and hence, such systems are termed *hybrid systems*. Here, the system can be thought of as inhabiting different operating *modes*, where in each mode there is a specific set of continuous dynamics that governs the system behavior within that mode. The discrete dynamics consist of the behavior governing the transitions between modes.

A circuit example, shown in Fig. 8.1, will be used throughout the chapter to illustrate the approach. The circuit includes a voltage source, V , two capacitors, C_1 and C_2 , two inductors, L_1 and L_2 , two resistors, R_1 and R_2 , and two switches, Sw_1 and Sw_2 , connected through a set of series and parallel connections. Sensors measure the current or voltage in different locations (i_3 , v_8 , and i_{11} , as indicated in Fig. 8.1). Each switch can be in one of two modes: *on* and *off*. Thus, this circuit can be represented as a hybrid system, with four system-level modes.

There are many different modeling formalisms to represent such a system, such as hybrid automata [22] and hybrid bond graphs [23]. From a modeling perspective, it is more convenient to follow a compositional modeling approach, where only local, component-level modes are explicitly defined, and the system-level modes are defined implicitly. In the following, the compositional modeling framework is described, followed by a discussion on causality assignment, and then the structural model decomposition approach.

8.2.1 Compositional Modeling

In a compositional modeling approach, the system is viewed as a set of connected components. Each component is defined by a set of discrete modes, with a different set of constraints describing the continuous dynamics of the component in each mode.

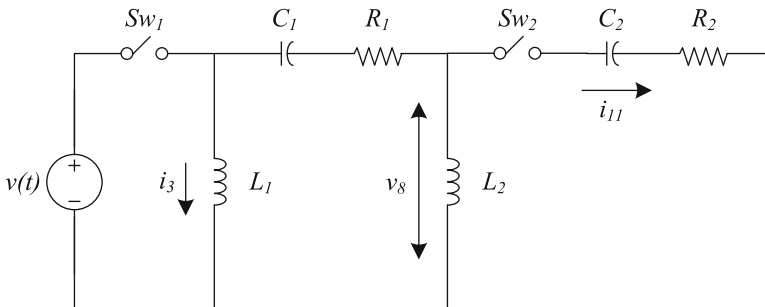


Fig. 8.1 Electrical circuit example

The fundamental building blocks of a hybrid system model are *variables* and *constraints* among those variables. A constraint is defined as follows:

Definition 1 (Constraint) A constraint c is a tuple (ε_c, V_c) , where ε_c is an equation involving variables V_c .

A *component* is defined by a set of constraints over a set of variables. The constraints are partitioned into different sets, one for each component mode:

Definition 2 (Component) A *component* κ with n discrete modes is a tuple $\kappa = (V_\kappa, \mathcal{C}_\kappa)$, where V_κ is a set of variables and $\mathcal{C}_\kappa = \{C_\kappa^1, C_\kappa^2, \dots, C_\kappa^n\}$ is a set of constraints sets, where C_κ^m refers to the set of constraints defining the continuous dynamics in mode m .

Example 1 The components of the circuit are defined in Table 8.1.¹ They include $V, C_1, C_2, L_1, L_2, R_1, R_2, Sw_1, Sw_2$, as well as components for series and parallel connections.

Example 2 Consider the component Sw_2 (κ_{10}). It has two modes: *off* (represented as mode 1 in Table 8.1) and *on* (represented as mode 2). In the *off* mode, it has three constraints setting each of its currents (i_9, i_{10}, i_{11}) to 0. In the *on* mode, it has also three constraints, setting the three currents equal to each other and establishing that the voltages sum up (it acts like a series connection when in the *on* mode).

A system model is defined as a set of components:

Definition 3 (Model) A *model* $\mathcal{M} = \{\kappa_1, \kappa_2, \dots, \kappa_k\}$ is a finite set of k components for $k \in \mathbb{N}$.

Example 3 The model of the electrical circuit is made up of all the components detailed in Table 8.1, i.e., $\mathcal{M} = \{\kappa_1, \kappa_2, \dots, \kappa_{15}\}$. For each component, the variables and constraints are defined for each component mode.

The set of variables for a model \mathcal{M} , $V_{\mathcal{M}}$, is the union of all the component variable sets, i.e., for d components, $V_{\mathcal{M}} = V_{\kappa_1} \cup V_{\kappa_2} \cup \dots \cup V_{\kappa_d}$. The interconnection structure of the model is captured using shared variables between components, i.e., components κ_i and κ_j are connected if $V_{\kappa_i} \cap V_{\kappa_j} \neq \emptyset$.

Example 4 In the circuit model, component κ_5 (Series Connection₁) is connected to κ_3 (Parallel Connection₁) through i_4 , to κ_6 (R_1) through i_5 and v_5 , to κ_7 (C_1) through i_6 and v_6 , and κ_8 (Parallel Connection₂) through i_7 and v_7 .

The model constraints, $C_{\mathcal{M}}$, are a union of the component constraints over all modes, i.e., $C_{\mathcal{M}} = \mathcal{C}_{\kappa_1} \cup \mathcal{C}_{\kappa_2} \cup \dots \cup \mathcal{C}_{\kappa_d}$. Constraints are exclusive to components, that is, a constraint $c \in C_{\mathcal{M}}$ belongs to exactly one \mathcal{C}_κ for $\kappa \in \mathcal{M}$.

¹Here, we denote derivatives using dot notation.

Table 8.1 Components of the electrical circuit

Component	Mode	Constraints
κ_1 : V	1	$v_1 = u_v$
κ_2 : Sw ₁	1	$i_1 = 0$ $i_2 = 0$
	2	$i_1 = i_2$ $v_1 = v_2$
κ_3 : parallel connection ₁	1	$v_2 = v_3$
		$v_2 = v_4$
		$i_2 = i_3 + i_4$
κ_4 : L ₁	1	$\dot{f}_3 = v_3$
		$i_3 = f_3/L_1$
		$f_3 = \int_{t_0}^t \dot{f}_3$
κ_5 : series connection ₁	1	$i_4 = i_5$
		$i_4 = i_6$
		$i_4 = i_7$
		$v_4 = v_5 + v_6 + v_7$
κ_6 : R ₁	1	$v_5 = i_5 * R_1$
κ_7 : C ₁	1	$\dot{q}_6 = i_6$
		$v_6 = q_6/C_1$
		$q_6 = \int_{t_0}^t \dot{q}_6$
κ_8 : parallel connection ₂	1	$v_7 = v_8$
		$v_7 = v_9$
		$i_7 = i_8 + i_9$
κ_9 : L ₂	1	$\dot{f}_8 = v_8$
		$i_8 = f_8/L_2$
		$f_8 = \int_{t_0}^t \dot{f}_8$
κ_{10} : Sw ₂	1	$i_9 = 0$
		$i_{10} = 0$
		$i_{11} = 0$
	2	$i_9 = i_{10}$ $i_9 = i_{11}$ $v_9 = v_{10} + v_{11}$
κ_{11} : R ₂	1	$v_{10} = i_{10} * R_2$
κ_{12} : C ₂	1	$\dot{q}_{11} = i_{11}$
		$v_{11} = q_{11}/C_2$
		$q_{11} = \int_{t_0}^t \dot{q}_{11}$
κ_{13} : current sensor ₁₁	1	$i_{11}^* = i_{11}$
κ_{14} : voltage sensor ₈	1	$v_8^* = v_8$
κ_{15} : current sensor ₃	1	$i_3^* = i_3$

To refer to a particular mode of a model we use the concept of a *mode vector*. A mode vector \mathbf{m} specifies the current mode of each of the components of a model. So, the constraints for a mode \mathbf{m} are denoted as $C_{\mathcal{M}}^{\mathbf{m}}$.

Example 5 Consider a model with five components, then if $\mathbf{m} = [1, 1, 3, 2, 1]$, it indicates that components κ_1 , κ_2 , and κ_5 use constraints of their mode 1, component κ_3 uses constraints of its mode 3, and component κ_4 uses constraints of its mode 2.

For shorthand, the mode vector will refer to the modes only of the components with multiple modes. So, for the circuit, it refers only to components κ_2 and κ_{10} , resulting in four possible mode vectors, $[1\ 1]$, $[1\ 2]$, $[2\ 1]$, and $[2\ 2]$.

The switching behavior of each component can be defined using a finite state machine or a similar type of control specification. For the purposes of this chapter, the switching behavior is viewed as a black box where the mode change event is given, and refer the reader to many of the approaches already proposed in the literature for modeling the switching behavior [22, 23]. Although we do not consider state resets explicitly in this paper, this may also be viewed as an output of the switching behavior.

8.2.2 Fault Modeling

A fault is the cause of an unexpected, persistent deviation of the system behavior from the acceptable nominal behavior. In the continuous dynamics, faults are represented as parameter changes in $\Theta_{\mathcal{M}} \subset V_{\mathcal{M}}$, and are termed *parametric faults*.

Definition 4 (Parametric Fault) A parametric fault f is a persistent constant deviation of exactly one parameter $\theta \in \Theta_{\mathcal{M}}$ of the system model \mathcal{M} from its nominal value.

For each parameter, both an increase and a decrease in the parameter value may be considered as a fault. A fault that is an increase in the value of parameter θ is denoted as θ^+ , and a fault that is a decrease is denoted as θ^- .

Example 6 In the circuit, $\Theta_{\mathcal{M}} = \{C_1, C_2, L_1, L_2, R_1, R_2\}$. The complete set of parametric faults is $\{C_1^+, C_1^-, C_2^+, C_2^-, L_1^+, L_1^-, L_2^+, L_2^-, R_1^+, R_1^-, R_2^+, R_2^-\}$.

In the discrete dynamics, faults are represented through component mode changes.

Definition 5 (Discrete Fault) A discrete fault f is a persistent change in the mode of exactly one component $\kappa \in \mathcal{M}$ from its nominal value.

Example 7 In the circuit, there are two switching components, Sw_1 and Sw_2 , and four associated discrete faults $\{\text{Sw}_1^{\text{off}}, \text{Sw}_1^{\text{on}}, \text{Sw}_2^{\text{off}}, \text{Sw}_2^{\text{on}}\}$. Here, the *off* subscript represents the fault where the component changes to the off mode when it should be in the on mode, and the *on* subscript represents the fault where the component changes to the on mode when it should be in the off mode.

8.2.3 Causality

A model is defined without consideration of computational *causality*, i.e., a specification of the computational direction of its constituent constraints. Causality must be considered in order to simulate a model and to study fault propagation.

Given a constraint c , which belongs to a specific mode of a specific component, the notion of a *causal assignment* is used to specify a possible computational direction, or causality, for the constraint c . The causality is indicated by specifying which $v \in V_c$ is the dependent variable in equation ε_c .

Definition 6 (Causal Assignment) A *causal assignment* α_c to a constraint $c = (\varepsilon_c, V_c)$ is a tuple $\alpha_c = (c, v_c^{out})$, where $v_c^{out} \in V_c$ is assigned as the dependent variable in ε_c . We use V_c^{in} to denote the independent variables in the constraint, where $V_c^{in} = V_c - \{v_c^{out}\}$.

In order to assign causality, we must first define which variables within a model are exogenous, i.e., the input variables $U_{\mathcal{M}} \subseteq V_{\mathcal{M}}$. Such variables must always be independent variables in any causal assignment to a constraint involving them, i.e., if a variable v is in $U_{\mathcal{M}}$, then for any constraint c in which $v \in V_c$, it must always be the case that $v \in V_c^{in}$ for any causal assignment.

Parameters that are associated with faults are associated with explicit variables $\Theta_{\mathcal{M}} \in V_{\mathcal{M}}$. They are a special kind of input variable, i.e., $\Theta_{\mathcal{M}} \subseteq U_{\mathcal{M}}$.

In addition, it is useful to refer to a specific set of *output variables*, $Y_{\mathcal{M}} \subseteq V_{\mathcal{M}}$, that are associated with measured outputs of the system.

Example 8 In the circuit, $\Theta_{\mathcal{M}} = \{C_1, C_2, R_1, R_2, L_1, L_2\}$, $U_{\mathcal{M}} = \{u_v\} \cup \Theta_{\mathcal{M}}$, and $Y_{\mathcal{M}} = \{i_3^*, v_8^*, i_{11}^*\}$.

In general, the set of possible causal assignments for a constraint c is as big as V_c , because each variable in V_c can act as v_c^{out} . However, in some cases some causal assignments may not be possible, e.g., if V_c contains any input variables, or if there are noninvertible nonlinear constraints. Also, assuming integral causality, then state variables must always be computed via integration, and so the derivative causality is not allowed. To denote this concept, \mathbb{A}_c refers to the set of permissible causal assignments of a constraint c . For example, for $u \in U_{\mathcal{M}}$, if $u \in V_c$ for some constraint c , (ε_c, u) will never be in \mathbb{A}_c .

For model \mathcal{M} in mode \mathbf{m} , $\mathcal{A}_{\mathcal{M}}^{\mathbf{m}}$ denotes a *complete* set of causal assignments, i.e., for every $c \in C_{\mathcal{M}}^{\mathbf{m}}$, there is exactly one corresponding $\alpha_c \in \mathcal{A}_{\mathcal{M}}^{\mathbf{m}}$. However, only some $\mathcal{A}_{\mathcal{M}}^{\mathbf{m}}$ are actually valid, and this is expressed through the notion of consistency:

Definition 7 (Consistent Causal Assignments) For model \mathcal{M} in mode \mathbf{m} , $\mathcal{A}_{\mathcal{M}}^{\mathbf{m}}$ is *consistent* if

- (i) for every $c \in C_{\mathcal{M}}^{\mathbf{m}}$, $\alpha_c \in \mathbb{A}_c$, i.e., the causal assignment must be permissible;
- (ii) for all $v \in V_{\mathcal{M}} - U_{\mathcal{M}}$, $\mathcal{A}_{\mathcal{M}}^{\mathbf{m}}$ contains exactly one $\alpha = (c, v)$, i.e., every variable that is not an input or parameter is computed by only one (causal) constraint.

Algorithm 1: $\mathcal{A} \leftarrow \text{AssignCausality}(\mathcal{M}, \mathbf{m}, \mathbb{A})$

```

1:  $\mathcal{A} \leftarrow \emptyset$ 
2:  $V \leftarrow U_{\mathcal{M}} \cup \Theta_{\mathcal{M}}$ 
3:  $Q \leftarrow U_{\mathcal{M}} \cup \Theta_{\mathcal{M}} \cup Y_{\mathcal{M}}$ 
4: for all  $c \in C_{\mathcal{M}}^{\mathbf{m}}$  do
5:   if  $|\mathbb{A}_c| = 1$  then
6:      $(c, v) \leftarrow \mathbb{A}_c(1)$ 
7:      $Q \leftarrow Q \cup v$ 
8:   while  $|Q| > 0$  do
9:      $v \leftarrow \text{pop}(Q)$ 
10:    for all  $c \in C_{\mathcal{M}}^{\mathbf{m}}(v)$  do
11:      if  $c \notin \{c : (c, v) \in \mathcal{A}\}$  then
12:         $\alpha^* \leftarrow \emptyset$ 
13:        for all  $\alpha \in \mathbb{A}_c$  do
14:          if  $V_c - \{v_{\alpha^*}\} \cup V \neq \emptyset$  then
15:             $\alpha^* \leftarrow \alpha$ 
16:          else if  $\alpha_v \in Y$  then
17:             $\alpha^* \leftarrow \alpha$ 
18:          else if  $v_{\alpha^*} = v$  and  $|C_{\mathcal{M}}^{\mathbf{m}}(v)| - |\{c' : (c', v) \in \mathcal{A} \wedge v \in v_{c'}\}| = 1$  then
19:             $\alpha^* \leftarrow \alpha$ 
20:        if  $\alpha^* \neq \emptyset$  then
21:           $\mathcal{A} \leftarrow \mathcal{A} \cup \{\alpha^*\}$ 
22:           $Q \leftarrow Q \cup (V_c - V)$ 
23:           $V \leftarrow V \cup \{v_{\alpha^*}\}$ 

```

Algorithm 1 describes the causality assignment process for a model given a mode. Causal assignment works by propagating causal restrictions throughout the model. The process starts at inputs, which must always be independent variables in constraints; and outputs, which must always be the dependent variables in at least one constraint. From these variables, we should be able to propagate throughout the model and compute a valid causal assignment for the model in the given mode. For the purposes of this paper, we assume integral causality and that the model possesses no algebraic loops.² In this case, there is only one valid causal assignment.

Specifically, the algorithm works as follows. It keeps queue of variables to propagate causality restrictions, Q , and a set of variables that are computed in the current causality, V . Initially, V is set to U , because these variables are not to be computed by any constraint. Q is set to U and Y , since the causality of constraints is restricted to U variables being independent variables and Y variables being dependent variables. We add also to Q any variables involved in constraints that have only one permissible causal assignment, because this will also restrict other causal assignments. The set of causal assignments is maintained in \mathcal{A} .

The algorithm goes through the queue, inspecting variables. For a given variable, we obtain all constraints it is involved in, and for each one that does not yet have a causal assignment (in \mathcal{A}), we go through all permissible causal assignments, and

²If algebraic loops exist, the algorithm will terminate before all constraints have been assigned a causality. Extending the algorithm to handle algebraic loops is similar to that for bond graphs; a constraint without a causality assignment is assigned one arbitrarily, and then effects of this assignment are propagated until nothing more is forced. This process repeats until all constraints have been assigned causality.

determine if the causality is forced into one particular causal assignment, α^* . If so, we assign that causality and propagate by adding the involved variables to the queue. A causal assignment $\alpha = (c, v)$ is forced in one of three cases: (i) v is in Y , (ii) all variables other than v of the constraint are already in V , and (iii) v is not yet in V , and all but one of the constraints involving v have an assigned causality, in which case no constraint is computing v and there is only one remaining constraint that must compute v . The algorithm must visit all constraints and does not backtrack, so the time complexity is linear in the size of the model.

Example 9 Consider the mode $\mathbf{m} = [1\ 2]$. Here, $\mathcal{A}^{[1\ 2]}$ is given in column 4 of Table 8.1, denoted by the v_c^{out} in the causal assignment. In this mode, the first switch is off, so i_1 and i_2 act as inputs. Given the integral causality assumption, a unique causal assignment to the model exists and is specified in the column.

Example 10 Consider the mode $\mathbf{m} = [2\ 1]$. Here, $\mathcal{A}^{[2\ 1]}$ is given in column 8 of Table 8.1. In this mode, the second switch is off, so i_9 , i_{10} , and i_{11} act as inputs. Given the integral causality assumption, a unique causal assignment to the model exists and is specified in the column. Note that some causal assignments are in the same as in $\mathbf{m} = [1\ 2]$, while others are different.

When the system mode changes, causality can be recomputed using Algorithm 1. More efficient, incremental, causality assignment, based on the assignment of the previous mode, can also be performed [14], however that is beyond the scope of this chapter.

8.2.4 Structural Model Decomposition

Structural model decomposition creates local submodels given a system model. For a hybrid system, the mode of the system must be specified. When the mode changes, the derived submodels may also change, i.e., if they include constraints of components that have changed mode. This section describes how submodels are generated. Their application to diagnosis will be described in Sect. 8.4.

The procedure for generating a submodel from a causal model is given as Algorithm 2 [13]. The following applies to a given mode, so in the remainder of the section the mode superscript is dropped. Given a causal model \mathcal{M} , and an output variable to be computed y , the `GenerateSubmodel` algorithm derives a causal submodel \mathcal{M}_i that computes y using as local inputs only variables from $U^* = U \cup (Y - \{y\})$. We briefly summarize the algorithm below.

In Algorithm 2, the *variables* queue represents the set of variables that have been added to the submodel but have not yet been resolved, i.e., they cannot yet be computed by the submodel. This queue is initialized to $\{y\}$, and the algorithm then iterates until this queue has been emptied, i.e., the submodel can compute y using only variables in U^* . For each variable v that must be resolved, we use Subroutine 3

Algorithm 2: $\mathcal{M}_i = \text{GenerateSubmodel}(\mathcal{M}, U^*, V^*)$

```

1:  $V_i \leftarrow V^*$ 
2:  $C_i \leftarrow \emptyset$ 
3:  $\mathcal{A}_i \leftarrow \emptyset$ 
4:  $\text{variables} \leftarrow V^*$ 
5: while  $\text{variables} \neq \emptyset$  do
6:    $v \leftarrow \text{pop}(\text{variables})$ 
7:    $c \leftarrow \text{GetBestConstraint}(v, V_i, U^*, \mathcal{A})$ 
8:    $C_i \leftarrow C_i \cup \{c\}$ 
9:    $\mathcal{A}_i \leftarrow \mathcal{A}_i \cup \{(c, v)\}$ 
10:  for all  $v' \in V_c$  do
11:    if  $v' \notin V_i$  and  $v' \notin \Theta$  and  $v' \notin U^*$  then
12:       $\text{variables} \leftarrow \text{variables} \cup \{v'\}$ 
13:       $V_i \leftarrow V_i \cup \{v'\}$ 
14:  $\mathcal{M}_i \leftarrow (V_i, C_i, \mathcal{A}_i)$ 

```

Algorithm 3: $c = \text{GetBestConstraint}(v, V_i, U^*, \mathcal{A})$

```

1:  $C \leftarrow \emptyset$ 
2:  $c_v \leftarrow \text{find } c \text{ where } (c, v) \in \mathcal{A}$ 
3: if  $V_{c_v} \subseteq V_i \cup U^*$  then
4:    $C \leftarrow C \cup \{c_v\}$ 
5: for all  $y \in Y \cap U^*$  do
6:    $c_y \leftarrow \text{find } c \text{ where } (c, y) \in \mathcal{A}$ 
7:   if  $v \in V_{c_y}$  and  $V_{c_y} \subseteq V_i \cup U^* \cup \Theta$  then
8:      $C \leftarrow C \cup \{c_y\}$ 
9: for all  $y \in Y \cap U^*$  do
10:   $c_y \leftarrow \text{find } c \text{ where } (c, y) \in \mathcal{A}$ 
11:   $V' \leftarrow V_{c_y} - \{y\}$ 
12:  for all  $v' \in V'$  do
13:     $c_{v'} \leftarrow \text{find } c \text{ where } (c, v') \in \mathcal{A}$ 
14:    if  $v \in V_{c_{v'}}$  and  $V_{c_{v'}} \subseteq \{v\} \cup U^* \cup \Theta$  then
15:       $C \leftarrow C \cup \{c_{v'}\}$ 
16: if  $C = \emptyset$  then
17:    $c \leftarrow c_v$ 
18: else if  $c_v \in C$  then
19:    $c \leftarrow c_v$ 
20: else
21:    $C' \leftarrow C$ 
22:   for all  $c_1, c_2 \in C$  where  $c_1 \neq c_2$  do
23:      $y_1 \leftarrow \text{find } y \text{ where } (c_1, y_1) \in \mathcal{A}$ 
24:      $y_2 \leftarrow \text{find } y \text{ where } (c_2, y_2) \in \mathcal{A}$ 
25:     if  $(y_1 \triangleleft y_2) \in P$  then
26:        $C' \leftarrow C' - \{c_1\}$ 
27:    $c \leftarrow \text{first}(C')$ 

```

(GetBestConstraint subroutine) to find the constraint that should be used to resolve v in the minimal way.

The GetBestConstraint subroutine, given as Algorithm 3, (which has been updated from [13]) tries to find a constraint that *completely* resolves the variable, i.e., resolves v without further backward propagation (all other variables involved in the constraint are in $V_i \cup \Theta \cup U^*$). Such a constraint may be the one that computes v in the current causality, if all needed variables are already in the submodel (in V_i) or are available local inputs (in U^*); such a constraint may be one that computes a

measured output $y^* \in U^*$, in which case the causality will be modified such that y^* becomes an input, i.e., the constraint in the new causality will compute v rather than y^* ; or such a constraint may be one that computes some y^* through some v' in an algebraic relation. If no such constraint exists, then the constraint that computes v in the current causal assignment is chosen, and further backward propagation will be necessary. A preferences list, P , is used to break ties if multiple minimal constraints exist to resolve v .

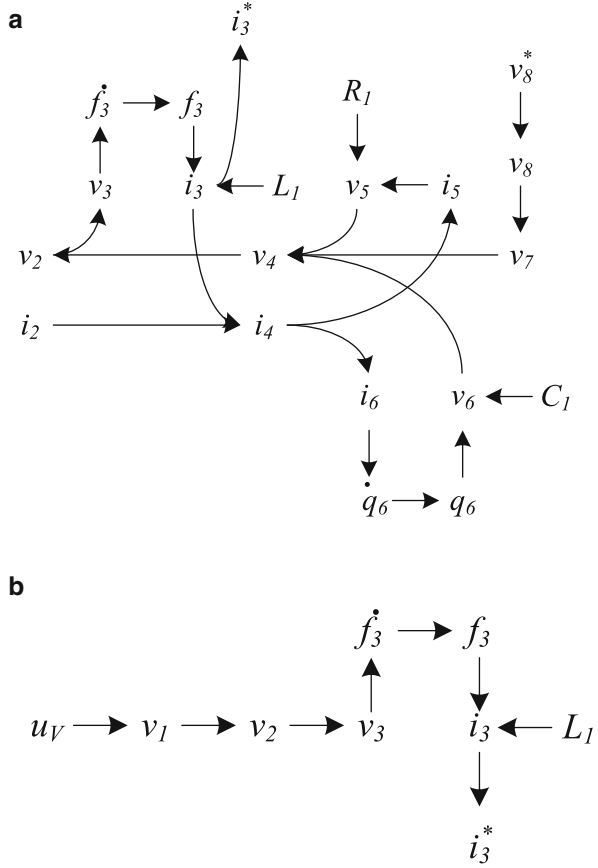
For a given causal model in a given mode, we have the equivalent of a continuous systems model for the purpose of structural model decomposition, and we can compute minimal submodels using the `GenerateSubmodel` algorithm described in previous work [13]. The algorithm finds a submodel, which computes a set of local outputs given a set of local inputs, by searching over the causal model. It starts at the local inputs, and propagates backwards through the causal constraints, finding which constraints and variables must be included in the submodel. When possible, causal constraints are inverted in order to take advantage of local inputs. In the worst case, the algorithm ends up with the global model, so would traverse the entire causal structure. Additional information and the pseudocode are provided in [13].

The local inputs for a submodel are selected from variables for which the values will be known. This includes variables in $U_{\mathcal{M}}$, which are assumed to be known, but could also include variables in $Y_{\mathcal{M}}$, because the values of these are being provided by sensors. On average, `GenerateSubmodel` will find a submodel that is a small subset of the global model. In the worst case, if no decomposition is possible, it will return the global model, minus the other outputs. However, in this case, this submodel is still computationally independent of the others and can still be run in parallel.

Example 11 Submodels can be represented visually using a graph notation, where vertices correspond to variables, and edges correspond to constraints with causal assignments, i.e., a directed edge from v_i to v_j means that v_j is computed using v_i . Consider a submodel for which the local output is i_3^* , and the available local inputs are $\{v_8^*, i_{11}^*\} \cup U_{\mathcal{M}}$. The submodel graphs for two modes are shown in Fig. 8.2. If Sw_1 is on with Sw_2 off, for example, i_3^* can be determined completely by u_V (see Fig. 8.2b). If Sw_1 is off with Sw_2 on, then it must be computed based on the value of v_8^* (see Fig. 8.2a).

The main advantage of structural model decomposition is that faults appear in only a subset of the submodels. Following a model-based approach, in which the submodels are used to compute the nominal system behavior, only a subset of the submodels will be affected by a single fault. Thus, the reasoning becomes much simpler.

Fig. 8.2 Submodel graphs for i_3^* . (a) i_3^* submodel graph in the mode with the first switch off and the second on. (b) i_3^* submodel graph in the mode with the first switch on and the second off



8.3 Problem Formulation

The diagnosis problem is one of mapping observations on a system to an explanation for that set of observations, specifically, which faults may have occurred to produce those observations. In general terms, a fault is a single change in the system. Here, we make the single-fault assumption.

Assumption 1 Only single faults occur in the system.

That is, we assume that only a single change in the system has occurred and can explain the given observations. Since faults are typically unlikely to occur in the first place, when faults are independent of each other, the probability of multiple faults is extremely low. So, the single-fault assumption is common in practical settings.

As described in Sect. 8.2, faults may be either parametric (represented as increases or decreases in system parameter values) or discrete (represented as changes in the modes of components). Further, we assume that faults are persistent, i.e., once a fault occurs, it remains.

Assumption 2 Faults are persistent.

Generally speaking, for the purposes of diagnosis, we consider an observation to be an event observed at a particular time.

Definition 8 (Observation) An *observation* is a tuple (e, t) , where e is an observed event and t is the time of observation.

Two kinds of events are considered, mode changes and fault signatures. We define mode change events specific to components.

Definition 9 (Mode Change Event) An event (κ, m) represents component κ changing to its mode m .

For the purposes of this chapter, we assume these are known/observable, i.e., they are considered an input to our system.

Assumption 3 (Mode Change Observability) All mode change events are observable.

Following a qualitative fault isolation approach, the remaining events take the form of qualitative symbols representing the transients caused by faults, termed fault signatures.³ These symbols are computed from system residuals, i.e., the differences between observed and model-predicted outputs.

Definition 10 (Residual) A residual r for output y as measured by a sensor is computed as $r = y - y^*$, where y^* is the model-predicted output value.

Under the single-fault assumption, a *diagnosis* is simply a fault that is consistent with a given observation sequence.

Definition 11 (Diagnosis) For a system with fault set F , and a sequence of observations O , a *diagnosis* for O , d_O is a fault $f \in F$ that is consistent with O . The set of all diagnoses for O is denoted as D_O .

The diagnosis problem can then be formally defined as follows.

Problem 1 For a system with fault set F , given a finite sequence of observations O , find the set of diagnoses $D_O \subseteq F$.

³Fault signatures will be defined formally in Sect. 8.4.

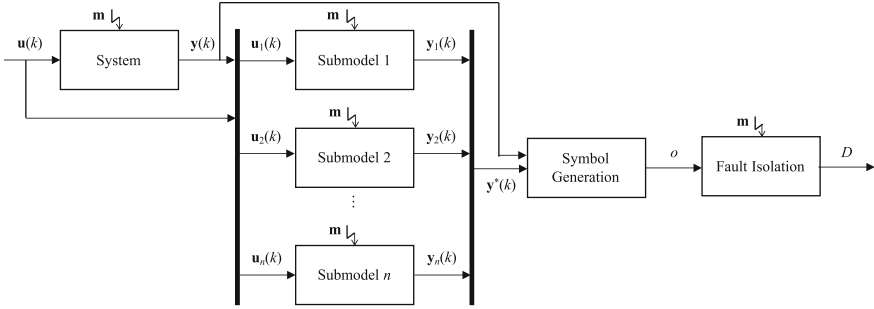


Fig. 8.3 Diagnosis architecture

8.3.1 Architecture

To solve this problem, the architecture shown in Fig. 8.3 is used. The inputs, $\mathbf{u}(k)$, are fed into the system, and the system produces measured outputs, $\mathbf{y}(k)$. These signals are then decomposed into local input and outputs for the local submodels. For each sensor, one submodel is defined where the corresponding output variable is the single local output of the submodel, and all other output variables may serve as local inputs along with $U_{\mathcal{M}}$. Thus, for a set of n sensors, there will be n submodels. The mode change \mathbf{m} is also fed into the system and the submodels.

If a mode change occurs, the system changes mode and the submodels also change modes to reflect the new system mode. This requires regenerating the submodels due to the change in causality, and this can be done efficiently using the causality reassignment algorithm.

The actual system outputs, $\mathbf{y}(k)$, along with the submodel-generated outputs, $\mathbf{y}^*(k)$ are then fed to the symbol generation module. Following a qualitative fault isolation approach, the residuals are transformed into qualitative 0 (no change), - (decrease), and + (increase) changes for the magnitude and slope in the residual. Once a residual is detected to deviate in a statistically significant manner from zero, symbols are generated for that residual, and fed into the fault isolation module.

The fault isolation module reasons over the sequence of observations, consisting of these qualitative symbols and mode change events, to isolate the fault. The algorithms underlying the fault isolation module are described in the following section.

8.4 Qualitative Fault Isolation for Hybrid Systems

As described in Sect. 8.3, the diagnosis problem is to map a sequence of fault signatures and mode change events to single faults that are consistent with the sequence. At the core of the qualitative fault isolation approach is the concept of a

fault signature. In this section, we first describe fault signatures. The fault isolation procedure is then described, followed by a discussion of scalability.

8.4.1 Fault Signatures

The basis of the qualitative fault isolation approach is the concept of a fault signature [10].

Definition 12 (Fault Signature) A *fault signature* for a fault f and residual r in mode m , denoted by $\sigma_{f,r,m}$ is a set of symbols representing changes in r caused by f at the point of the occurrence of f in mode m . The set of all fault signatures for a fault f over residuals R in mode m is denoted as $\Sigma_{f,R,m}$.

In this work, fault signatures are made up of a set of two symbols: the qualitative change in residual magnitude, and the qualitative change in residual slope. Each one of these symbols can take the values + (increase), - (decrease), and 0 (no change). These symbols are based on the transient that is produced when a fault occurs [9]. We write always the magnitude symbol followed by the slope symbol, e.g., a signature +- represents an increase in magnitude and a decrease in slope.

A fault signature provides a prediction of the observation that will be made for a system in a particular mode when that fault happens. For the case of a parametric fault, this is a straightforward concept and we refer the reader to previous works [24]. For discrete faults, the interpretation of the fault signature remains the same, although a discrete fault will change the mode. Specifically, if the system is in mode m and a discrete fault f occurs (thus changing the mode), the signatures in $\Sigma_{f,R,m}$ will be those observations predicted for the fault occurring in mode m , and not the mode in which the fault drives the system into. So, if we know the system is in mode m and fault signatures are observed, we always look in $\Sigma_{f,R,m}$ for every $f \in F$ to reason about which fault has occurred.

Example 12 Table 8.2 shows the fault signatures for the circuit example for the two modes considered for the local submodel residuals. Consider that fault L_1^- occurs in the system. In $\mathbf{m} = [2 \ 1]$, it affects only the residual for i_3^* , as it is the only local submodel where it appears (see Table 8.2). In $\mathbf{m} = [2 \ 1]$, it also affects i_3^* . The fault R_1^+ will also, but they can be distinguished by the specific change produced by the fault. For L_1^- , a +- is produced, whereas for R_1^+ , a 0- is produced.

Fault signatures can be derived from analysis of the system model [9, 19] or via simulation. Here, we assume they are given as input.

Since we have a single submodel for each residual, fault isolation within a single mode is straightforward. Given an observed sequence of fault signatures, Σ in mode m , we determine which faults match all signatures in Σ .

Example 13 For the circuit, given $\mathbf{m} = [1 \ 2]$ and $\Sigma = \{r_{v_8}^{+-}\}$, then $D = \{C_1^-, L_2^-\}$. Note that the * symbol may match either + or -.

For the purposes of this paper, we assume that signatures are correctly observed.⁴

Assumption 4 (Correct Observation) If a fault f occurs in mode m , then if the system does not change mode after the occurrence of the fault, the observed fault signatures will belong to $\Sigma_{f,R,m}$.

8.4.2 Hybrid Systems Diagnosis

For hybrid systems, fault signatures are always given as a function of the system mode. If there is no mode change occurring between the point of fault occurrence and the diagnosis of the fault, then the problem reduces to the continuous systems case. Otherwise, some combination of fault signatures from different modes may be observed, depending on when the mode changes take place and how long it takes for fault signatures to manifest.

Example 14 Consider the residuals in Table 8.2. Assume that the system starts in $\mathbf{m} = [1 \ 2]$ and R_1^+ occurs. Then we could observe $r_{i_3^*}^{0-}$. Now, assume that the system moves to mode $\mathbf{m} = [2 \ 1]$, now we would observe $r_{v_8^*}^{+-}$. This set of fault signatures is not found in any single mode, so the reasoning must extend over the sequence of mode changes.

As shown in the example, the first challenge of the approach for hybrid systems is that now the observed fault signatures may correspond to different modes. Thus, the fault isolation process must span over several potential mode changes. By knowing the mode of the system, we can know which set of fault signatures corresponds to the predicted observations for each fault.

The advantage of structural model decomposition here is that such combinations are limited and easier to deal with compared to an approach using a single global model, where potentially all residuals may be affected by every fault. In that case,

Table 8.2 Fault signatures for minimal submodels of the electrical system

Mode	$\mathbf{m} = [1 \ 2]$			$\mathbf{m} = [2 \ 1]$		
Fault	$r_{i_{11}^*}$	$r_{i_3^*}$	$r_{v_8^*}$	$r_{i_{11}^*}$	$r_{i_3^*}$	$r_{v_8^*}$
C_1^-	00	0+	00	00	00	-+
C_2^-	00	00	-0	00	00	00
L_1^-	00	+−	00	00	+0	00
L_2^-	−0	00	00	00	00	−*
R_1^+	00	0−	00	00	00	+−
R_2^+	00	00	+0	00	00	00

⁴Relaxation of this assumption has been explored for continuous systems in [25].

there are many potential combinations and increased ambiguity. The decoupling of faults and residuals provided by structural model decomposition helps to reduce this complexity.

As discussed in the previous section, in this work we assume that all commanded mode change event are observable. However, even if we know the current mode of the system, there is another related layer of complexity to consider, the *observation delay*, which refers to the delayed observation of fault signatures. The difficulty relies in that the system may be in one mode, but when the observation arrives it might have moved to a different system mode, thus the mode in which the observation was made may not be known exactly.

The observation delay can be manifested in different ways. For example, fault detection in our framework is done by checking if the residual crosses a threshold. Due to the presence of noise and in order to perform a robust fault detection, we use a statistical test that computes the mean of the residual over a small time window and check if that mean has crossed the threshold. In practice, this means that the signal could actually cross the threshold in one mode, but the mean of the signal could cross only in the next mode. Thus, the observation of this signature is delayed. In this work we assume that the observation delay is finite and bounded.

Assumption 5 (Bounded Observation Delay) The delay of any observation is no greater than Δ .

Given our assumptions, the algorithm for a single step of fault isolation for hybrid systems is shown as Algorithm 4. Note that we reason through fault signatures the same for discrete and for parametric faults, hence the algorithm presented is the same for both situations. As inputs, the algorithm takes the current diagnosis, D_i , the previous sequence of fault signatures, λ_i , the new fault signature, σ_{i+1} , and the set of recent modes that falls within $[t - \Delta, t]$, $M_{r,\Delta}$, for the submodel that generates residual r . The main difference of this algorithm against the previous version for continuous systems is that we need to check signatures for each one of the recent modes.

Another advantage here of structural model decomposition is that the set of recent modes is dependent on the model used for fault isolation, and consequently is a function of the residual associated with the signature. If a global model is used, the residual generator will contain all system modes. However, if local submodels are used, the residual generator will only contain the local modes of that submodel (which is always less than the number of system modes). Thus, fewer modes must be searched and efficiency is improved.

Algorithm 4: $D_{i+1} = \text{FaultIsolation}(D_i, \lambda_i, \sigma_{i+1}, M_{r,\Delta})$

```

1:  $D_{i+1} \leftarrow \emptyset$ 
2: for all  $q \in M_{r,\Delta}$  do
3:   for all  $f \in D_i \cap F_{r,q}$  do
4:     if  $\sigma_{i+1} \in \Sigma_{f,r\sigma_{i+1},m}$  then
5:        $D_{i+1} \leftarrow \{f\}$ 

```

If the signature is consistent in any of the modes, it must be added to D_{i+1} . Here, for a given mode m , we need to check only the subset of faults that are included in the current diagnosis and can actually affect this residual in this mode, denoted as $F_{r,m}$. An observed signature, σ_{i+1} , is consistent with a fault if the predicted signature for its residual ($r_{\sigma_{i+1}}$) is included in the signature set for that fault and residual in the given mode.

Algorithm 4 just presents a single reasoning step, when there is a new observed signature, of the fault isolation process. When implemented, this algorithm would be placed within a general progressive monitoring algorithm that keeps track of the current diagnosis, and computes the set of recent modes based on the times events are observed. In the worst case, it must check consistency with all faults and all deviated residuals for all given mode changes, so in the worst case is $O(|F||R||M_r|)$. On average, it is much less, since the candidate set reduces with each newly observed fault signature.

8.4.3 Scalability

The complexity of the fault isolation algorithm is dependent on the number of faults, $|F|$, the number of residuals, $|R|$, and the number of modes, $|M|$. For the global model case, all faults, residuals, and modes in $M_{r,\Delta}$ must be searched. Because r is computed using the global model, it is a function of the system-level mode. For an n -tank system, there are $n - 1$ switching components and so 2^{n-1} system-level modes. Clearly, diagnosis in this case will not scale.

For the local submodel case, each residual is generated by a minimal submodel, so it improves over the global model approach by simultaneously reducing both the effective $|R|$ and the effective $|M|$. The effective $|R|$ is decreased because with structural model decomposition each fault affects only a subset of the residuals, so for each new residual deviation only a subset of faults needs to be checked for consistency. The effective $|M|$ is reduced because with structural model decomposition each residual is reconfigured only based on a few local component modes, whereas for the global model each residual is dependent on the system-level modes (which increases exponentially with the number of switching components). Due to these properties of structural model decomposition, the complexity grows at a significantly smaller rate as the system size increases than with the global model approach.

8.5 Case Study

The Advanced Diagnostics and Prognostics Testbed (ADAPT) is an electrical power distribution system that was built to mimic the operation of such systems on spacecraft [21]. Through the International Diagnostic Competition (DXC), it has

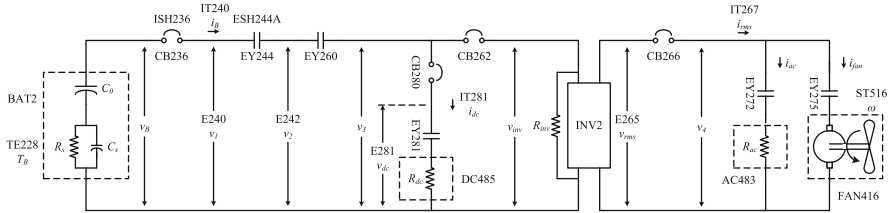


Fig. 8.4 ADAPT-Lite schematic

Table 8.3 Components of ADAPT-Lite and their failure modes

Components	Failure mode
AC483, DC485	Resistance offset
E240, E242, E265, E281, TE228, IT240, IT267, IT281, ST516	Offset
ESH244A, ISH236	Stuck
EY244, EY260, EY272, EY275, EY284	Stuck open
	Stuck closed
FAN416	Underspeed
	Overspeed

been established as a diagnostic benchmark system [26–28]. The diagnosis approach is applied to a subset of ADAPT, called ADAPT-Lite.

Figure 8.4 provides a system schematic for ADAPT-Lite. A battery (BAT2) supplies electrical power to several loads, passing through several circuit breakers (CB236, CB262, CB266, and CB280), and controlled by relays (EY244, EY260, EY281, EY272, and EY275). An inverter (INV2) converts dc to ac power. ADAPT-Lite has one dc load (DC485) and two ac loads (AC483 and FAN416). Sensors report electrical voltage (names beginning with “E”), electrical current (“IT”), and the positions of relays and circuit breakers (“ESH” and “ISH”, respectively). There is one sensor to report the operating state of a load (fan speed, ST516) and another to report the battery temperature (TE228).

Table 8.3 summarizes component fault modes in ADAPT-Lite. The resistance faults, fan speed faults, and sensor faults are modeled as parametric faults, and the remaining faults are modeled as discrete faults.

8.5.1 System Modeling

Following the component-based modeling approach outlined in Sect. 8.2, each component of the system is represented as a set of modes and constraints for each mode. The components are one of the following types: battery, relay, inverter, dc load, ac load, fan, and sensor. Since no faults are considered for the circuit breakers,

they are omitted from the model. In the following, the component models are each described in turn.

BAT2 consists of two 12 V lead-acid batteries in series, which are lumped together into a single battery model. A simplified electrical circuit equivalent model, consisting of a single large capacitance, C_0 , in series with a capacitor-resistor pair, C_s and R_s , that subtracts from the voltage provided by C_0 (see Fig. 8.4), is used. The battery may then be described as

$$\dot{v}_0 = \frac{1}{C_0} (-i_B), \quad (8.1)$$

$$v_0 = \int_{t_0}^t \dot{v}_0 dt, \quad (8.2)$$

$$\dot{v}_s = \frac{1}{C_s} (i_B R_s - v_s), \quad (8.3)$$

$$v_s = \int_{t_0}^t \dot{v}_s dt, \quad (8.4)$$

$$v_B = v_0 - v_s, \quad (8.5)$$

where i_B is the battery current, v_B is the battery voltage, v_0 is the voltage across C_0 , and v_s is the voltage drop across C_s and R_s . The battery temperature is assumed constant, i.e.,

$$\dot{T}_B = 0, \quad (8.6)$$

$$T_B = \int_{t_0}^t \dot{T}_B dt. \quad (8.7)$$

The relays each have two modes, on and off. When off, the constraints are:

$$i_l = 0 \quad (8.8)$$

$$v_r = 0 \quad (8.9)$$

$$p = 0, \quad (8.10)$$

where i_l is the current on the left side of the relay, v_r is the voltage on the right side, and p is the position. When on, constraints are:

$$i_l = i_r \quad (8.11)$$

$$v_l = v_r \quad (8.12)$$

$$p = 1, \quad (8.13)$$

where i_r is the current on the right side, and v_l is the voltage on the left side. When on, the voltages and currents on either side of the relay must be equal. When off, the current on the left side is set to zero, while the current on the right is determined by the component on the right. Similarly, the voltage on the right is set to zero, while the voltage on the left is determined by the component on the left. Following this modeling convention, voltage is always determined by the component on the left, and current by the component on the right, no matter which mode the system is in, and with consistent causality assignment in each mode.

The dc load is a simple resistance:

$$v_{dc} = i_{dc} \cdot R_{dc}, \quad (8.14)$$

where v_{dc} is the voltage across the load, i_{dc} is the current through the load, and R_{dc} is the load resistance. Similarly, the ac load is also a simple resistance:

$$v_{ac} = i_{ac} \cdot R_{ac}, \quad (8.15)$$

however the corresponding voltage and current are rms values.

The fan current is a function of the applied voltage:

$$v_{fan} = i_{fan} \cdot R_{fan}, \quad (8.16)$$

where R_{fan} is the magnitude of the fan impedance, and v_{fan} and i_{fan} are the rms voltage and current, respectively. The fan speed is expressed as a function of its current

$$\dot{\omega} = \frac{1}{J_{fan}} (i_{fan} \cdot g_{fan} - \omega), \quad (8.17)$$

$$\omega = \int_{t_0}^t \dot{\omega} dt, \quad (8.18)$$

where J_{fan} is an inertia parameter and g_{fan} is a gain parameter.

The inverter transforms dc power to ac power. When operating nominally, the rms voltage v_{rms} is controlled very close to 120 V ac as long as the input voltage is above 18 V:

$$v_{rms} = 120 \cdot (v_{inv} > 18). \quad (8.19)$$

From a power balance of the ac and dc sides of the inverter, it results that $v_{inv} \cdot i_{inv} = e \cdot v_{rms} \cdot i_{rms}$, where e is the inverter efficiency, i_{rms} is the inverter rms current, v_{inv} is the inverter voltage on the dc side, and i_{inv} is the input dc current to the inverter. The inverter still draws a small amount of current even when $i_{rms} = 0$, and this is captured as a dc resistance parallel to the inverter, R_{inv} . Hence, the following equation is derived:

$$i_{\text{inv}} = \frac{v_{\text{rms}} \cdot i_{\text{rms}}}{e \cdot v_{\text{inv}}} + \frac{v_{\text{inv}}}{R_{\text{inv}}}. \quad (8.20)$$

The sensors are modeled using a bias term. For sensor s , the constraint is:

$$y_s = s_s + b_s, \quad (8.21)$$

where s_s is the raw signal value, b_s is the bias term, and y_s is the sensor output.

Component models are instantiated and connected according to Fig. 8.4 through series and parallel connections. With five relays, each with two modes, there are a total of $2^5 = 32$ system modes.

The three loads (dc, ac, and fan) each have a single parametric fault, represented through their respective resistance parameters. Further, each of the eleven sensors has an offset fault represented as a change in the bias parameter. Discrete faults are also associated with each of the five relays. A relay can turn on/off without a command, or fail to turn on/off in response to a command. So for a single system mode, there are 19 potential faults that may occur.

8.5.2 Structural Model Decomposition

As described in Sect. 8.3, one submodel is defined for each sensor. In general, there may be a different submodel for each system-level mode, however, many of these submodels are the same for different system-level modes, because the behavior of many of the switching components isolated from a given submodel due to the decomposition.

Example 15 Consider the submodel for E281. It has only two modes: one in which the voltage is determined by the measured value of IT281 (and EY281 is on), and one in which is set to zero (and EY281 is off). Its mode depends only on the state of relay EY281. When off, the submodel consists of the following constraints:

$$\begin{aligned} v_{r,\text{EY260}} &= 0 \\ v_{l,\text{P1}} &= v_{r,\text{EY260}} \\ v_{r,2,\text{P1}} &= v_{l,\text{P1}} \\ v_{l,\text{CB280}} &= v_{r,2,\text{P1}} \\ v_{r,\text{CB280}} &= v_{l,\text{CB280}} \\ v_{l,\text{EY284}} &= v_{r,\text{CB280}} \\ s_{\text{E281}} &= v_{l,\text{EY284}} \\ y_{\text{E281}} &= b_{\text{E281}} + s_{\text{E281}}, \end{aligned}$$

where P1 refers to a parallel connection with the 1 and 2 subscripts indicating the connection. The only local input is b_{E281} , which is assumed to be 0 in the nominal case. When on, the submodel consists of the following constraints:

$$\begin{aligned}
 s_{E242} &= -b_{E242} + y_{E242} \\
 v_{r,EY260} &= s_{E242} \\
 v_{l,P1} &= v_{r,EY260} \\
 v_{r,2,P1} &= v_{l,P1} \\
 v_{l,CB280} &= v_{r,2,P1} \\
 v_{r,CB280} &= v_{l,CB280} \\
 v_{l,EY284} &= v_{r,CB280} \\
 s_{E281} &= v_{l,EY284} \\
 y_{E281} &= b_{E281} + s_{E281}.
 \end{aligned}$$

The local inputs are b_{E281} , which is assumed to be zero, b_{E242} , which is assumed to be zero, and y_{E242} , which is the measured value of E242.

Over the whole system, each submode has between 1 and 4 modes. This greatly simplifies the required diagnostic reasoning, since any given mode change will only affect a minimal number of submodels.

8.5.3 *Diagnosability*

Given any mode of the system, fault signatures can be derived from any fault that may occur within that mode through the qualitative fault propagation algorithms. Signatures for the mode in which all relays are on are shown in Table 8.4. For space, only the signature for parametric faults increasing in the positive direction is shown (for the negative direction, the signature signs are flipped). For example, a bias fault in E265 will produce a deviation in the residual for E265, along with those for IT240, IT267, and ST516, because the value of E265 is used as a local input in the submodels for those sensors. A fault in the resistance of AC483, in contrast, will produce a change only in the residual of IT267, since it appears only in the submodel for IT267. Note that a * symbol is used to denote an indeterminate effect (i.e., the sign of the change depends on the system state).

It is important to note here that some ambiguity in the fault isolation results is expected, i.e., the system is not fully diagnosable. For example, a resistance offset in DC485 will result in a single change in the residual of IT281. A bias in that sensor could result in the same signature, along with a bias in E281 and EY281 turning off. So if the resistance fault occurs, the change in IT281 will be observed, resulting in all

Table 8.4 Fault signatures for faults occurring in the mode with all relays on

Fault	r_{E240}	r_{E242}	r_{E265}	r_{E281}	$r_{ESH244A}$	r_{ISH236}	r_{IT240}	r_{IT267}	r_{IT281}	r_{ST516}	r_{TE228}
$EY244^{off}$	00	—*	00	00	—0	00	—*	00	00	00	00
$EY260^{off}$	00	—*	—*	—*	00	00	—*	00	00	00	00
$EY272^{off}$	00	00	00	00	00	00	00	—*	00	00	00
$EY275^{off}$	00	00	00	00	00	00	00	—*	00	0—	00
$EY281^{off}$	00	00	00	00	00	00	—*	00	—*	00	00
R_{AC483}^+	00	00	00	00	00	00	00	—0	00	00	00
R_{DC485}^+	00	00	00	00	00	00	00	00	—0	00	00
$R_{RFAN416}^+$	00	00	00	00	00	00	00	—0	00	0—	00
b_{E240}^+	+0	—0	00	00	00	00	00	00	00	00	00
b_{E242}^+	00	+0	—0	—0	00	00	*0	00	00	00	00
b_{E265}^+	00	00	+0	00	00	00	*0	*0	00	0*	00
b_{E281}^+	00	00	00	+0	00	00	00	00	*0	00	00
$b_{ESH244A}^+$	00	00	00	00	+0	00	00	00	00	00	00
b_{ISH236}^+	00	00	00	00	00	+0	00	00	00	00	00
b_{IT240}^+	00	00	00	00	00	00	+0	00	00	00	00
b_{IT267}^+	00	00	00	00	00	00	*0	+0	00	00	00
b_{IT281}^+	00	00	00	00	00	00	—0	00	+0	00	00
b_{ST516}^+	00	00	00	00	00	00	00	00	00	+0	00
b_{TE228}^+	00	00	00	00	00	00	00	00	00	00	+0

those faults as diagnoses. No further residuals will deviate to further reduce the fault set. If time limits are set for how long to wait to observe further deviations, then this would improve the diagnosability and allow this fault to be uniquely isolated [24].

In general, the diagnosability results from using residuals from local submodels generated from structural model decomposition may not be the same as using those from a global model, as is proven in [29]. In practice, the diagnosability should be compared to determine if there is any loss of diagnosability from using structural model decomposition.

8.5.4 Results

As an example to illustrate the diagnosis process, consider the initial mode 11100 (here, the mode is designated by the sequence of relay states, for the relays in alphabetical ordering), i.e., EY244, EY260, and EY275 are on, so power is sent only to the ac load. The fault, EY244 turning off uncommanded, occurs at 120.0 s. At 121.0 s, a decrease in the residuals for both y_{E242} and $y_{ESH244A}$ is detected (see Fig. 8.5). Considering the decrease first in y_{E242} , the initial diagnoses are $\{b_{E240}^+, EY260^{off}, EY244^{off}, b_{E242}^-\}$. Considering next the decrease in $y_{ESH244A}$, we can reduce the diagnoses to $EY244^{off}$, which is the true fault; only a fault in EY244

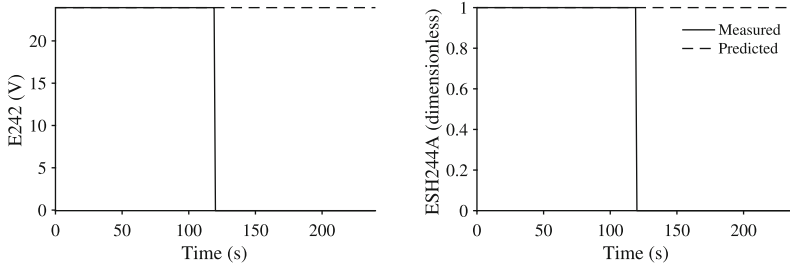


Fig. 8.5 Measured and predicted values for E242 and ESH244A for $EY244^{off}$

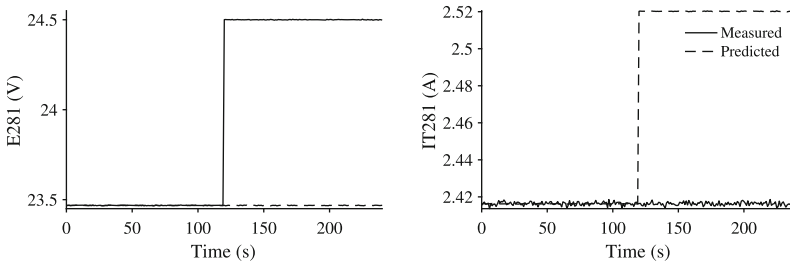


Fig. 8.6 Measured and predicted values for E281 and IT281 for b_{E281}^+

or ESH244A can produce a change in the relay position sensor, so given the previous information from E242, we can discover the true fault. In this case, no mode changes occurred near the time of fault detection, so the reasoning is simplified.

As a second example, consider the initial mode 11001, i.e., EY244, EY260, and EY281 are on, so power is sent only to the dc load. The fault, a positive bias in E281, occurs at 120.0 s. At 121.0 s, an increase in the residual of y_{E281} and decrease in the residual of y_{IT281} are detected (see Fig. 8.6). Considering first the increase in y_{E242} , the initial diagnoses are $\{b_{E281}^+, EY260^{off}\}$. Considering next the decrease in y_{IT281} , we can reduce the diagnoses to $\{b_{E281}^+\}$, which is the true fault. A mode change also occurs at 121.0 s, connecting the fan. This mode change does not change the modes of the submodels of interest so does not affect the reasoning process or produce new signatures.

A comprehensive set of experiments were performed in simulation to validate the approach. The initial system mode, fault, and sequence of mode changes were all selected randomly. For each experiment, we determined whether the true fault was found within the final set of diagnoses, and the diagnostic accuracy, computed as $1/|D|$, where D is the final set of diagnoses.

In 129 total experiments, the true fault was found 97.67% of the time. The average accuracy was 69.49%. Note that 100% accuracy is not expected since the system is not fully diagnosable in all modes, so in some cases there will be ambiguity based on qualitative fault signatures only. For the small percentage of the time in which the true fault was not found, this was due to false positives on some of the fault detectors

for the different residuals. In most of these cases, the true fault is found initially, but then eliminated because a false positive results in a signature inconsistent with the true fault. With additional tuning, the performance could potentially be improved.

Comparing discrete and parametric faults, when a discrete fault was the true fault, it was found 96.00% of the time, with an average accuracy of 67.67%. When a parametric fault was the true fault, it was found 98.08% of the time, with an average accuracy of 69.93%. Thus, performance was about equal for the two different fault types.

8.6 Conclusions

This chapter described a qualitative fault isolation approach for hybrid systems diagnosis. The key features are a compositional modeling approach and the use of structural model decomposition. Structural model decomposition plays a significant role in reducing the complexity of the hybrid systems diagnosis problem, by minimizing the local effects of faults to only a subset of residuals, reducing the number of mode changes to consider, and reducing the effects mode changes will have on the reasoning process.

The approach was demonstrated on a complex electrical power system, considering both parametric and discrete faults. Faults were quickly and correctly diagnosed, with some expected ambiguity due to the use of only qualitative information for diagnosis. This can be followed by quantitative fault identification to uniquely isolate the true fault.

Although only single faults are considered here, and all mode changes (except for faults) are assumed to be observable, the approach can be extended to handle multiple faults in the presence of unobservable mode changes. Preliminary work in this area has been described in the literature [1, 5, 19, 30]. Unobservable mode changes that are not tracked with nominal behavior will result in nonzero residuals, and thus appear as faults. Our framework can be extended to handle this case simply by considering fault signatures associated with these mode changes. As such, these mode changes would be identified.

Acknowledgements Matthew Daigle and Indranil Roychoudhury's work has been partially supported by the NASA SMART-NAS project in the Airspace Operations and Safety Program of the Aeronautics Mission Directorate. Anibal Begon's work has been funded by the Spanish MINECO DPI2013-45414-R grant.

References

1. Narasimhan, S., & Biswas, G. (2007). Model-based diagnosis of hybrid systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 37(3), 348–361.

2. Cocquempot, V., El Mezyani, T., & Staroswiecki, M. (2004). Fault detection and isolation for hybrid systems using structured parity residuals. In *5th Asian Control Conference* (Vol. 2, pp. 1204–1212).
3. Sundström, C., Frisk, E., & Nielsen, L. (2014). Selecting and utilizing sequential residual generators in FDI applied to hybrid vehicles. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *44*(2), 172–185.
4. Koutsoukos, X., Kurien, J., & Zhao, F. (2003). Estimation of distributed hybrid systems using particle filtering methods. In *Hybrid Systems: Computation and Control (HSCC 2003). Lecture notes on computer science* (pp. 298–313). Berlin: Springer.
5. Hofbauer, M. W., & Williams, B. C. (2004). Hybrid estimation of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *34*(5), 2178–2191.
6. Benazera, E., & Travé-Massuyès, L. (2009). Set-theoretic estimation of hybrid system configurations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *39*, 1277–1291.
7. Bayouhdh, M., Travé-Massuyès, L., & Olive, X. (2008). Coupling continuous and discrete event system techniques for hybrid system diagnosability analysis. In *18th European Conference on Artificial Intelligence* (pp. 219–223).
8. Bayouhdh, M., Travé-Massuyès, L., & Olive, X. (2009). Diagnosis of a class of non linear hybrid systems by on-line instantiation of parameterized analytical redundancy relations. In *20th International Workshop on Principles of Diagnosis* (pp. 283–289).
9. Mosterman, P. J., & Biswas, G. (1999). Diagnosis of continuous valued systems in transient operating regions. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, *29*(6), 554–565.
10. Daigle, M. J., Koutsoukos, X., & Biswas, G. (2009). A qualitative event-based approach to continuous systems diagnosis. *IEEE Transactions on Control Systems Technology*, *17*(4), 780–793.
11. Narasimhan, S., & Brownston, L. (2007). HyDE: A general framework for stochastic and hybrid model-based diagnosis. In *Proceedings of the 18th International Workshop on Principles of Diagnosis* (pp. 186–193).
12. Trave-Massuyes, L., & Pons, R. (1997). Causal ordering for multiple mode systems. In *Proceedings of the Eleventh International Workshop on Qualitative Reasoning* (pp. 203–214).
13. Roychoudhury, I., Daigle, M., Bregon, A., & Pulido, B. (2013). A structural model decomposition framework for systems health management. In *Proceedings of the 2013 IEEE Aerospace Conference*.
14. Daigle, M., Bregon, A., & Roychoudhury, I. (2015). A structural model decomposition framework for hybrid systems diagnosis. In *Proceedings of the 26th International Workshop on Principles of Diagnosis*, Paris, France.
15. Bregon, A., Daigle, M., & Roychoudhury, I. (2016). Qualitative fault isolation of hybrid systems: A structural model decomposition-based approach. In *Third European Conference of the PHM Society 2016*.
16. Daigle, M., Bregon, A., & Roychoudhury, I. (2016). A qualitative fault isolation approach for parametric and discrete faults using structural model decomposition. In *Annual Conference of the Prognostics and Health Management Society 2016* (pp. 413–425).
17. Alonso, N. M., Bregon, A., Alonso-González, C. J., & Pulido, B. (2013). A common framework for fault diagnosis of parametric and discrete faults using possible conflicts. In *Advances in artificial intelligence* (pp. 239–249). Berlin: Springer.
18. Bregon, A., Narasimhan, S., Roychoudhury, I., Daigle, M., & Pulido, B. (2013). An efficient model-based diagnosis engine for hybrid systems using structural model decomposition. In *Proceedings of the Annual Conference of the Prognostics and Health Management Society*, 2013.
19. Daigle, M. (2008). *A Qualitative Event-Based Approach to Fault Diagnosis of Hybrid Systems*. PhD thesis, Vanderbilt University.

20. Daigle, M., Koutsoukos, X., & Biswas, G. (2010). An event-based approach to integrated parametric and discrete fault diagnosis in hybrid systems. *Transactions of the Institute of Measurement and Control*, 32(5), 487–510.
21. Poll, S., Patterson-Hine, A., Camisa, J., Garcia, D., Hall, D., Lee, C., et al. (2007). Advanced diagnostics and prognostics testbed. In *Proceedings of the 18th International Workshop on Principles of Diagnosis* (pp. 178–185).
22. Henzinger, T. A. (2000). *The theory of hybrid automata*. Berlin: Springer.
23. Mosterman, P. J., & Biswas, G. (2000). A comprehensive methodology for building hybrid models of physical systems. *Artificial Intelligence*, 121(1–2), 171–209.
24. Daigle, M., Roychoudhury, I., & Bregon A. (2015). Qualitative event-based diagnosis applied to a spacecraft electrical power distribution system. *Control Engineering Practice*, 38, 75–91.
25. Daigle, M., Roychoudhury, I., & Bregon, A. (2014). Qualitative event-based fault isolation under uncertain observations. In *Annual Conference of the Prognostics and Health Management Society 2014* (pp. 347–355).
26. Kurtoglu, T., Narasimhan, S., Poll, S., Garcia, D., Kuhn, L., de Kleer, J., et al. (2009). First international diagnosis competition – DXC’09. In *Proceedings of 20th International Workshop on Principles of Diagnosis* (pp. 383–396).
27. Poll, S., de Kleer, J., Abreau, R., Daigle, M., Feldman, A., Garcia, D., et al. (2011). Third international diagnostics competition – DXC’11. In *Proceedings of the 22nd International Workshop on Principles of Diagnosis* (pp. 267–278).
28. Sweet, A., Feldman, A., Narasimhan, S., Daigle, M., & Poll, S. (2013). Fourth international diagnostic competition – DXC’13. In *Proceedings of the 24th International Workshop on Principles of Diagnosis* (pp. 224–229).
29. Bregon, A., Biswas, G., Pulido, B., Alonso-González, C., & Khorasgani, H. (2014). A common framework for compilation techniques applied to diagnosis of linear dynamic systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(7), 863–876.
30. Daigle, M., Bregon, A., Koutsoukos, X., Biswas, G., & Pulido, B. (2016). A qualitative event-based approach to multiple fault diagnosis in continuous systems using structural model decomposition. *Engineering Applications of Artificial Intelligence*, 53, 190–206.

Chapter 9

Diagnosis of Hybrid Systems Using Hybrid Particle Petri Nets: Theory and Application on a Planetary Rover



Quentin Gaudel, Elodie Chanthery, Pauline Ribot, and Matthew J. Daigle

9.1 Introduction

Real systems have become so complex that it is often impossible for humans to capture and explain their behaviors as a whole, especially when they are exposed to failures. System health management (SHM) or prognostics and health management (PHM) aims at developing tools that can support maintenance and repair tasks, reducing the global costs due to unavailability and repair actions, but also optimizing the mission reward by replanning or reconfiguring the system [37]. An efficient health monitoring technique has to be adopted to determine the health state of the system at any time by using diagnostics and prognostics techniques. A diagnosis method is used to determine the current health state and identify the possible causes of failures that lead to this state by reasoning on observations. Prognosis is used to predict the future health states and the times of the occurrences of the faults that lead to these states. Hybrid systems have been defined by Henzinger [18] as follows.

Definition 9.1 (Hybrid System) A *hybrid system* exhibits both discrete and continuous dynamics.

Sensor data and commands are designated as continuous or discrete observations on the system. Hybrid systems are usually described as a multi-mode system com-

Q. Gaudel
EasyMile SAS, Toulouse, France
e-mail: quentin.gaudel@easymile.com

E. Chanthery (✉) · P. Ribot
LAAS-CNRS, Université de Toulouse, CNRS, INSA, UPS, Toulouse, France
e-mail: elodie.chanthery@laas.fr; pauline.ribot@laas.fr

M. J. Daigle
NIO USA, Inc., San Jose, CA, USA
e-mail: matthew.daigle@nio.io

posed of an underlying discrete-event system (DES) representing the mode changes and various underlying continuous dynamics associated with each mode [3].

Definition 9.2 (Discrete State, Continuous State) The system *discrete state* is defined as the current discrete state of the DES. The evolution of the system *continuous state* depends on continuous dynamics associated with the current system mode.

In most industrial systems, if the degradation is not observable, it is estimated as fault occurrence probabilities. The degradation thus depends on the stress level of the current health mode of the system and, in some cases, also relies on the current continuous state and also on the analysis of the events that occurred on the system [16]. Because of these dependencies and its importance for PHM, we choose to evaluate the degradation separately from the discrete state and the continuous state of the system.

Definition 9.3 (Degradation State) The system *degradation state* is the current value of the degradation whose evolution is represented by degradation dynamics.

We extend the multi-mode system by associating underlying degradation dynamics (e.g. degradation laws) with each mode.

Definition 9.4 (Mode, Event, State) A *mode* is defined as a combination of a discrete state of the DES with continuous dynamics and degradation dynamics. The changes of modes are associated with occurrences of discrete *events*. The *state* of the hybrid system is defined as the combination of its discrete, continuous and degradation states.

Our previous works introduced a framework called *Hybrid Particle Petri Nets* (HPPN). Gaudel et al. [15] proposed to use HPPN to model an uncertain hybrid system and track its current health state by generating a diagnoser. The methodology uses information about the system degradation that is a significant advantage to compute a more accurate diagnosis and to perform prognosis. In [16], we tested the proposed approach on a simulated three-tank system.

This chapter presents in detail the HPPN-based health monitoring method exposed in [16] that has been improved concerning computation performance. The method is hence recalled and new notions are precised, such as the definition of discrete events, the calculation of mode scores or the choice of scale parameters for the diagnoser process. This chapter then exposes results of the implemented health monitoring method on the K11 planetary rover prototype. The K11 is a testbed developed by NASA Ames Research Center which is used for diagnostics and prognostics purposes [1, 7, 9, 37]. A hybrid model of the rover is proposed, based on the discretization of its health evolution. Experimental results are given, illustrating how the methodology is robust to real system data and constraints.

The chapter is organized as follows. Section 9.2 presents related works on diagnosis of hybrid systems. Section 9.3 recalls and deepens the health monitoring methodology based on the modeling of the hybrid system and the generation of a diagnoser in the HPPN framework. Section 9.6 focuses on the application of

the proposed methodology on the K11 planetary rover prototype. It provides the K11 hybrid model and exposes the experimental results and performance metrics. Conclusions and future works are discussed in the final section.

9.2 Related Work

Hybrid systems are a very important subject in many fields, such as modeling, verification, control, and monitoring.

Some models, initially purely continuous have been extended with the integration of events [30]. Similarly, discrete event models have been extended with some continuous aspects, such as Continuous Petri Nets (CPN) [10] and Hybrid Petri Nets (HPN) [12, 44], which introduce a new type of place (continuous place) with a rational marking. Finally, some other hybrid models have been built by the explicit combination of a discrete event model and a continuous model, such as Hybrid Automaton [3, 19] or particle Petri nets [27]. All of these models have been widely used and extended for monitoring hybrid systems.

Zhao et al. [45] propose an approach based on timed Petri nets and mode estimation. The Petri nets use is justified by significant computational advantages over concurrent automata. Fault detection and estimation are done sequentially. Uncertainty about discrete events is not considered. Bayouh et al. [3] model the system with a hybrid automaton. The hybrid system is described as a multi-mode system in which each mode is associated with a continuous dynamic. These works then exploit the analytical redundancy relations of the continuous models and the parity space approach to generate a DES diagnosis that recognizes the signatures associated with each operational modes. Horton et al. [20] introduce fluid stochastic Petri nets. The arcs of a fluid stochastic Petri net carry fluid flow which limits the passage of tokens, and create continuous marking. However, the continuous dynamics is limited to a speed and is not appropriate to represent any hybrid system. Lesire and Tessier [27] combine a discrete event model (Petri nets) and a continuous model (dynamic equations) in an extension of the hybrid Petri nets called Particle Petri Nets (PPN). They propose to distribute the rational marking of the continuous spaces in a set of particles. The tokens of the discrete places (named configurations) and these particles are then used in a monitoring mechanism combining the possibilistic firing [5] with a particle filter to manage all the uncertainties relative to the system and discrete and continuous observations. This work is oriented towards mission monitoring, not health monitoring.

The Modified Particle Petri Nets (MPPN) formalism [46] is proposed as an extension of the PPN. The main advantage of MPPN is that they propose to use transitions associated with conditions that deal with both the configuration and particle values. The application is essentially oriented towards mission monitoring, not health management. The different health states of the system are not considered. Moreover, there is no correspondence with the diagnoser object defined in the literature and the problem of ambiguity in the model is not addressed. The diagnoser

approach was introduced in [34]. The diagnoser is basically a monitor that is able to process any possible observable event that occurs in the system. It consists in recording these observations and providing the set of possible faults whose occurrence is consistent with the observations. However, this approach is restricted to DES and does not manage uncertainty. Some approaches extend the diagnoser to DES modelled by Petri nets. However, none of these approaches take into account continuous aspects, nor consider uncertainty in the system. In [36], an approach for the localization of intermittent faults by dealing with partial observability in the discrete event framework is proposed. The method is based on Petri nets that model the normal functioning of the system observable behavior. A localization mechanism, based on the diagnoser approach, points out the set of events potentially responsible for the faults.

Some studies are particularly focused on the diagnosis of systems with the intention of using it for prognosis purposes. These approaches consider monitoring the degradation of the system. This is called the advanced diagnosis.

In [41], the diagnosis method uses an extended Kalman filter and an Interactive Multiple-Model (IMM) algorithm to monitor both the behavior of the system and its degradation in order to obtain a better system state estimation as a starting point for the prognosis process. However, the approach is limited to continuous systems.

Chanthery and Ribot [6] propose to extend the diagnosis approach proposed in [3] by associating to each mode of the hybrid system a degradation dynamics. However, dynamics are limited to aging laws which estimate the probabilities of occurrences of anticipated faults. The approach does not take into account the uncertainties on the system model and the observations, both for the continuous and discrete part.

In [43], fault isolation is performed dynamically with a Hybrid Bond Graph (HBG). The method proposes to use a fault signature matrix for each mode and introduces a delay to allow each fault to express its symptoms on the residuals (especially the only detectable faults with the continuous signals). However, no indication of the waiting time is given. In parallel with the monitoring of the evolution of these new faults, the degradations of each component depend on the current operational mode and are estimated with a hybrid differential evolution algorithm.

This paper focuses on the application of the health monitoring methodology on the K11 rover, that is subject to inherent uncertainty of real systems.

Uncertainty has been widely studied for state estimation of continuous systems. Concerning hybrid systems, Koutsoukos et al. [24] use a particle filtering technique to estimate the state of a hybrid system modeled as a hybrid automaton. Uncertainty related to discrete events is not taken into account and the system degradation is not considered. In [32], a consistency-based approach combined with particle filters is proposed. Noise and uncertainty are taken into account, but only discrete faults are addressed. Biswas et al. [4] and Wang et al. [42] both propose a robust state estimation and fault diagnosis for uncertain hybrid nonlinear systems, where the discrete dynamics has unknown transition functions. However, they only consider discrete faults. Ru and Hadjicostis [33] use partially observed Petri nets. Partially

observed Petri nets are transformed into an equivalent labelled Petri net and an online monitor is built to diagnose faults and provide beliefs (degrees of confidence) regarding the occurrences of faults. However, this approach is limited because it only takes into account uncertainty in the diagnosis results, not about the model or the event observations. Basile et al. [2] propose to reduce the explosion of the state space by introducing generalized markings (negative tokens) to take into account uncertainty about the firing of transitions. The stochastic Petri nets are used by Jianxiong et al. [21] to build a formal model of each component of an integrated modular avionics architecture. However, for all these approaches, no continuous aspect in the model is taken into account.

In previous works, health monitoring and diagnosis was applied to the K11 rover. In [29], two diagnosis algorithms were applied, Qualitative Event-based Diagnosis (QED) [8], and the Hybrid Diagnosis Engine (HyDE) [31]. QED performs diagnosis based on reasoning over symbols representing qualitative deviations of the sensor signals with respect to model-predicted values. Sensor and process noise are handled by using an observer to estimate the current system state, however no uncertainty in the symbols computed for diagnosis is considered, and all diagnostic hypotheses are viewed as equally likely. HyDE is a consistency-based diagnosis engine that uses hybrid and stochastic models and reasoning. Reasoning is performed by hypothesizing alternative system trajectories inferred from the transition and behavior models of the system, and considers a priori fault probabilities and mode transition probabilities. Both diagnosis algorithms were used to diagnose parasitic load, motor friction, and voltage sensor faults in simulation. In [37], QED diagnosed parasitic load faults and voltage sensor faults in real-world scenarios.

9.3 Health Monitoring Methodology for Hybrid Systems

This section details the methodology proposed in [16] to perform model-based health monitoring of hybrid systems.

We are interested in modeling changes in system dynamics when one or several anticipated faults occur.

Definition 9.5 (Health Modes) The *health modes* are the hybrid system modes (a discrete state with continuous dynamics and degradation dynamics) and represent different health conditions.

Definition 9.6 (Nominal Mode) As long as the system does not encounter any fault, it is in a *nominal mode*.

Definition 9.7 (Degraded Mode) Tracked faults are assumed to be permanent, i.e. once a fault happens, the system moves from a nominal mode to a *degraded mode* or faulty mode.

Definition 9.8 (Failure Mode) Without repair, the system ends in a *failure mode* in which it is not operational anymore.

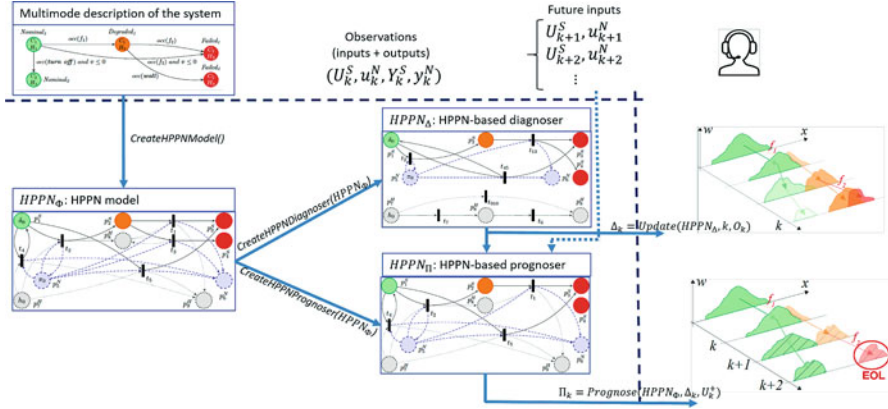


Fig. 9.1 Overview of the health monitoring methodology for hybrid systems

Algorithm 1: HPPN-based monitoring methodology

- 1: $HPPN_{\Phi} \leftarrow CreateHPPNModel()$
- 2: $HPPN_{\Delta} \leftarrow GenerateHPPNDiagnoser(HPPN_{\Phi})$
- 3: **for all** k **do**
- 4: $O_k \leftarrow (U_k^S, u_k^N, Y_k^S, y_k^N)$
- 5: $\Delta_k \leftarrow Update(HPPN_{\Delta}, k, O_k)$
- 6: **end for**

The set of health modes is the superset of nominal, degraded and failure modes.

An overview of the health monitoring method is illustrated in Fig. 9.1 and described by Algorithm 1. Three different objects are defined in the HPPN framework: a hybrid system model $HPPN_{\Phi}$, a HPPN-based diagnoser $HPPN_{\Delta}$ and a HPPN-based prognoser $HPPN_{\Pi}$. Note that the generation of the prognoser object for prognosis purpose is not detailed in this chapter.

The first offline step is the modeling of the hybrid system (line 1) using the HPPN framework, as described in Sect. 9.4. The system model $HPPN_{\Phi}$ can be built either from a multi-mode description of the system or directly from expert knowledge. The second offline step (line 2) is the generation of a HPPN-based diagnoser $HPPN_{\Delta}$ from the system model $HPPN_{\Phi}$ described in Sect. 9.5.2. Then the online diagnoser process (lines 3–6) uses the system consecutive observations O_k (inputs and outputs) to update the diagnoser result and compute the diagnosis Δ_k (see Sect. 9.5).

Example 9.1 Throughout Sect. 9.3, an example of a mobile robot, described in Fig. 9.2, is used to illustrate the definitions and concepts.

The system is described with an oriented graph, in which the nodes represent the health modes and the arcs represent the mode changes. Variables that can be observed or estimated with observations are in bold.

The robot mission is to move without encountering an obstacle or failure, until it reaches a specific area and is turned off. The initial mode is $Nominal_1$: the robot

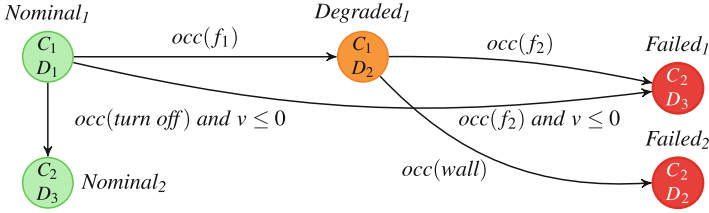


Fig. 9.2 Mobile robot description

is not degraded and is moving in a non-hostile zone. Its velocity v can be estimated with continuous dynamics C_1 and continuous observations, and is positive. Two faults are expected and the robot degradation is estimated as fault occurrence probabilities with degradation dynamics D_1 , in which the probabilities increase with time.

When the (discrete and observable) on-off command *turn off* occurs, the robot stops and its velocity decreases to 0. The robot enters mode *Nominal₂*, where its motor is turned off and its velocity thus stays 0 (continuous dynamics C_2). Because the robot is turned off, the fault occurrence probabilities stagnate, following degradation dynamics D_3 .

Fault f_2 represents a disconnection of the robot motor. Its occurrence leads the system to the failure mode *Failed₁*. The occurrence of f_2 implies the robot stops, so its velocity decreases to 0. Once the motor is disconnected, the robot has the same continuous and degradation dynamics (C_2 and D_3) as if it was turned off.

Fault f_1 represents the robot entrance in a hostile zone where it is degrading faster due to environmental conditions. The robot is still moving at the same velocity (C_1). The physical conditions in mode *Degraded₁* imply that the probability of f_2 increases more significantly than in mode *Nominal₁*. This is defined with degradation dynamics D_2 .

From mode *Degraded₁*, the robot can still enter in mode *Failed₁* with fault f_2 occurrence but it does not match with any condition on the velocity in that case (see arc between *Degraded₁* and *Failed₁*). The velocity estimation is considered less accurate in the hostile zone than in the non-hostile zone, indeed.

Finally, the hostile zone contains obstacles. The robot can encounter a wall, that stops the robot but not its motor. In that case, the mission fails and the robot enters in failure mode *Failed₂*. This event *wall* is not predictable (not estimated with probabilities) but is observable with an environmental on-off sensor. Even if the mission is compromised and the robot is not moving anymore (C_2), its motor is still on so the degradation laws remain the same (D_2).

9.4 Hybrid System Modeling

We propose to model the system by using the *Hybrid Particle Petri Nets* (HPPN) framework introduced in [15].

9.4.1 Hybrid Particle Petri Nets

The HPPN formalism is an extension of Petri nets. Definition 9.9 gives the complete structure of a Hybrid Particle Petri Net before explaining each notation.

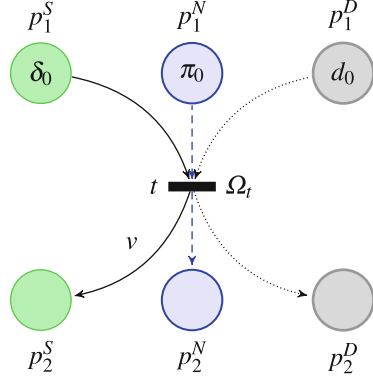
Definition 9.9 A HPPN is defined as a 11-tuple $\langle P, T, A, \mathcal{A}, E, X, D, \mathcal{C}, \mathcal{D}, \Omega, M_0 \rangle$ which describes discrete evolutions (with symbolic places), continuous evolutions (with numerical places) and degradation evolutions (with degradation places) and relations between them:

- P is the set of places, partitioned into numerical places P^N , symbolic places P^S and degradation places P^D , $P = P^S \cup P^N \cup P^D$,
- T is the set of transitions,
- $A \subseteq P \times T \cup T \times P$ is the set of arcs,
- \mathcal{A} is the set of arc annotations,
- E is the set of event labels,
- $X \subseteq \mathbb{R}^{n_N}$ is the state space of the continuous state vector, with $n_N \in \mathbb{N}_+$ the number of continuous state variables,
- $D \subseteq \mathbb{R}^{n_D}$ is the state space of the degradation state vector, with $n_D \in \mathbb{N}_+$ the number of degradation state variables,
- \mathcal{C} is the set of dynamic equation sets associated with numerical places,
- \mathcal{D} is the set of dynamic equation sets associated with degradation places,
- Ω is the set of conditions associated with transitions,
- \mathbb{M}_0 is the initial marking of the Petri net.

An example of a simple HPPN is illustrated in Fig. 9.3. Symbolic places are represented by thin green circles, numerical places are represented by blue circles and degradation places are represented by thick grey circles. Transitions are represented by black lines. Arcs connecting transitions and symbolic places (resp. numerical and degradation places) are represented by plain arrows (resp. discontinuous and dotted arrows).

The set $E = E_o \cup E_{uo}$ is the set of event labels that is partitioned into observable event labels E_o and unobservable event labels E_{uo} . For example, an anticipated fault in the system model is represented by an unobservable event $f \in E_{uo} \subset E$. An event e is a couple $e = (v, k)$ where $v \in E$ is an event label (or type) and $k \in \mathbb{R}$ the time of occurrence of e . For example, $(z, 4)$ means an event type z has occurred at time 4. An event (v, k) is unobservable if for all k , v does not belong to the set of discrete observations of the system.

Fig. 9.3 Example of a simple HPPN at time $k = 0$



The places of a HPPN are marked by tokens that carry different types of information.

Symbolic places P^S model the discrete states of the system and are marked by symbolic tokens called configurations. The set of configurations at time k is denoted M_k^S . Each configuration in a HPPN carries the trace of events that occurred on the system until time k . A configuration $\delta_k \in M_k^S$ is a token at time k whose value is a set of events b_k that occurred on the system until time k : $b_k = \{(v, \kappa) \mid \kappa \leq k\}$.

Numerical places P^N represent continuous dynamics of the system and related uncertainty. Each numerical place $p^N \in P^N$ is associated to a set of dynamic equations $C_{p^N} \in \mathcal{C}$ modeling system continuous dynamic and its corresponding model noise and measurement noise:

$$C_{p^N} = \begin{cases} x_{k+1} = \mathbf{f}(x_k, u_k) + \mathbf{v}(x_k, u_k) \\ y_k = \mathbf{h}(x_k, u_k) + \mathbf{w}(x_k, u_k) \end{cases}, \quad (9.1)$$

where $x_k \in X$ is the continuous state vector, $u_k \in \mathbb{R}^{n_u}$ is the vector of n_u continuous input variables, \mathbf{f} is the noise-free continuous evolution function, \mathbf{v} is a noise function, $y_k \in \mathbb{R}^{n_y}$ is the vector of n_y continuous output variables, \mathbf{h} is the noise-free output function and \mathbf{w} is the noise function associated to observation. Functions \mathbf{f} , \mathbf{v} , \mathbf{h} and \mathbf{w} depend on the considered place p^N . Numerical places are marked by numerical tokens called particles. The set of particles at time k is denoted M_k^N . More precisely, a particle $\pi_k \in M_k^N$ is a token whose value represents a possible continuous state $x_k \in X$ of the system at time k .

Degradation places P^D represent degradation dynamic of the system and related uncertainty. Each degradation place $p^D \in P^D$ is associated with a set of equations $D_{p^D} \in \mathcal{D}$ modeling system degradation dynamic:

$$D_{p^D} = \{ \bar{d}_{k+1} = \mathbf{g}(\bar{d}_k, b_k, x_k, u_k) + \mathbf{z}(\bar{d}_k, b_k, x_k, u_k) \}, \quad (9.2)$$

where $\bar{d}_k \in D$ is the degradation state vector, \mathbf{g} is the noise-free degradation evolution function and \mathbf{z} is a noise function. Functions \mathbf{g} and \mathbf{z} depend on the

considered place p^D . It has to be noticed that, as said earlier, the difference between continuous and degradation places is that degradation system states are function of the continuous state and the set of events b_k that have occurred time k .

Degradation places are marked by degradation tokens. The set of degradation tokens at time k is denoted M_k^D . A degradation token $d_k \in M_k^D$ links a configuration δ_k to a particle π_k and its value is a possible degradation state $d_k \in D$ of the system at time k .

The set of places P of the HPPN is:

$$P = P^S \cup P^N \cup P^D = \{p_1^S, \dots, p_s^S\} \cup \{p_1^N, \dots, p_n^N\} \cup \{p_1^D, \dots, p_d^D\} \quad (9.3)$$

where s , n and d are respectively the number of symbolic, numerical and degradation places. In Fig. 9.3, we have, for example, $P = \{p_1^S, p_2^S, p_1^N, p_2^N, p_1^D, p_2^D\}$.

Let M_k denote the set of tokens of the HPPN at time k :

$$M_k = M_k^S \cup M_k^N \cup M_k^D, \quad (9.4)$$

where M_k^S , M_k^N and M_k^D are respectively the set of configurations, particles and degradation tokens at time k . In Fig. 9.3, we have, for example, $M_0 = \{\delta_0, \pi_0, d_0\}$.

The marking \mathbb{M}_k of a HPPN at time k is the distribution of tokens in the different places:

$$\mathbb{M}_k = \mathbb{M}_k^S \cup \mathbb{M}_k^N \cup \mathbb{M}_k^D, \quad (9.5)$$

where $\mathbb{M}_k^S \in (2^{M_k^S})^s$, $\mathbb{M}_k^N \in (2^{M_k^N})^n$ and $\mathbb{M}_k^D \in (2^{M_k^D})^h$ are respectively symbolic, numerical and degradation markings at time k . For the example illustrated in Fig. 9.3:

$$\begin{aligned} \mathbb{M}_0^S &= [[\delta_0] \emptyset], \\ \mathbb{M}_0^N &= [[\pi_0] \emptyset], \\ \mathbb{M}_0^D &= [[d_0] \emptyset]. \end{aligned}$$

Initial marking \mathbb{M}_0 represents the initial conditions of the system (the initial continuous and degradation states and the set of events that have occurred until time 0).

Definition 9.10 (Hypothesis) A *hypothesis* on the system contains all knowledge about the system state at time k and the events that have occurred on the system until time k . A *hypothesis* $\{\delta_k, \pi_k^1, \dots, \pi_k^{n_k}, d_k^1, \dots, d_k^{n_k}\}$ at time k is composed of a configuration δ_k , a set of particles $\{\pi_k^i | i \in \{1, \dots, n_k\}\}$ and a set of degradation tokens $\{d_k^i | i \in \{1, \dots, n_k\}\}$, where each degradation token d_k^i links the particle π_k^i to the configuration δ_k .

For example, if the event set is b_0 , the continuous state x_0 and the degradation state d_0 are precisely known, the initial set of tokens $M_0 = \{\delta_0, \pi_0, d_0\}$, where d_0 links δ_0 and π_0 , is the unique hypothesis. A hypothesis at time k may contain several

Fig. 9.4 Illustration of particle clusters

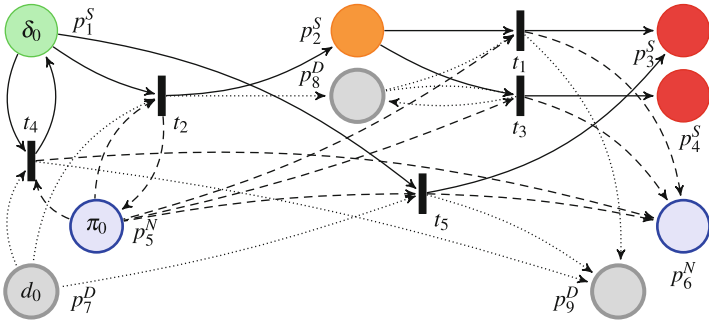
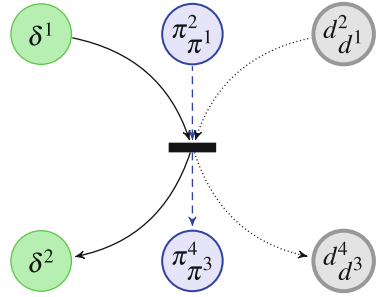


Fig. 9.5 HPPN model of the mobile robot

particles and degradation tokens to represent imprecise knowledge on continuous and degradation states, e.g. $\{\delta_k^1, \pi_k^1, \dots, \pi_k^{n_k}, d_k^1, \dots, d_k^{n_k}\}$, where $n_k \in \mathbb{N}_+$ is the number of particles used to represent the continuous state, and where n_k degradation tokens links n_k particles to the configuration δ_k^1 . The number n_k of particles and degradation tokens is representative of the hypothesis precision at time k .

Definition 9.11 (Particle Cluster) The set of n_k particles linked to the same configuration with n_k degradation tokens is called a *particle cluster*.

In Fig. 9.4, d^1 and d^2 links π^1 and π^2 to δ^1 , and d^3 and d^4 links π^3 and π^4 to δ^2 . Two hypotheses are represented $\{\delta^1, \pi^1, \pi^2, d^1, d^2\}$ and $\{\delta^2, \pi^3, \pi^4, d^3, d^4\}$, with two particle clusters $\{\pi^1, \pi^2\}$ and $\{\pi^3, \pi^4\}$. At each time k , the set of particle clusters defines a partition of the particle set M_k^N of the HPPN.

9.4.2 Illustration Example

The HPPN model of the mobile robot is presented in Fig. 9.5. Symbolic places are represented by places with regular thicknesses, while numerical and degradation places are represented by places with medium and large thicknesses, respectively.

Arcs that connect transitions and symbolic (numerical and degradation) places are represented by solid (dashed and dotted) arrows.

Health modes of a hybrid system (nominal, degraded and failure modes) are represented by combinations of discrete states, continuous dynamics and degradation dynamics. Transitions model changes of health modes, so any transition have three places (one of each type) in its sets of input places and three places in its set of output places. Two transitions cannot have both the same set of input places and the same set of output places.

We decompose the five health modes of the robot into four symbolic places, two numerical places and three degradation places. Four discrete health states are identified from the robot description (Fig. 9.2). One nominal state, one degraded state, and two different failure states are represented by the four symbolic places p_1^S , p_2^S , p_3^S and p_4^S , respectively. The two numerical places p_5^N and p_6^N represent the continuous dynamics C_1 and C_2 . The three degradation places p_7^D , p_8^D and p_9^D represent the degradation dynamics D_1 , D_2 and D_3 , respectively. Five transitions represent the health mode changes. For example, transition t_4 represents the change from mode $Nominal_1$ to mode $Nominal_2$ so ${}^\circ t_4 = \{p_1^S, p_5^N, p_7^D\}$ and $t_4^\circ = \{p_1^S, p_6^N, p_9^D\}$.

The initial mode is $Nominal_1$ so the tokens δ_0 , π_0 and d_0 are in p_1^S , p_5^N and p_7^D , respectively. At time $k = 0$, no event has occurred, $b_0 = \{\}$. The only estimated state is the velocity, $x_0 = [v_0]^T$ with $v_0 > 0$ because the velocity is initially positive. The initial fault occurrence probabilities $\rho_0^{f_1}$ and $\rho_0^{f_2}$ are very low. Thus, $d_0 = [\rho_0^{f_1}, \rho_0^{f_2}]^T$ with $\rho_0^{f_1} = 0.01$ and $\rho_0^{f_2} = 0.05$.

9.4.3 Marking Evolution Rules in HPPN for Diagnosis

Firing rules in a HPPN may be different depending on the utilization for model simulation, diagnosis or prognosis purposes. Semantics of transition firing are proposed here only for diagnosis purpose.

The following assumptions are considered. The set of input places of a transition is composed of at least one place of any type and at most a place of each type. The set of output places of a transition is composed of at least as many places of each type contained in its set of input places.

Let ${}^\circ t$ (resp. t°) denote the set of input (resp. output) places of t . The firing of a transition $t \in T$ depends on its associated condition set $\Omega_t \in \Omega$. This condition set Ω_t contains as many conditions as there are input places in ${}^\circ t$:

$$\forall t \in T, |\Omega_t| = |{}^\circ t|. \quad (9.6)$$

For example, if t has a place of each type in ${}^\circ t$, its condition set is $\Omega_t = \{\omega_t^S, \omega_t^N, \omega_t^D\}$. A condition $\omega : M_k \rightarrow \mathbb{B}$, with $\mathbb{B} = \{\top, \perp\}$ (set of logic values TRUE and FALSE), can be a test on the token value, always satisfied (\top), or never satisfied (\perp).

A symbolic condition ω_i^S can be \top or \perp , or it can test the occurrence of an event $v \in E$ (as fault, mission event, interaction with environment, etc.). In this case, the condition $\omega_i^S(\delta_k) = \text{occ}(b_k, v)$ tests if the event set b_k of the configuration δ_k contains the event (v, k) .

A numerical condition ω_i^N (resp. degradation condition ω_i^D) can be \top or \perp or it can test a constraint on the continuous state (resp. degradation state) of the system. In this case, the condition $\omega_i^N(\pi_k) = c(x_k)$ tests the value of the continuous state vector x_k of the particle π_k .

Example 9.2 For the example of the mobile robot illustrated in Fig. 9.5, condition $\Omega(t_4)(\delta_k, \pi_k, d_k) = \text{occ}(b_k, \text{turn off}) \wedge (x_k^0 \leq 0)$ tests if an event labeled with *turn off* occurred at time k and if v_k is 0. We assume that a fault occurs if its probability of occurrence is greater than a predefined threshold 0.9. Consequently, the condition associated with transition t_2 is $\Omega(t_2)(\delta_k, \pi_k, d_k) = \text{occ}(b_k, f_1) \vee (d_k^0 > 0.9)$. With the same reasoning, we have $\Omega(t_1)(\delta_k, \pi_k, d_k) = \text{occ}(b_k, f_2) \vee (d_k^1 > 0.9)$, $\Omega(t_3)(\delta_k, \pi_k, d_k) = \text{occ}(b_k, f_2) \wedge (x_k^0 \leq 0) \vee (d_k^1 > 0.9)$ and $\Omega(t_5)(\delta_k, \pi_k, d_k) = \text{occ}(b_k, \text{wall})$.

The firing of a transition t at time k is illustrated in Fig. 9.6. A token is accepted by the conditions Ω_t at time k if it satisfies the condition of its type. Let $M_k(p)$ be the set of tokens in the place $p \in P$ at time k . \mathcal{S}_k^t is the set of tokens in the input places of the transition t that are accepted by the conditions Ω_t at time k :

$$\begin{aligned} \mathcal{S}_k^t = & \{ \delta_k \in M_k(p^S) \mid \omega_i^S(\delta_k) = \top \} \cup \\ & \{ \pi_k \in M_k(p^N) \mid \omega_i^N(\pi_k) = \top \} \cup \\ & \{ d_k \in M_k(p^D) \mid \omega_i^D(d_k) = \top \} , \end{aligned} \tag{9.7}$$

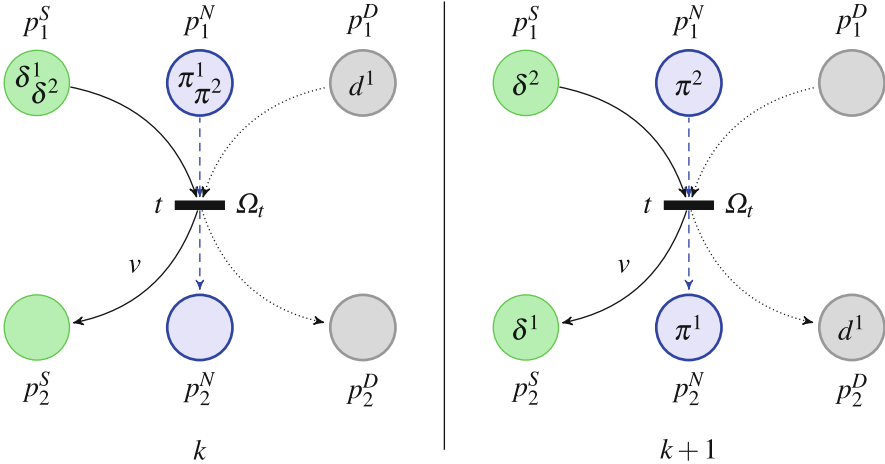


Fig. 9.6 Illustration of a transition firing at time k

where $(p^S, p^N, p^D) \in (P^S \cap \circ t) \times (P^N \cap \circ t) \times (P^D \cap \circ t)$ and $\omega_t^S \in \Omega_t$, $\omega_t^N \in \Omega_t$, $\omega_t^D \in \Omega_t$.

Definition 9.12 (Fireable Transition) A transition $t \in T$ is fireable at time k if it exists at least one token in each of its input places which are accepted by the conditions Ω_t :

$$\forall p \in \circ t, |\mathcal{S}_k^t(p)| > 0. \quad (9.8)$$

In Fig. 9.6, we suppose at time k that $\omega_t^S(\delta^1) = \top$, $\omega_t^N(\pi^1) = \top$, $\omega_t^D(d^1) = \top$, $\omega_t^S(\delta^2) = \perp$ and $\omega_t^N(\pi^2) = \perp$, then the transition t is fireable.

Definition 9.13 (Transition Firing) The firing of a transition $t \in T$ at time k is defined as follows: $\forall P^o \in \{P^S, P^N, P^D\}, p \in P^o \cap \circ t, p' \in P^o \cap t^\circ$,

$$\begin{aligned} M_{k+1}(p) &= M_k(p) \setminus \mathcal{S}_k^t(p), \\ M_{k+1}(p') &= M_k(p') \cup \mathcal{S}_k^t(p), \end{aligned} \quad (9.9)$$

where $\mathcal{S}_k^t(p)$ is the set of tokens of \mathcal{S}_k^t which are in the place p .

In Fig. 9.6, after firing the transition t , the three accepted token are in the output places of t . During the transition firing, accepted tokens are moved, their links are conserved and their values are either conserved or updated. This property is the main difference from ordinary Petri nets in which tokens are consumed and new tokens are created in the output places of the transition. The conservation of token values exists in some extensions of Petri nets, like in colored Petri nets for example but the existence of links between tokens and their conservation during the transition firing is specific to HPPN.

An arc $a \in A$ connecting a transition t to a symbolic place p^S can be annotated with an event label $v \in E$. In this case, the set of event b of a configuration δ which has moved in p^S after the firing of t at time k is updated with the event (v, k) . The values of configurations evolve with the annotations $\mathcal{A} \subset A \times E$ during the firing of transitions. In Fig. 9.6, we suppose that d^1 links δ^1 and π^1 at time k . After the firing of t , the value of δ^1 is $b_{k+1}^1 = b_k^1 \cup (v, k)$, the value of π^1 is $x_{k+1}^1 = x_k^1$, the value of d^1 is $d_{k+1}^1 = d_k^1$, and d^1 still links δ^1 and π^1 .

9.5 Hybrid System Diagnosis

Diagnosis aims at tracking the system current health state. The system health state is the combination of its discrete, continuous and degradation states. We propose to build a diagnoser from the HPPN model of a hybrid system [17]. The HPPN-based diagnoser monitors both the system behavior and degradation under uncertainty. Its online process takes as inputs the set of discrete and continuous observations on the system. The output of the diagnoser process at any time k is an estimation of the system health state that takes the form of the marking of the HPPN-based diagnoser $\Delta_k = \hat{M}_k$.

9.5.1 Uncertainty

Several types of uncertainty are taken into account by using HPPN. Knowledge-based uncertainty must be taken into account because the model does not reflect perfectly reality, as for the symbolic part of the model than the numerical one. Due to the inherent imprecision of sensors, we also consider uncertainty about observations. Two types of uncertainty are then considered: the symbolic uncertainty dealing with the discrete model and observations; and the numerical uncertainty dealing with the imprecision on the continuous model and numerical values.

Regarding the symbolic aspects, the discrete model of the system may include symbolic uncertainty as impossible or incomplete event sequences. Concerning the discrete observations, an event may occur without being observed: this is a missing observation. Dually, an event may be observed whereas it has not really occurred: we talk about false observation.

Symbolic uncertainty is managed at two levels in the HPPN-based diagnoser:

- Every symbolic condition of transitions is replaced by a TRUE condition during the diagnoser generation. It means that pseudo-firing is used for these transitions with modified symbolic conditions.
- During the prediction step of the online diagnoser process, the diagnoser uses pseudo-firing of transitions [27, 46], introduced in [5] to consider the occurrences of each event consistent with the discrete dynamic. Pseudo-firing creates new hypotheses.

Transition pseudo-firing duplicates tokens: tokens in the input places of the transition are not moved but duplicated and their duplicates are moved in the output places of the transition.

Definition 9.14 (Transition Pseudo-Firing) Let $t \in T$ be an enabled transition. For each type of input and output place of t , the pseudo-firing of $t \in T$ at time $k - 1$ is formally defined by: $\forall P^o \in \{P^S, P^N, P^D\}, p \in P^o \cap {}^\circ t, p' \in P^o \cap t^\circ$,

$$\begin{aligned} M_k(p) &= M_{k-1}(p), \\ M_k(p') &= M_{k-1}(p') \cup \mathcal{S}_{k-1}^t(p), \end{aligned} \quad (9.10)$$

where $\mathcal{S}_{k-1}^t(p)$ is the token set of \mathcal{S}_{k-1}^t that are in place p .

Example 9.3 Figure 9.7 illustrates the pseudo-firing of a transition t . At time k , d^1 is supposed to link δ^1 and π^1 and transition t is supposed to be enabled. After pseudo-firing t , tokens δ^1 , π^1 and d^1 are not moved and tokens δ^2 , π^2 and d^2 are created and moved in the output places of t . Moreover, d^1 links δ^1 and π^1 , and d^2 links δ^2 and π^2 .

Besides the intrinsic deviation between reality and continuous model of a system, numerical uncertainty embodies the fact that the numerical values are imprecise. This is an inevitable problem in real case studies. For example, in Fig. 9.8, we can see the difference between the measured data of a battery voltage and its non-noisy discharge model.

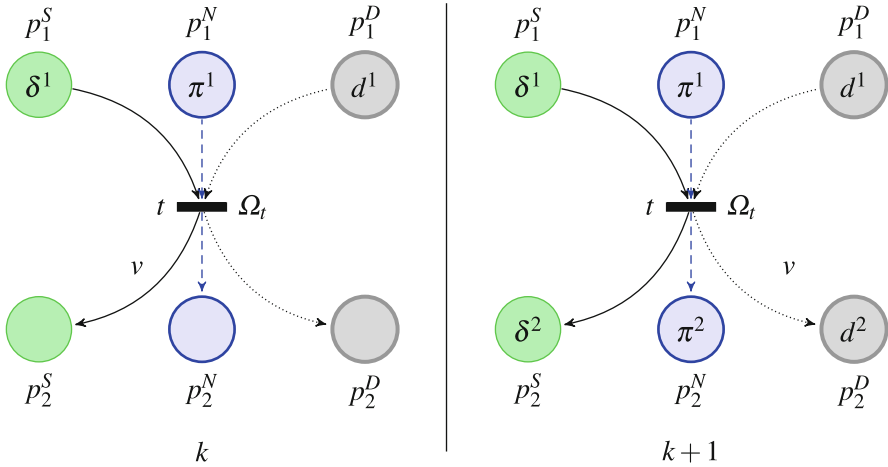


Fig. 9.7 Pseudo-firing of a transition t at time k

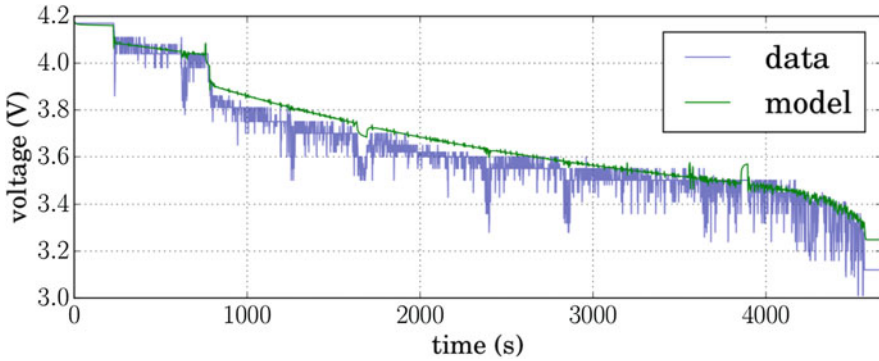


Fig. 9.8 Comparison between measured data of a battery voltage and its non-noisy discharge model

Numerical uncertainty is often dealt with through an estimator [11], that aims at estimating the continuous state according to model noise and measurement noise. We use particle filters [40] to estimate the continuous state through the set of particles of the HPPN. The use of particulate filters is relevant for estimating discrete, continuous and degradation states since the representation of the continuous state estimate is already discretized into particles.

For example, in this work, a particle filter is applied independently to each cluster of particles thanks to the links between the configurations and the particles, provided by the degradation tokens. During the diagnosis prediction step, the values of the particles evolve as a function of the continuous dynamics associated with the numerical places to which the particles belong. Then, during the correcting step of the online diagnoser process, each particle cluster is resampled independently.

The links between the configurations and the particles, provided by the degradation tokens, are thus used to prevent the particle distribution to be disturbed by pseudo-firing.

9.5.2 Diagnoser Generation

The system model HPPN_Φ is a tuple $\langle P_\Phi, T_\Phi, A_\Phi, \mathcal{A}_\Phi, E_\Phi, X_\Phi, D_\Phi, \mathcal{C}_\Phi, \mathcal{D}_\Phi, \Omega_\Phi, -\mathbb{M}_{0\Phi} \rangle$ as defined in Sect. 9.4. The diagnoser HPPN_Δ is a tuple:

$$\text{HPPN}_\Delta = \langle P_\Delta, T_\Delta, A_\Delta, \mathcal{A}_\Delta, E_\Delta, X_\Delta, D_\Delta, \mathcal{C}_\Delta, \mathcal{D}_\Delta, \Omega_\Delta, \mathbb{M}_{0\Delta} \rangle, \quad (9.11)$$

which is generated from the system model HPPN_Φ in six steps that are described hereafter.

The diagnoser has to estimate the discrete, continuous and degradation states of the system. Step 1 consists in copying the HPPN system model. Indeed, discrete, continuous and degradation state spaces, as well as continuous and degradation dynamics are the same as those of the model. As a result, all the places, event labels, state spaces and diagnosis dynamics remain the same as those of the model HPPN_Φ :

$$P_\Delta = P_\Phi, E_\Delta = E_\Phi, X_\Delta = X_\Phi, D_\Delta = D_\Phi, \mathcal{C}_\Delta = \mathcal{C}_\Phi, \mathcal{D}_\Delta = \mathcal{D}_\Phi. \quad (9.12)$$

The initial marking $\mathbb{M}_{0\Delta}$ of the HPPN-based diagnoser HPPN_Δ corresponds to the initial marking $\mathbb{M}_{0\Phi}$ of the system model, which contains knowledge about mode, state and events that occurred on the system at time 0 (usually none): $\mathbb{M}_{0\Delta} = \mathbb{M}_{0\Phi}$.

Step 2 consists in separating the HPPN-based diagnoser into two levels: the *behavioral level* manages the observable part of the system whereas the *degradation level* includes the unobservable part. Thus, the behavioral level contains only the symbolic and numerical places, while the degradation level contains the degradation places.

Each transition $t \in T_\Phi$ thus generates a pair of transitions (t', t'') during the diagnoser generation. Transition t' inherits arcs linking t to symbolic and numerical places, as well as symbolic and numerical conditions. Transition t'' inherits arcs linking t to the degradation places, as well as the degradation condition. t' and t'' are then defined by:

$$\circ t' = \circ t \cap (P^S \cup P^N), \quad t'^\circ = t^\circ \cap (P^S \cup P^N), \quad (9.13)$$

and:

$$\circ t'' = \circ t \cap P^D, \quad t''^\circ = t^\circ \cap P^D, \quad (9.14)$$

with the following conditions $\Omega_{t'}$ and $\Omega_{t''}$:

$$\Omega_{t'} = \langle \omega_t^S, \omega_t^N \rangle, \quad \Omega_{t''} = \langle \omega_t^D \rangle. \quad (9.15)$$

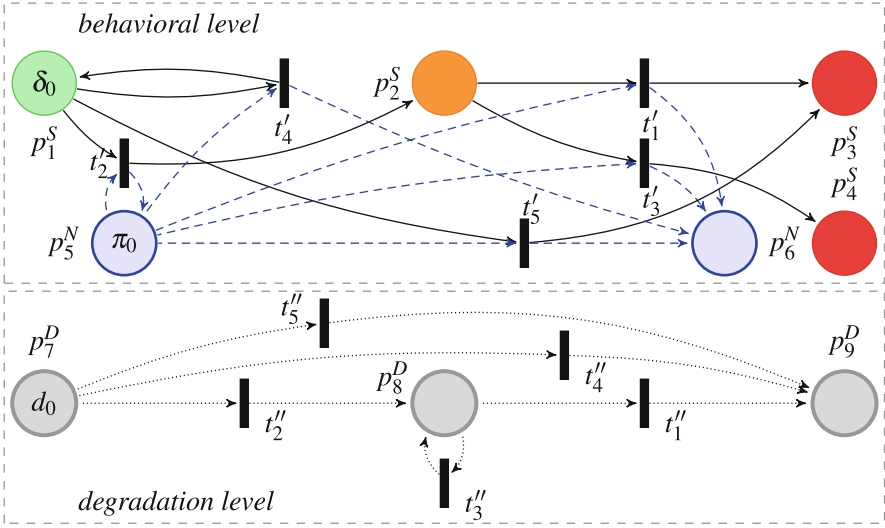


Fig. 9.9 Diagnoser generation of the mobile robot—Step 2: separation into two levels

The set of all transitions in the behavioral level and in the degradation level is denoted T_Δ .

Example 9.4 Figure 9.9 illustrates the second step of the diagnoser generation $HPPN_\Delta$, the separation into two levels, for the example of the mobile robot. Degradation places and associated transitions are put in the degradation level.

Step 3 consists in setting to TRUE all the symbolic conditions, in agreement with the uncertainty management concerning the occurrences of events (see Sect. 9.5.1):

$$\forall t \in T_\Delta, \omega_t^S \in \Omega_t \Rightarrow \omega_t^S \leftarrow \top. \tag{9.16}$$

Thus, all configurations in the HPPN-based diagnoser satisfy the symbolic conditions. This means that the diagnoser considers at any time the occurrence of each event that can occur from the estimated current mode. The arc annotations remain unchanged:

$$\mathcal{A}_\Delta = \mathcal{A}_\Phi. \tag{9.17}$$

Example 9.5 In the example of the mobile robot, conditions $\Omega(t_1) = \Omega(t_2) = \Omega(t_5) = \top$ because of the OR logical expression. Transitions t_3 and t_4 keep their numerical conditions because of the AND logical expression.

Step 4 consists in removing degradation conditions in order to disconnect the marking evolution of the degradation level from the degradation state:

$$\forall t \in T_\Delta, \omega_t^D \in \Omega_t \Rightarrow \Omega_t \leftarrow \Omega_t \setminus \{\omega_t^D\}. \quad (9.18)$$

This step allows to manage computation performance and to keep focus on observations during the diagnosis process.

Example 9.6 In the example of the mobile robot, condition $\Omega(t_3) = \top$ after Step 4.

Step 5 also improves the computation performance by merging transitions having the same sets of input and output places in order. It reduces the size of the possible state space. As a consequence, in the behavioral level, hypotheses sharing the same cluster of particles are created during the prediction step of the online diagnoser. In other words, several hypotheses are monitored according to the same continuous dynamics with a single cluster of particles instead of having as many clusters as hypotheses.

In the degradation level, this step eliminates concurrent transitions which have the same degradation place as input and the same degradation place as output.

Two transitions are mergeable if they represent the same change in continuous dynamics (the same numerical places as input and as output, and the same numerical condition) and have the same symbolic input place.

Definition 9.15 (Mergeable Transitions) Two transitions $(t', t'') \in T^2$ are mergeable if and only if:

$$(\circ t' = \circ t'') \wedge (t'^\circ \cap P^N = t''^\circ \cap P^D) \wedge (t'^\circ \cap P^D = t''^\circ \cap P^N) \wedge (\Omega_{t'} = \Omega_{t''}). \quad (9.19)$$

Step 5 of the diagnoser generation consists in merging every pair of mergeable transitions as long as there are at least two mergeable transitions using the following definition.

Definition 9.16 (Merging of Two Transitions) Merging two mergeable transitions $(t', t'') \in (T)^2$ is defined by:

1. Create a new transition t such as:

$$\circ t \leftarrow \circ t', \quad t^\circ \leftarrow t'^\circ \cup t''^\circ, \quad \Omega_t \leftarrow \Omega_{t'}. \quad (9.20)$$

2. Update T_δ :

$$T_\delta \leftarrow (T_\delta \setminus \{t', t''\}) \cup \{t\}. \quad (9.21)$$

Example 9.7 Figure 9.10 illustrates the merging step of the diagnoser generation for the mobile robot. To simplify the reading, transitions of Fig. 9.9 have been renamed according to the following correspondence table.

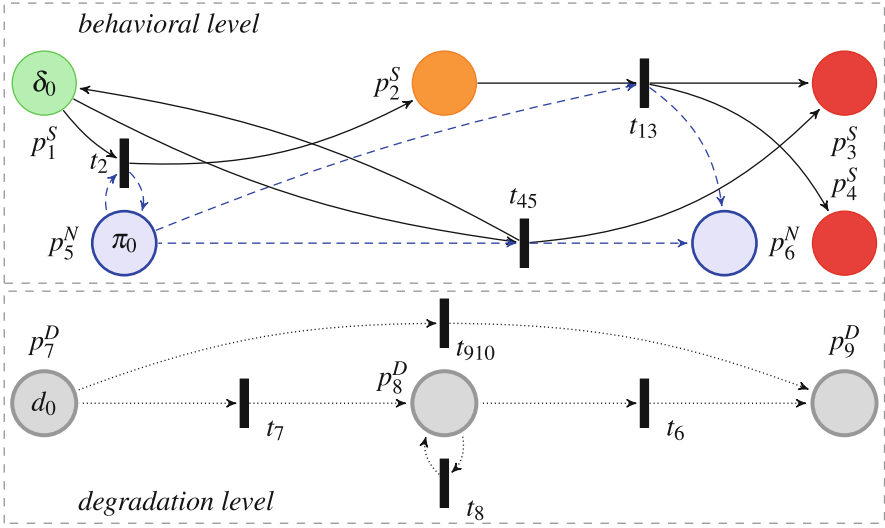


Fig. 9.10 Diagnoser generation of the mobile robot—Step 5 : transition merging

Figure 9.9	t'_1	t'_2	t'_3	t'_4	t'_5	t''_1	t''_2	t''_3	t''_4	t''_5
Figure 9.10	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}

In the behavioral level, transitions t_1 and t_3 (t_4 and t_5) are merged into t_{13} (respectively t_{45}) as they have the same set of input places $\{p_2^S, p_5^N\}$ (respectively $\{p_1^S, p_5^N\}$) and the same numerical place p_6^N as output. In the degradation level, the transitions t_9 and t_{10} are merged into $t_{9,10}$ as they were concurrent.

Step 6 consists in removing the transitions resulting in an elementary loop in the degradation level (pure Petri net).

$$T_\Delta \leftarrow T_\Delta \setminus \{t \mid t \cap P^D = t^\circ \cap P^D\}. \tag{9.22}$$

The goal is to improve the computational performance by avoiding the displacement of the degradation tokens through a transition that loops on the same degradation place. This step has no impact on the tracking quality of the degradation.

Example 9.8 Figure 9.11 shows the diagnoser of the mobile robot. Transition t_8 is removed because it formed an elementary loop with p_8^D .

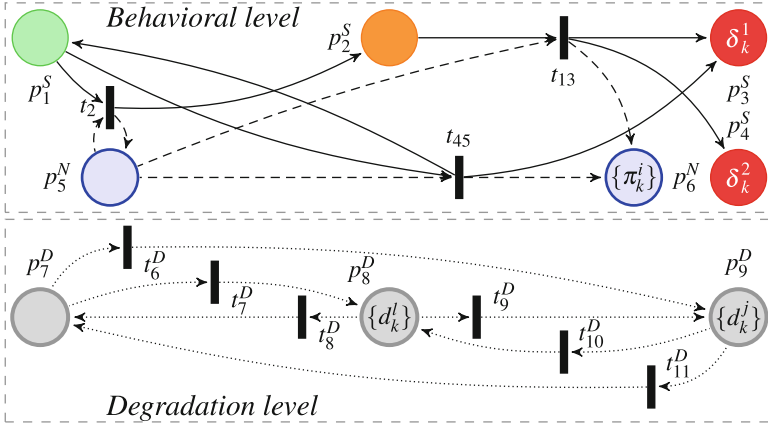


Fig. 9.11 HPPN-based diagnoser HPPN_Δ for the mobile robot

9.5.3 Diagnoser Process

The initial marking $M_0 = \{M_0^S, M_0^N, M_0^D\}$ of the HPPN-based diagnoser represents the system’s initial mode. It is composed of one configuration with value b_0 , n_0^N particles with value x_0 and n_0^D degradation tokens with value d_0 , where n_0^N is the initial number of particles. As long as only one hypothesis is considered in the initial marking, two hypotheses cannot share the same configuration. However, two hypotheses can share the same set of particles if they have the same continuous dynamics but different discrete states (see Example 9.7). From the initial marking and the initial commands, the diagnoser marking \hat{M}_k evolves at time k according to the observations $O_k = O_k^S \cup O_k^N$, where O^S and O^N respectively represent the observations corresponding to the symbolic part and the numerical part.

The estimated marking at time k , $\hat{M}_k = \{\hat{M}_k^S, \hat{M}_k^N, \hat{M}_k^D\}$ where $\hat{M}_k = \hat{M}_{k|k}$, represents all the possible hypotheses on the system mode at time k .

The marking evolution in the HPPN-based diagnoser is based on two steps, prediction and correction, which combine the transition pseudo-firing, particle filters and an algorithm called the Stochastic Scaling Algorithm (SSA).

In particle filtering, the number of particles defines the precision of the filter. The goal of the SSA is to avoid the combinatory explosion and to limit the number of tokens at each step of the algorithm. It dynamically adapts the hypotheses precision. This algorithm is not described in this chapter, but the reader could refer to [13], [28] or [14] to obtain more information about resampling methods for particle filtering.

The prediction step of the online diagnoser process aims at determining all possible next states of the diagnoser $\hat{M}_{k+1|k}$. It is based on the firing of the enabled transitions and on the update of the token values. All the enabled transitions are fired according to the rules described in Sect. 9.4.3. This implies the assumption that a single event can occur at time k . The event set b_k of a configuration δ_k

moved through an arc $a \in A$ during the transition firing is updated according to the annotation $\mathcal{A}(a)$. The value x of a particle π is updated according to the continuous dynamics associated to the numerical place $p^N \in P^N$ in which π belongs after the transition firing. Noise is added during the particle value update to take into account uncertainty about model continuous dynamics. The value d of a degradation token d is updated according to the degradation dynamics associated to the degradation place $p^D \in P^D$ in which d belongs after the transition firing.

The correction step of the online diagnoser process updates the predicted marking $\hat{M}_{k+1|k}$ to the estimated marking $\hat{M}_{k+1|k+1}$ according to new observations O_{k+1} . It is based on the computation of the scores of all hypotheses contained in the marking and on the resampling of the tokens depending on the scores of the hypotheses they represent. The scores of hypotheses are computed with Pr^S and Pr^N , the probability distributions over the symbolic and the continuous states, respectively. Pr^S gives the configuration weights. A configuration weight is computed as the inverse of exponential of the distance between the configuration event set and $O_{k+1}^- = \{O_k | k \leq k+1\}$, the set of symbolic observations until $k+1$. Pr^N gives the normalized particle weights, calculated according to the distance between the particle values and numerical observations O_{k+1}^N . Then, the score of one hypothesis is computed using a weighted function of the sum of its particle weights and its configuration weight:

$$\text{Score}(\delta_k^i, \{\pi_k^j\}, \{d_k^l\}) = \alpha \times \text{Pr}^S(\delta_k^i) + (1 - \alpha) \times \sum_{j=1}^{n_k^N} \text{Pr}^N(\pi_k^j), \quad (9.23)$$

where $\alpha \in [0, 1]$ is the coefficient indicating the global confidence of the symbolic part relatively to the numerical part and $n_k^N = |\{\pi_k^j\}|$ is the number of particles considered for the hypothesis. The score of a hypothesis is always between 0 and 1. A decision making process associates a new number of particles n_{k+1}^N to each set of particles, according to the best score of all the possible modes it belongs and three scale parameters, denoted n_{\min}^N , n_{suff}^N and n_{\max}^N . Each set of particles is then resampled with its associated n_{k+1}^N particles, like in classical particle filtering. Parameters n_{\min}^N and n_{suff}^N are respectively the minimum and the sufficient numbers of particles (but also the number of degradation tokens) to monitor a hypothesis. It means that any n_{k+1}^N is chosen to satisfy the predicate $n_{\min}^N \leq n_{k+1}^N \leq n_{\text{suff}}^N$. Parameter n_{\max}^N is the maximum number of particles (or degradation tokens) available to monitor all hypotheses. It means the total number of particles after the resampling is always less than or equal to n_{\max}^N . During the resampling, degradation tokens linked to duplicated particles are duplicated while those linked to deleted particles are deleted. Finally, configurations that are no longer linked with any degradation tokens are deleted. The correction mechanism highlights that the degradation tokens, in addition to estimate the degradation state, prevent the particle distribution of one hypothesis to be disturbed by the particle distributions of the other hypotheses. In particle filtering, the number of particles defines the precision of the filter but is also a computational performance factor. The scale parameters of the diagnoser process

thus compromise the number of hypotheses to monitor and the precision granted to each one of them, relative to the available computational power (r_{\max}^N can be set up to fulfill performance constraints).

The diagnosis Δ_k is deduced from the marking of the HPPN-based diagnoser HPPN_{Δ} at time k :

$$\Delta_k = \hat{M}_k = \{\hat{M}_k^S, \hat{M}_k^N, \hat{M}_k^D\}. \quad (9.24)$$

It represents all diagnosis hypotheses as a distribution of beliefs over the current health mode and how this mode has been reached. In other words, the marking \hat{M}_k indicates the belief over the continuous state, the fault occurrences and the degradation state. The HPPN-based diagnoser results include the results of a classical diagnoser in terms of fault occurrences. In a classical diagnoser, however, every diagnosis hypotheses has the same belief degree. A HPPN-based diagnoser handles more uncertainty and evaluates the ambiguity according to the tokens places and values.

9.6 Case Study

This section focuses on the application of the proposed methodology on the K11 planetary rover prototype. The K11 is a four-wheeled rover designed as a platform for testing power-efficient rover architectures in Antarctic conditions [25]. The K11 has then been redesigned by NASA Ames Research Center for diagnostics and Prognostics-enabled Decision Making research [1, 7, 37]. It has been transformed into a testbed to simulate some fault occurrences and failures. In this work, it is studied as a functional rover exposed to failures and executing missions.

9.6.1 Rover Description

The K11 rover is powered by twenty-four 2.2 Ah lithium-ion single cell batteries. A typical mission of the rover consists in visiting and performing desired science functions at a set of waypoints, before joining its charging station. A decision making module (DM) is responsible for determining the order in which to visit the waypoints according to the terrain map, the waypoint positions and rewards, and the rover conditions. The rover has four wheels, denominated by their location: the front-left (FL) wheel, the front-right (FR) wheel, the back-left (BL) wheel and the back-right (BR) wheel. Each wheel is driven by an independent 250 W graphite-brush motor, with control performed by a single-axis digital motion controller. An onboard laptop computer runs the control and data acquisition software. The rover is a skid-steered vehicle, meaning that the wheels cannot be steered and the rover is rotated by commanding the wheel speeds on the left and right sides

Table 9.1 Continuous commands, continuous measurements, and fault types on the K11

<i>Command type</i>	<i>Comments</i>	<i>Units</i>
Wheel speed	Commanded speeds for wheels on the same side are the same	rad/s
<i>Measurement type</i>	<i>Comments</i>	<i>Units</i>
Wheel speed	One for each wheel	rad/s
Total current	A current sensor on the power bus	A
Motor current	One for each motor	A
Motor temperature	One for each motor	°C
Battery temperature	One for each battery cell	°C
Battery voltage	One for each battery cell	V
<i>Fault event labels</i>	<i>Fault descriptions</i>	<i>Effects</i>
f_1	Battery charge depletion	Lead to failure
f_2	Parasitic electric load	Increase battery drain
f_3, f_4, f_5, f_6	Increased motor frictions	Increase battery drain and motor temperatures
f_7, f_8, f_9, f_{10}	Motor overheating	Lead to failure
$f_{11}, f_{12}, f_{13}, f_{14}$	Failed motor temperature sensors	Unable to sense motor temperatures

to different values. The battery management system provides battery charging and load balancing capabilities. It also sends voltage and temperature measurements for each of the individual cells to the onboard computer. The data acquisition module collects current and motor temperature measurements and sends them to the onboard computer. The motor controllers send back motion data such as commanded speeds and actual speeds. More details on the rover can be found in [1].

All the continuous observations on the rover and the list of faults we consider in this study are presented in Table 9.1. Four signals command the wheels with a proportional-integral-derivative controller and the set of sensors returns 61 measurement signals. Several fault types have been implemented on the testbed and are related to the power system (battery), the electro-mechanical system (motors, controller), and the sensors (drift, bias, scaling or failure).

The K11 rover has no discrete actuator or discrete sensor and thus has mostly been studied as a continuous system, where faults were defined as constraints on the continuous state. We propose to abstract anticipated faults into unobservable events. The multi-mode system that describes the rover health evolution is presented in Fig. 9.12. To simplify the description, only a part of the multi-mode system is shown. The modes corresponding to consecutive fault occurrences are not included and only the front-left motor is considered.

The rover is in mode *Nominal*₁ with continuous dynamics C_1 as long as no fault has occurred. Fault f_1 occurrence represents the *end of discharge* (EOD) of the battery, i.e. the date when the battery is too discharged to power the system. This is assumed to occur when the battery voltage is lower than 3.25 V and it leads to the mission failure (mode *Failed*₁ with continuous dynamics C_5). Fault

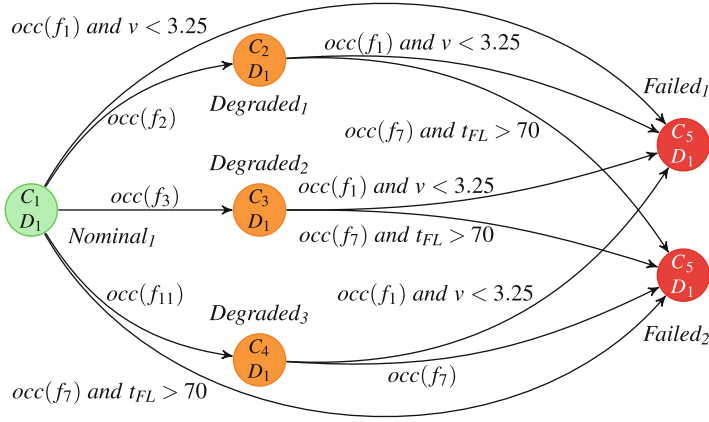


Fig. 9.12 Streamlined description of the rover health evolution

f_2 represents the emergence of a parasitic battery load arising from an electrical submodule continuously engaged, for example. The parasitic load increases the total current and thus the battery drain (mode *Degraded₁* with continuous dynamics C_2), which causes the system to reach the EOD prematurely. Fault f_3 (f_4 , f_5 and f_6) represents an increased friction of the FL (FR, BL and BR) motor. The increased friction induces the need for a larger amount of current to satisfy the same speed (mode *Degraded₂* with continuous dynamics C_3). Furthermore, the load demands will be higher, raising the motor temperature. The most critical scenario for a motor is an overheating. In such case, the heat will eventually destroy the insulation of the windings, causing electrical shorts and leading to motor failure. The overheating of the FL (FR, BL and BR) motor is represented by fault f_7 (f_8 , f_9 and f_{10}). The occurrence of any one of these faults leads to the rover failure (mode *Failed₂* with continuous dynamics C_5) and thus represents the rover *end of life* (EOL). A motor is assumed to overheat when its temperature exceeds 70 °C. The motor temperatures are measured by four sensors. These sensors, however, are known to fail unexpectedly, sending inconsistent values. These failures are represented by faults f_{11} , f_{12} , f_{13} and f_{14} . We consider that the temperature model is not accurate enough without a correction step with observations. As a consequence, once f_{11} (f_{12} , f_{13} and f_{14}) has occurred, the occurrence of fault f_7 (f_8 , f_9 and f_{10}) does not match with any condition on the FL (FR, BL and BR) motor temperature (see the arc between *Degraded₃* and *Failed₂*). In Fig. 9.12, mode *Degraded₃* with continuous dynamics C_4 represents the mode where the temperature sensor of the FL motor has failed. The rover degradation state can be monitored with degradation dynamics D_1 , which corresponds to the identity dynamics.

9.6.2 Rover Modeling

Considering all the motors and the consecutive fault combinations, we identified 192 modes and 240 mode changes. The HPPN-based model of the rover has 241 places (192 symbolic places, 48 numerical places, 1 degradation place) and 240 transitions. The HPPN-based diagnoser has the same number of places and transitions because the merging step of the diagnoser generation (Step 5) does not reduce the number of transitions (it is specific to this case study). Actually, the merging step merges transitions having exactly the same sets of input and output places. This kind of transition does not exist in this real application, but it may be very useful in other cases. The degradation place is removed from the transition inputs and outputs, reducing the complexity of the net. Because there is only one degradation place, all transitions in the degradation level are removed by Step 6 of the diagnoser generation. The underlying DES of the multi-mode system and HPPN-based model and diagnoser of the K11 rover are available at <https://homepages.laas.fr/echanthe/PetriNets2016>.

The nominal continuous dynamics is represented as a set of differential equations that unifies the battery model with the rover motion model and the temperature models. It can be converted to a discrete-time representation and solved with a sample time of 1/20 s, while continuous observation sampling is about 1 s. We consider 30 state variables for the rover, including the rover three-dimensional position, its relative angle position, the wheel control errors, the motor temperatures and motor winding temperatures. The 24 batteries are lumped into a single one to only consider 5 battery state variables (3 charges, the temperature and the voltage) instead of 120. The battery model has been validated with experimental data in previous works [7, 37]. Unifying the battery model with motion and temperatures, however, increases uncertainty about the rover model.

Fault f_2 occurrence and effect on the system behavior are modeled as a time varying parameter. The parasitic battery load is captured as an additional current reaching a value between 1.5 and 4.5 A from value 0 A in a few seconds after the fault occurrence. First, two parameters are added to the continuous state vector to monitor both the duration since the fault occurrence and the additional current value. Then, the uncertain rise of the additional current is modeled by adding a Gaussian noise, with a mean and standard deviation values starting respectively at 3 and 0.3, and decreasing to 0 while the duration since the fault occurrence increases.

Finally, the temperature model is quite uncertain so temperature measurements are assumed to be reliable when sensors are not failed. We model fault f_{11}, f_{12}, f_{13} and f_{14} by increasing significantly the motor temperature sensor noise because failed sensors only send inconsistent large values with no pattern. Fault f_3, f_4, f_5 and f_6 and increased motor frictions can be modeled with time varying parameters (as additional motor resistances) like f_2 but are not monitored in this study.

9.6.3 Simulation Results

The HPPN framework is implemented in Python 3.4. The tests were performed on a 4 Intel(R) Core(TM) i5-4590 CPU at 3.30 GHz with 16 GB of RAM and running GNU/Linux (Linux 3.13.0 – 74, x8664). In order to reduce computation time, the token value update step is multithreaded on the four physical cores. The rest of this implementation only uses one core.

Two scenarios studied in [37] are considered in this work. The rover mission is to visit a maximum of 12 waypoints and to go back to its starting position. All waypoints have different associated rewards. In nominal conditions, the rover DM system returns a 5-waypoints path, starting and finishing at the same position. For all scenarios, the K11 rover starts at 0 s with batteries fully charged and with all components at the ambient temperature. The K11 rover currently has, however, two motor temperature sensors (FL and BL) failed. These faults do affect the monitoring but not the physical system, so the DM returns the same path as in nominal conditions. These sensor faults are diagnosed in one sampling period by the diagnoser if we consider the initial mode to be unknown, so we assume to know the rover initial degraded mode, and we have only one hypothesis in the initial diagnoser marking.

For the sake of clarity, in the rest of the paper, health modes are designated with representative keywords of the rover state. For example, the initial mode is designated as *Sensor BL FL fault*. The initial number of particles and degradation tokens is $n_0^N = 100$. The scale parameters of the diagnoser process are set to $(n_{\min}^N, n_{\text{suff}}^N, n_{\max}^N) = (40, 80, 6000)$.

9.6.3.1 Scenario 1

In Scenario 1, no fault occurs. The rover successfully executes its mission. Figure 9.13 presents the diagnosis hypotheses as the distribution of beliefs over the current health mode at any time.

The belief degree of a possible mode is the score computed for the related hypothesis with Eq. (9.23) and α set to 0.5. Any belief degree is between 0 and 1, this represents a score, so the sum of the belief degrees of all possible modes is not 1. In Fig. 9.13, the maximum belief degree of a mode at any time is represented by the thickness of the line and the highest belief degree of all the modes is plotted in blue. The gap between 81 and 281 s corresponds to a break during the experiment. The figure shows that the diagnoser keeps the real mode *Sensor BL FL fault* in its set of hypotheses and assigns it the highest belief degree almost all along the scenario. Other modes are also highly considered by the diagnoser at any time because of the model-based uncertainty. The combination of continuous and discrete evolutions is explained by the marking rules in the HPPN. As the diagnoser generation process replaces every symbolic condition by a TRUE condition, the emphasis is put on the continuous evolution of the system. It means that faults are essentially detected

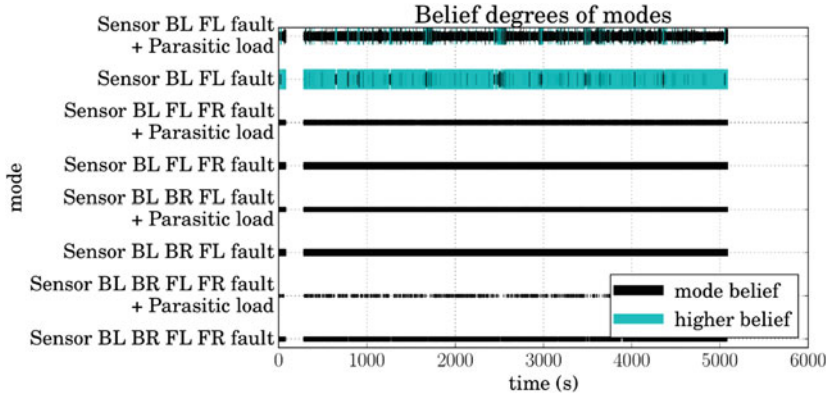


Fig. 9.13 Scenario 1: mode belief at any time

by continuous clues. Even if a discrete event occurs before the satisfaction of the associated degradation condition, the evolution of the system will always be followed thanks to the pseudo-firing process.

9.6.3.2 Scenario 2

In Scenario 2, a battery parasitic load occurs between 660 and 695 s, and the DM system cancels the visit of the farthest waypoint. Fault f_2 occurrence is immediately detected by the diagnoser (Fig. 9.14). After 678 s, the possibility of being in mode *Sensor BL FL fault + Parasitic load* is the highest until the end of the mission. The fault load is estimated (most likely) at 1.39 A at 678 s, 1.73 A at 679 s, 2.16 A at 683 s and 2.16 A at 3906 s. A zoom between 570 and 760 s on the trajectories of the modes that are still possible at 3906 s (Fig. 9.15) shows that fault f_2 is believed to occur between 631 and 694 s, and most likely between 677 and 689 s. These results are consistent with our analysis of the measured total current.

Faults are always detected in one sampling period because the HPPN-based diagnoser considers all hypotheses (including the hypotheses concerning faults with slow degradation) during the prediction step due to pseudo-firing. Moreover it keeps the matching marking during the correction step. However, the isolation may be longer than one sampling period. The results show that the diagnoser grants most of the time, but not always, the highest belief to the real mode. The diagnosis, however, carries all the explanation of the observations as a distribution of beliefs, and then the real mode is always considered in the set of diagnosis hypotheses. This illustrates the robustness of the HPPN-based diagnoser to the rover model and data. The average diagnosis computation time and token number are 13.3 s and 8801.4, respectively. These metrics point out the diagnosis computation time remains acceptable compared to the system model computational complexity.

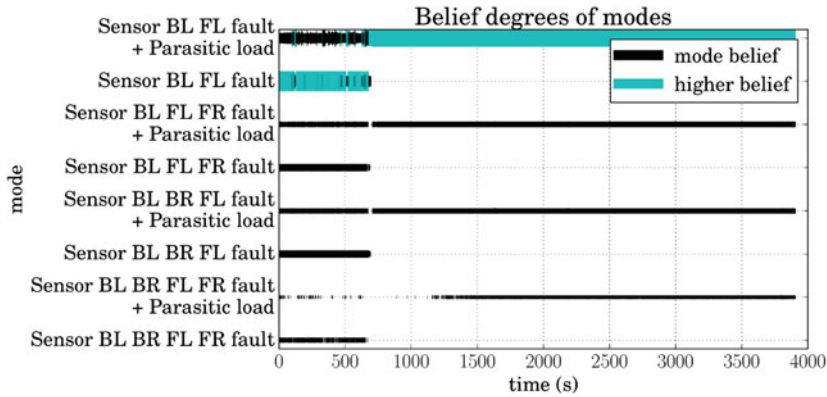


Fig. 9.14 Scenario 2: mode belief at any time

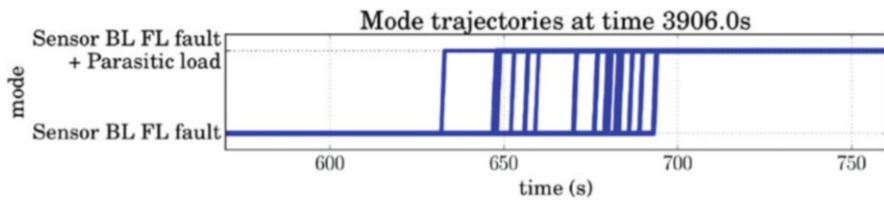


Fig. 9.15 Scenario 2: trajectories of possible modes at time 3906 s

The maximum RAM used by Scenario 1 and 2 are 140.7 and 141.8 MB. More extended performance analyses are proposed in the next section.

The case study results show that HPPN-based diagnosis is robust to real system data and constraints and adaptable to systems without discrete observations nor degradation knowledge.

9.6.3.3 Performance Analysis, Comparison with Other Approaches

Diagnosis computation times, and the maximum RAM used for different sets of scaling parameters, are given in Table 9.2. Tests have been performed on three scenarios (including the nominal and faulty scenarios presented above) and run 12 times. 54,403 diagnoses are computed.

These metrics point out that computation times with the initial scaling parameters remain acceptable but do not respect real-time constraints; observations sampling is about 1 s and the average diagnosis computation time is 3.35 s. This is mainly due to that diagnosis process relies on parallel step-by-step simulations but it is also due to the rover model computational complexity.

The methodology theoretical complexity is difficult to evaluate because it depends on the continuous equations, the DES structure and the token number,

Table 9.2 Computational performances of the HPPN-based diagnosis method for different scaling parameters

Scaling parameters		Δ_k time (s)	Max. RAM (MB)
(40, 80, 1500) $_{\Delta}$	Minimum	0.28	126.73
	Maximum	4.54	
	Average	3.35	
(40, 80, 400) $_{\Delta}$	Minimum	0.28	122.15
	Maximum	1.00	
	Average	0.56	
(20, 60, 400) $_{\Delta}$	Minimum	0.22	112.18
	Maximum	0.98	
	Average	0.74	

among others. Moreover, software implementation, compilation optimization or virtual machine execution are also other performance factors difficult to evaluate in practice. This is why we propose in this work to approach performance constraints by tuning the scaling parameters.

Other diagnosis works have been conducted on the rover [1], such as the QED algorithm, described in [8] or the HyDe (Hybrid Diagnostic Engine) [31]. The main advantage of our approach is that the diagnosis estimates are not presented as a set of candidates, but as a distribution of candidates.

In case of decision making in a health management context, the operator may take a more justified decision. In terms of detection delay, QED and HyDe detect the fault in less than 1 s. QED isolates it in 26 s and has a good estimate of the parasitic load (3% of relative error) in about 50 s. Our method detects the fault after the first diagnosis. The estimation of the worse isolation time is about 18 s. The estimation of the parasitic load is good (8% of relative error) after 23 s.

9.7 Conclusion

This work applies the approach of health monitoring based on Hybrid Particle Petri Nets to a real case study, the K11 planetary rover prototype. The HPPN framework is particularly useful to take into account knowledge-based and observation-based uncertainty. The HPPN-based diagnoser deals with event occurrence possibility and knowledge imprecision. It monitors both discrete and continuous dynamics, as well as degradation evolution, in order to introduce concepts that will be useful to perform prognosis and health management of hybrid systems under uncertainty. In addition, diagnosis results can be used as probability distributions for decision making.

Then, the methodology was applied on the K11 rover. A hybrid model of the rover has been proposed by discretizing its health evolution and defining fault events. The system model and diagnoser have been generated in the HPPN

framework and two scenarios have been tested to illustrate the proposed method advantages. The diagnoser results are consistent with the expected ones and show that HPPN-based diagnosis is robust to real system data and constraints and adaptable to systems without discrete observations nor degradation knowledge.

Other works aim at formalizing and developing a prognosis process that will interleave diagnosis and prognosis methods to obtain more accurate results. The HPPN-based prognostics methodology has been defined and tested on a three-tank system as well as on the K11 rover.

This work has a lot of interesting perspectives. The first one is the extension of the work to very large systems. To be applied on real large scale systems, the proposed methodology could be adapted in the context of decentralized diagnosis structures as the approaches developed in [35]. The second perspective deals with the major hypothesis on the HPPN model-based approach: the system model is assumed to be correct and complete. Machine learning techniques may be used to adapt this predefined model with new collected data [23, 26]. Another perspective is to use machine learning methods to improve the detection of small drift in the system parameters. The combination of model-based and data-driven approaches for diagnosis is under investigation [22, 38, 39].

References

1. Balaban, E., Narasimhan, S., Daigle, M. J., Roychoudhury, I., Sweet A., Bond, C., et al. (2013). Development of a mobile robot test platform and methods for validation of prognostics-enabled decision making algorithms. *International Journal of Prognostics and Health Management*, 4(006), 1–19.
2. Basile, F., Chiacchio, P., & Tommasi, G. D. (2009). Fault diagnosis and prognosis in Petri nets by using a single generalized marking estimation. In *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Barcelona, Spain.
3. Bayouhd, M., Travé-Massuyes, L., & Olive, X. (2008). Hybrid systems diagnosis by coupling continuous and discrete event techniques. In *IFAC World Congress*, Seoul, Korea (pp. 7265–7270).
4. Biswas, G., Simon, G., Mahadevan, N., Narasimhan, S., Ramirez J., & Karsai, G. (2003). A robust method for hybrid diagnosis of complex systems. *IFAC Proceedings Volumes*, 36(5), 1023–1028.
5. Cardoso, J., Valette, R., & Dubois, D. (1999). Possibilistic Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(5), 573–582.
6. Chanthery, E., & Ribot, P. (2013). An integrated framework for diagnosis and prognosis of hybrid systems. In *3rd Workshop on Hybrid Autonomous System*, Rome, Italy.
7. Daigle, M., Roychoudhury, I., & Bregon, A. (2014). Integrated diagnostics and prognostics for the electrical power system of a planetary rover. In *Annual Conference of the PHM Society*, Fort Worth, TX, USA.
8. Daigle, M., Roychoudhury, I., & Bregon, A. (2015). Qualitative event-based diagnosis applied to a spacecraft electrical power distribution system. *Control Engineering Practice*, 38, 75–91.
9. Daigle, M., Sankararaman, S., & Kulkarni, C. S. (2015). Stochastic prediction of remaining driving time and distance for a planetary rover. In *IEEE Aerospace Conference*.
10. David, R., & Alla, H. (2005). *Discrete, continuous, and hybrid Petri nets*. New York: Springer.

11. Ding, S. X. (2014). *Data-driven design of fault diagnosis and fault-tolerant control systems*. New York: Springer.
12. Dotoli, M., Fanti, M. P., Giua, A., & Seatzu, C. (2008). Modelling systems by hybrid Petri nets: An application to supply chains. In *Petri net, theory and applications*. InTech.
13. Douc, R., & Cappé, O. (2005). Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005. ISPA 2005* (pp. 64–69). New York: IEEE.
14. Doucet, A., & Johansen, A. M. (2009). A tutorial on particle filtering and smoothing: Fifteen years later. In *Oxford handbook of nonlinear filtering*. University Press.
15. Gaudel, Q., Chantry, E., & Ribot, P. (2014). Health monitoring of hybrid systems using hybrid particle Petri nets. In *Annual Conference of the PHM Society*, Fort Worth, TX, USA.
16. Gaudel, Q., Chantry, E., & Ribot, P. (2015). Hybrid particle Petri nets for systems health monitoring under uncertainty. *International Journal of Prognostics and Health Management*, 6(022), 1–20.
17. Gaudel, Q., Chantry, E., Ribot, P., & Le Corronc, E. (2014). Hybrid systems diagnosis using modified particle Petri nets. In *25th International Workshop on Principles of Diagnosis*, Graz, Austria.
18. Henzinger, T. (1996). The theory of hybrid automata. In *11th Annual IEEE Symposium on Logic in Computer Science* (pp. 278–292).
19. Hofbaur, M. W., & Williams, B. C. (2002). Mode estimation of probabilistic hybrid systems. *Lecture Notes in Computer Science*, 2289, 253–266.
20. Horton, G., Kulkarni, V. G., Nicol, D. M., & Trivedi, K. S. (1998). Fluid stochastic petri nets: Theory, applications, and solution techniques. *European Journal of Operational Research*, 105(1), 184–201.
21. Jianxiong, W., Xudong, X., Xiaoying, B., Chuang, L., Xiangzhen, K., & Jianxiang, L. (2013). Performability analysis of avionics system with multilayer HM/FM using stochastic Petri nets. *Chinese Journal of Aeronautics*, 26(2), 363–377.
22. Jung, D., Ng, K. Y., Frisk, E., & Krysander, M. (2016). A combined diagnosis system design using model-based and data-driven methods. In *3rd Conference on Control and Fault-Tolerant Systems (SysTol)* (pp. 177–182). New York: IEEE.
23. Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., & Ghédira, K. (2016). Discussion and review on evolving data streams and concept drift adapting. In *Evolving systems* (pp. 1–23). Berlin: Springer.
24. Koutsoukos, X., Kurien, J., & Zhao, F. (2002). Monitoring and diagnosis of hybrid systems using particle filtering methods. In *15th International Symposium on Mathematical Theory of Networks and Systems*, Notre Dame, IN, USA.
25. Lachat, D., Krebs, A., Thueer, T., & Siegwart, R. (2006). Antarctica rover design and optimization for limited power consumption. In *4th IFAC Symposium on Mechatronic Systems*.
26. Leclercq, E., Medhi, E., Ould, S., & Lefebvre, D. (2008). Petri nets design based on neural networks. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (pp. 529–534).
27. Lesire, C., & Tessier, C. (2005). Particle Petri nets for aircraft procedure monitoring under uncertainty. In G. Cardio & P. Darondeau (Eds.), *ICATPN 2005. Lecture notes in computer science* (Vol. 3536, pp. 329–348). Heidelberg: Springer.
28. Li, T., Bolic, M., & Djuric, P. M. (2015). Resampling methods for particle filtering: Classification, implementation, and strategies. *IEEE Signal Processing Magazine*, 32(3), 70–86.
29. Narasimhan, S., Balaban, E., Daigle, M., Roychoudhury, I., Sweet, A., Celaya, J., et al. (2012). Autonomous decision making for planetary rovers using diagnostic and prognostic information. In *8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Mexico (pp. 289–294).
30. Narasimhan, S., & Biswas, G. (2007). Model-based diagnosis of hybrid systems. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(3), 348–361.

31. Narasimhan, S., & Brownston, L. (2007). Hyde—A general framework for stochastic and hybrid model-based diagnosis. In *18th International Workshop on Principles of Diagnosis* (Vol. 7, pp. 162–169).
32. Narasimhan, S., Dearden, R., & Benazera, E. (2004). Combining particle filters and consistency-based approaches for monitoring and diagnosis of stochastic hybrid systems. In *15th International Workshop on Principles of Diagnosis*.
33. Ru, Y., & Hadjicostis, C. N. (2009). Fault diagnosis in discrete event systems modeled by partially observed Petri nets. *Discrete Event Dynamic Systems*, 19(4), 551–575.
34. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., & Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on Robotics and Automation*, 40(9), 1555–1575.
35. Sayed-Mouchaweh, M., & Lughofer, E. (2015). Decentralized approach without a global model for fault diagnosis of discrete event systems. *International Journal of Control*, 88(11), 2228–2241. <https://doi.org/10.1080/00207179.2015.1>
36. Soldani, S., Combacau, M., Subias, A., & Thomas, J. (2007). On-board diagnosis system for intermittent fault: Application in automotive industry. In *7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems* (Vol. 7-1, pp. 151–158). <https://doi.org/10.3182/20071107-3-FR-3907.00021>
37. Sweet, A., Gorospe, G., Daigle, M., Celaya, J. R., Balaban, E., Roychoudhury, I., et al. (2014). Demonstration of prognostics-enabled decision making algorithms on a hardware mobile robot test platform. In *Annual Conference of the PHM Society*, Fort Worth, TX, USA.
38. Tidriri, K., Chatti, N., Verron, S., & Tiplica, T. (2016). Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: A review of researches and future challenges. *Annual Reviews in Control*, 42, 63–81.
39. Toubakh, H., & Sayed-Mouchaweh, M. (2016). Hybrid dynamic classifier for drift-like fault diagnosis in a class of hybrid dynamic systems: Application to wind turbine converters. *Neurocomputing*, 171, 1496–1516.
40. van der Merwe, R., Doucet, A., De Freitas, N., & Wan, E. (2000). The unscented particle filter. In *Annual Conference on Neural Information Processing Systems* (Vol. 2000, pp. 584–590).
41. Vianna, W. O. L., & Yoneyama, T. (2015). Interactive multiple-model application for hydraulic servovalve health monitoring. In *Annual Conference of the PHM Society*, Coronado, CA, USA.
42. Wang, W., Li, L., Zhou, D., & Liu, K. (2007). Robust state estimation and fault diagnosis for uncertain hybrid nonlinear systems. *Nonlinear Analysis: Hybrid Systems*, 1(1), 2–15.
43. Yu, M., Wang, D., Luo, M., & Huang, L. (2011). Prognosis of hybrid systems with multiple incipient faults: Augmented global analytical redundancy relations approach. *IEEE Transactions on Systems, Man, and Cybernetics*, 41(Part A, 3), 540–551.
44. Zaidi, A., Zanzouri, N., & Tagina, N. (2006). Modelling and monitoring of hybrid systems by hybrid Petri nets. In *10th WSEAS International Conference on Systems*.
45. Zhao, F., Koutsoukos, X., Haussecker, H., Reich, J., & Cheung, P. (2005). Monitoring and fault diagnosis of hybrid systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35(6), 1225–1240.
46. Zouaghi, L., Alexopoulos, A., Wagner, A., & Badreddin, E. (2011). Modified particle Petri nets for hybrid dynamical systems monitoring under environmental uncertainties. In *IEEE/SICE International Symposium on System Integration* (pp. 497–502).

Chapter 10

Diagnosis of Hybrid Dynamic Systems Based on the Behavior Automaton Abstraction



Ramon Sarrate, Vicenç Puig, and Louise Travé-Massuyès

10.1 Introduction

The majority of real systems are controlled online and supervised by means of automatic computer-based control systems. The behavior of these systems arises from continuous plant dynamics that can be described by continuous state variables and supervisory control that generates actuator signals at discrete-time points to change regulator set-points or the plant configuration. Diagnosing these systems is a real issue as they are subject to faults that may appear in any of the plant components, in sensors or actuators [14, 24, 28, 30].

These complex systems are modeled using hybrid models that integrate continuous and discrete dynamics. These often take the form of hybrid automaton models [19] or hybrid bond graph models [15, 24]. Then, this model can support the monitoring of the system, fault diagnosis and control tasks. Model-based online diagnosis requires quick and robust reconfiguration processes when a mode change occurs, as well as the ability to keep the nominal behavior of the system on track during transient states [11].

A hybrid automaton models the behavior of a hybrid system through a set of operation modes and a set of transitions between modes which trigger upon discrete events or based on continuous state conditions. Continuous dynamics within each mode are described by a set of differential equations which constrain the continuous state, input and output variables. Input and output variables are

R. Sarrate · V. Puig (✉)

Automatic Control Department, Universitat Politècnica de Catalunya (UPC), Terrassa, Spain
e-mail: ramon.sarrate@upc.edu; vicenc.puig@upc.edu

L. Travé-Massuyès

Laboratoire d'Analyse et d'Architecture des Systèmes, Centre National de la Recherche Scientifique (LAAS-CNRS), Université de Toulouse, Toulouse, France
e-mail: louise@laas.fr

measured. Discrete events may be observable or unobservable. Observable events may represent commands issued by the controller or changes in state variables recorded by sensors (i.e., when a state variable crosses a threshold). Unobservable events may represent failure events or other events that cause changes in the system state not directly recorded by sensors.

This chapter focuses on the use of the hybrid automaton framework to develop a method for diagnosing hybrid systems [14, 28, 30]. Diagnosis is directly performed by interpreting the events and measurements issued by the physical system with respect to the hybrid automaton model. The presented framework is the result of a series of works related historically in Sect. 10.2.2 and it can cope with both structural and non-structural faults. The idea is to consider a hybrid automaton model as a twofold mathematical object. The discrete event part constrains the possible transitions among modes and is referred to as the *underlying DES*. The restriction of the hybrid system to the continuously-valued part of the model is defined as the *multimode system*. The diagnosis method relies on abstracting the continuous dynamics by defining a set of “distinguishability-aware” events, called *signature-events*, capturing the consistency checks performed on the set of residuals associated to every mode. Signature-events are used to enrich appropriately the underlying DES to obtain the so-called *behavior automaton (BA)* from which a diagnoser can be built following standard methods of the discrete event system field.

The proposed hybrid diagnosis method can operate in a non-incremental and an incremental manner. In the non-incremental form, algorithms are executed taking into account global models whereas in the incremental form only the useful parts of the diagnoser are built, developing the branches that are needed to explain the occurrence of incoming events.

Generally, a hybrid system operates in a small region compared to the entire behavioral space defined by the hybrid automaton states. Thus, significant gain can hence be expected from the incremental approach in terms of memory storage compared to building the full diagnoser offline. The methodology is validated by the application to a case study based on a representative part of the Barcelona sewer network.

The structure of the paper is the following. In Sect. 10.2, a historical review of the behavior automaton abstraction approach to diagnose hybrid systems and an overview of the proposed method is provided. In Sect. 10.3, the principles of the diagnosis method are explained and the method to build the behavior automaton and the corresponding diagnoser is presented. Section 10.4 motivates the incremental version of the diagnosis method and presents how it can be implemented in an incremental framework. In Sect. 10.4, an application case study based on the sewer network of the Barcelona city is used to illustrate the proposed approach. Finally, conclusions are drawn in Sect. 10.6.

10.2 Hybrid System Diagnosis Methodology Overview

10.2.1 Diagnosis Architecture

The architecture of the diagnosis approach based on the behavior automaton abstraction to detect and isolate faults in hybrid systems is given in Fig. 10.1. Compared to the classical FDI conceptual block, the architecture includes a mode recognition block that adapts the FDI module online. Indeed, FDI algorithms must take into account the current operation mode q_i of the hybrid system to adapt the model used to generate predictions. The diagnosis procedure includes an offline and an online process.

In the offline process, the hybrid automaton model (HA) is built through the component parallel composition and the generation of a set of equations which depend on the operation mode. Residuals for each mode are generated and mode discernibility, also called distinguishability by other authors, is analyzed. Discernibility analysis and observable events allow to build the behavior automaton (B) which carries all the information about the hybrid system diagnosability. The behavior automaton is turned into a diagnoser used to detect mode changes and identify the current mode of the system.

In the online process, the tasks are carried out by the three blocks highlighted in blue in Fig. 10.1. *Mode recognition* and *fault diagnosis* blocks cooperate to deal with possible changes in the system operation mode based on consistency indicators and

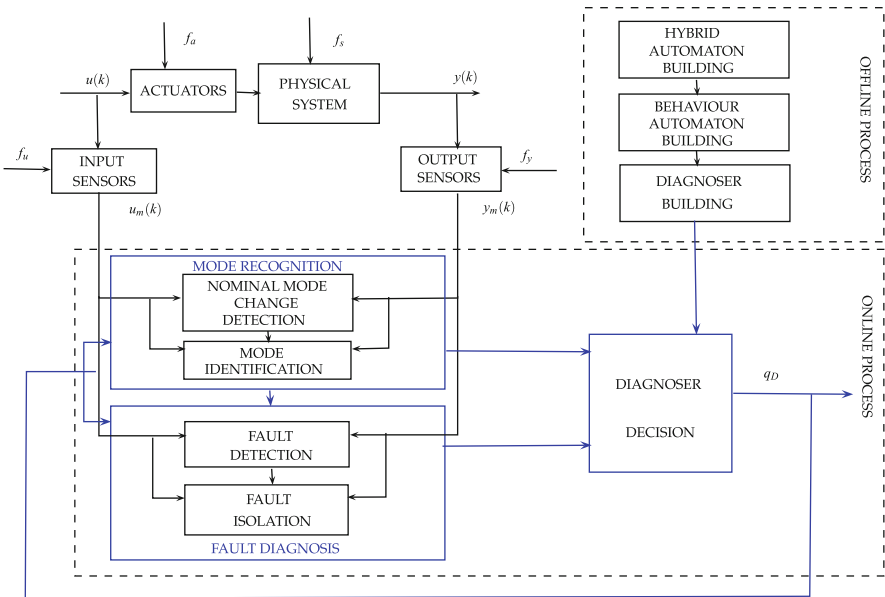


Fig. 10.1 Conceptual block diagram of the hybrid diagnosis method

observable event occurrences. The *diagnoser decision* block gives a final diagnosis according to the information provided by the *mode recognition* and *fault diagnosis* blocks.

The current diagnoser state (q_D) reports the set of possible modes of the hybrid system at a given time. If more than one mode are in q_D , those modes are non-discernible. A mode change in *HA* implies a nominal, structural faulty or non-structural faulty mode change.

Discernibility is used to predict if a mode change can be detected and identified when the operation mode is described by a dynamic model [6, 14, 23]. In the case of non-structural faults, discernibility properties are related to *detectability* and *isolability* based on the fault signature matrix [23]. An abstract concept of discernibility is defined which includes all the properties in a unique and general form to predict whether a mode change has occurred according to the nature of the mode (indicating properly when a fault is present).

The steps implied by the behavior automaton method are shown in Fig. 10.2.

Let us mention that in the online diagnosis process, the following assumptions are made:

Assumption 1 Two mode changes do not occur at the same time.

Assumption 2 The residual dynamics have time to stabilize between two consecutive mode switchings.

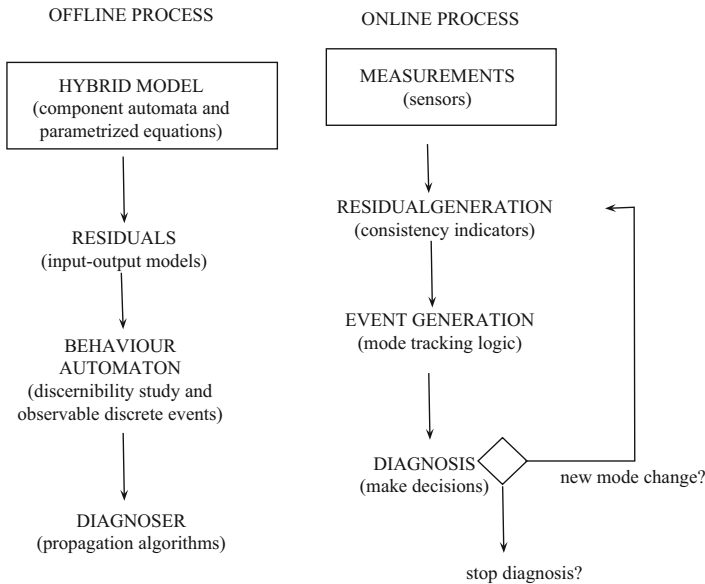


Fig. 10.2 Design methodology steps

Assumption 2 implies that transitions between modes should be slower than the residual generator dynamics. This concerns a dwell time requirement, i.e. the time needed by the continuous dynamics to reach the steady state of a given operation modes before another transition occurs.

Assumption 3 After a mode change occurrence, all the residuals sensitive to this change must be activated at some time and persist during the whole mode change isolation process.

Assumption 4 No mode change occurs after a non-structural fault.

The last assumption implies that once a non-structural fault is detected, the online diagnosis process must stop. Indeed, the set of residuals and models must be adapted to appropriately continue to perform diagnosis. In the case of a structural fault occurrence, the diagnosis task can continue even if the system is not repaired.

10.2.2 Historical Review

Hybrid models are powerful formalisms able to represent multiple continuous dynamics associated to several operation modes of a system. When used to support the diagnosis task, these mathematical objects may include the representation of normal and faulty modes. The modes of the system then represent a valuation of the discrete state of the system whereas the continuous state is given by a subset of the continuous variables defining the continuous dynamics. Discrete state and continuous state form the hybrid state of the system.

For a while, the methods to diagnose hybrid systems were based on estimating the full hybrid state with methods like multiple model filtering [10, 18] and particle filtering [16] or other hybrid estimation schemes like [19] and [8]. Nevertheless, diagnosis information is mainly carried by the discrete state. Based on this observation, hybrid diagnosis approaches based on estimating only the discrete state of the system were then proposed. These are based on abstracting continuous dynamics for each operation mode in the form of a set of residuals issued from analytical redundancy relations (ARRs). ARR are input–output relations obtained from the continuous model by eliminating the state variables thanks to the elimination theory [9]. In the diagnosis field targeting continuous systems, this approach is also known as the *parity-space* approach.

Residuals provide a way to check the consistency of the measurements with respect to the continuous dynamics associated to each mode and hence to eventually identify the current mode of the system. For this to be possible, one must rely on *discernibility*, concept introduced for the first time in [14], which defines the property assessing whether or not two modes can be distinguished based on continuous measurements.

In [14] and later in [30], the operation modes represent only nominal behavior and diagnosis focuses on fault detection and isolation of *non-structural faults*, i.e.

faults that do not change the structure of the model like sensor and actuator faults which are additive faults. The impact of non-structural faults on the residuals of every mode is assumed to be known and is captured by theoretical signatures generated using the *sensitivity* concept [23]. Tracking the system mode involves detecting the mode change, i.e. detecting that the set of residuals of the current mode are not consistent with measurements, and identifying which successor mode is the actual mode of the system by determining which set of residuals is consistent.

In the above works, the hybrid system evolution is viewed as a change of continuous dynamics and assessed only from continuous measurements. In [4], the authors elaborate on the concept of discernibility and propose for the first time to account for discrete events that may be involved in the discrete dynamics, e.g. an opening valve action or a generator switch off. The hybrid system is represented by a hybrid automaton in which some transitions are labelled by such discrete events, these being observable or not observable. Not observable events represent the change of status of a guard, i.e. a condition about the continuous state, or the occurrence of a fault. By doing so, operation modes may be nominal or faulty, leading to the capability of detecting and isolating *structural faults*. Most importantly, the bridge towards diagnosis methods for the discrete event systems (DES) is thrown.

Following this idea, Bayouth et al. [6] uses the residuals of each mode to define *mode signatures* and proposes to capture the mode signature changes caused by system transitions with so-called *signature-events*. A signature-event labeling a transition between discernible modes is stamped observable whereas it is stamped unobservable if the modes are not discernible. Signature-events are therefore used as a way to abstract the continuous dynamics while preserving the information useful for diagnosis. The behavior of the abstract hybrid system is then modeled by the so-called *behavior automaton* that generates a prefix-closed language over the original event alphabet enriched by these additional events. Based on this language, diagnosis of the hybrid system is cast into a discrete-event framework. In particular, the behavior automaton can be used to build a *diagnoser* [12, 27], which can support diagnosis as well as diagnosability analysis [3, 5].

Thereafter, Vento et al. [31] proposed to extend the behavior automaton method proposed in [6] so that it can account for non-structural faults. These are integrated in the behavior automaton as operation modes with unknown continuous dynamics, according to the idea introduced in [30]. The transition between a nominal and a non-structural faulty mode may turn observable if discernibility is fulfilled. As mentioned earlier, this property is determined through the analysis of a fault signature matrix associated to each nominal mode, which is based on the sensitivity concept [22].

Other extensions of the behavior automaton abstraction method have been proposed to improve diagnosis performances. A method based on parameter uncertainty using a passive robust strategy can be found in [32], where an adaptive threshold for residual evaluation is generated using the equivalence between the parity-space approach and input/output models. Another method proposed in [33] allows to diagnose hybrid systems using a diagnoser that reasons on components, considering

non-linear models and including multiple fault detection hypotheses. It has also been shown that the idea of using RRAs can benefit full hybrid state estimation schemas [26].

Despite the interest of the behavior automaton abstraction approach, its main issue is twofold:

- the number of states of the diagnoser grows exponentially with the number of states of the behavior automaton and it may require too much memory storage,
- generating the set of residuals for every mode is a tedious task that is generally unnecessary because the system remains restricted to a limited number of modes.

The latter problem was identified early and some solutions were proposed for specific cases [29]. The problem was considered as a whole later in [34, 35] that proposed an incremental method to avoid the task of building the entire behavior automaton and diagnoser offline. Diagnosis is performed by interpreting the events and measurements issued by the physical system directly on the initial hybrid automaton model. Mode tracking and diagnoser building are carried out synchronously, considering the possible current modes of the system and its successors. The idea is to build incrementally the behavior automaton and the corresponding diagnoser when events occur. These are recalculated whenever the system reaches a new operation mode. Everytime the diagnoser is updated, the set of events linking the current diagnoser state with its successors are taken into account to track the system mode. Assuming that the current mode is known, the set of residuals for the current mode and their successors are generated.

This approach named the *incremental behavior automaton abstraction* allows to construct the useful parts of the diagnoser developing only the branches that are required to explain the occurrence of incoming events. The resulting diagnoser adapts to the system operational life and is much less demanding in terms of memory storage than the entire diagnoser.

10.3 Hybrid System Diagnosis Framework

10.3.1 The Hybrid Automaton Model

The system is composed by a set of components, denoted by $COMP$, connected according to the system structure. We assume that the behavior of a component $\mathcal{C}_j \in COMP$ is governed by linear affine equations (algebraic or differential) and parametrized with the mode. Model equations depend on a set of physical variables, which are divided into two subsets, unknown and known variables. The discrete event behavior of each component is represented by an automaton.

The hybrid automaton model results from the parallel composition of the component automata and the parametrized linear equations of the system [3, 6, 21, 31].

The hybrid automaton is given by $HA = \langle \mathcal{Q}, \mathcal{X}, \mathcal{U}, \mathcal{Y}, \mathcal{F}, \mathcal{G}, \mathcal{H}, \Sigma, \mathcal{I} \rangle$, where:

- \mathcal{Q} is a set of modes. Each $q_i \in \mathcal{Q}$ with $|\mathcal{Q}| = n_q$ represents an operation mode, which may be a nominal mode or a structural or non-structural faulty mode of the system, i.e. $\mathcal{Q} = \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s} \cup \mathcal{Q}_{\mathcal{F}_{ns}}$.
- $q_0 \subseteq \mathcal{Q}$ is a set of initial modes.
- $\mathcal{X} \subseteq \mathbb{R}^{n_x}$ defines the continuous state space. $\mathbf{x}(k) \in \mathcal{X}$ is the discrete-time state vector and \mathbf{x}_0 the initial state vector.
- $\mathcal{U} \subseteq \mathbb{R}^{n_u}$ defines the continuous input space. $\mathbf{u}(k) \in \mathcal{U}$ is the discrete-time input vector.
- $\mathcal{Y} \subseteq \mathbb{R}^{n_y}$ defines the continuous output space. $\mathbf{y}(k) \in \mathcal{Y}$ is the discrete-time output vector.
- \mathcal{F} is the set of faults that can be partitioned into structural and non-structural faults, i.e. $\mathcal{F} = \mathcal{F}_s \cup \mathcal{F}_{ns}$. Every faulty mode $q_i \in \mathcal{Q}_{\mathcal{F}_s}$ or $q_i \in \mathcal{Q}_{\mathcal{F}_{ns}}$ has a corresponding fault $f_i \in \mathcal{F}_s$ or $f_i \in \mathcal{F}_{ns}$ and is associated with a fault event defined in the set $\Sigma_{\mathcal{F}}$. Modes associated with structural faults have a dynamic model specifying their continuous behavior, whereas those associated with non-structural faults have not. These faults are captured by the modification of the system dynamics they imply. They are modeled by a vector \mathbf{f}_{ns} impacting the equations of the other modes.
- \mathcal{G} defines a set of discrete-time state affine functions for each mode $q_i \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$:

$$\mathbf{x}(k+1) = \mathbf{A}_i \mathbf{x}(k) + \mathbf{B}_i \mathbf{u}(k) + \mathbf{F}_{x_i} \mathbf{f}_{ns}(k) + \mathbf{E}_{x_i} \quad (10.1)$$

where $\mathbf{A}_i \in \mathbb{R}^{n_x \times n_x}$, $\mathbf{B}_i \in \mathbb{R}^{n_x \times n_u}$ and $\mathbf{E}_{x_i} \in \mathbb{R}^{n_x \times 1}$ are the state matrices in mode q_i , $\mathbf{f}_{ns}(k)$ is the vector representing non-structural faults with \mathbf{F}_{x_i} being the fault distribution matrix. The case $\mathbf{f}_{ns}(k) = 0$ corresponds to a nominal or structural fault behavior.

- \mathcal{H} defines a set of discrete-time output affine functions for each mode $q_i \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$:

$$\mathbf{y}(k) = \mathbf{C}_i \mathbf{x}(k) + \mathbf{D}_i \mathbf{u}(k) + \mathbf{F}_{y_i} \mathbf{f}_{ns}(k) + \mathbf{E}_{y_i} \quad (10.2)$$

where $\mathbf{C}_i \in \mathbb{R}^{n_y \times n_x}$, $\mathbf{D}_i \in \mathbb{R}^{n_y \times n_u}$ and $\mathbf{E}_{y_i} \in \mathbb{R}^{n_y \times 1}$ are the output matrices in mode q_i and \mathbf{F}_{y_i} is the fault distribution matrix.

- $\Sigma = \Sigma_s \cup \Sigma_c \cup \Sigma_{\mathcal{F}}$ is a set of events. Spontaneous mode switching events (Σ_s), input events (Σ_c) and fault events ($\Sigma_{\mathcal{F}} = \Sigma_{\mathcal{F}_s} \cup \Sigma_{\mathcal{F}_{ns}}$) are considered. Σ can be partitioned into $\Sigma_o \cup \Sigma_{uo}$ where Σ_o represents a set of observable events and Σ_{uo} represents a set of unobservable events. $\Sigma_{\mathcal{F}} \subseteq \Sigma_{uo}$, $\Sigma_c \subseteq \Sigma_o$ and $\Sigma_s \subseteq \Sigma_{uo} \cup \Sigma_o$.

- $\mathcal{T} : \mathcal{Q} \times \Sigma \rightarrow \mathcal{Q}$ is the transition function. The transition from mode q_i to mode q_j labeled with an event $\sigma \in \Sigma$ is denoted by $\mathcal{T}(q_i, \sigma) = q_j$ or by τ_{ij} when the event is of no interest.¹

Alternatively, the model given by Eqs. (10.1)–(10.2) can be expressed in input–output form using the delay operator which is denoted by p^{-1} and considering initial conditions equal to zero, as follows:

$$\mathbf{y}(k) = \mathbf{M}_i(p^{-1})\mathbf{u}(k) + \mathbf{\Upsilon}_i(p^{-1})\mathbf{f}_{ns}(k) + \mathbf{E}_{mi}(p^{-1}) \quad (10.3)$$

where:

$$\mathbf{M}_i(p^{-1}) = \mathbf{C}_i(p\mathbf{I} - \mathbf{A}_i)^{-1}\mathbf{B}_i + \mathbf{D}_i \quad (10.4)$$

$$\mathbf{\Upsilon}_i(p^{-1}) = \mathbf{C}_i(p\mathbf{I} - \mathbf{A}_i)^{-1}\mathbf{F}_{xi} + \mathbf{F}_{yi} \quad (10.5)$$

$$\mathbf{E}_{mi}(p^{-1}) = (\mathbf{C}_i(p\mathbf{I} - \mathbf{A}_i)^{-1}\mathbf{E}_{xi} + \mathbf{E}_{yi}) \frac{p}{p-1} \quad (10.6)$$

considering that $\mathbf{M}_i(p^{-1})$ represents the system input/output transfer function, $\mathbf{\Upsilon}_i(p^{-1})$ is the non-structural fault transfer function and $\mathbf{E}_{mi}(p^{-1})$ is associated with the terms \mathbf{E}_{xi} and \mathbf{E}_{yi} in the state space model.

The automaton for a component \mathcal{C} is defined by $DA_{\mathcal{C}} = \langle \mathcal{Q}_{\mathcal{C}}, \Sigma_{\mathcal{C}}, \mathcal{T}_{\mathcal{C}}, \Gamma_{\mathcal{C}} \rangle$, where $\mathcal{Q}_{\mathcal{C}}$ is the set of discrete modes, $\Sigma_{\mathcal{C}}$ is the set of events, $\mathcal{T}_{\mathcal{C}}$ is the transition function and $\Gamma_{\mathcal{C}} : \mathcal{Q}_{\mathcal{C}} \rightarrow 2^{\Sigma_{\mathcal{C}}}$ is the active event function. Events may be observable or unobservable like events corresponding to the occurrence of a structural fault. The active event function contains the set of all possible events $\sigma_{\mathcal{C}} \in \Sigma_{\mathcal{C}}$ such that $\mathcal{T}_{\mathcal{C}}(q_{\mathcal{C}}, \sigma_{\mathcal{C}})$ is defined.

In this work, the hybrid model is built by combining the operation modes of the component models through the parallel composition of their automata [12]. Given two automata DA_1 and DA_2 , the parallel composition is defined as:

$$DA_{\mathcal{C}_1} || DA_{\mathcal{C}_2} = Ac(\mathcal{Q}_{\mathcal{C}_1} \times \mathcal{Q}_{\mathcal{C}_2}, \Sigma_1 \cup \Sigma_2, \mathcal{T}_{Ac}, \Gamma_{1||2}, (q_{01}, q_{02}))$$

$$\mathcal{T}_{Ac}((q_1 \cdot q_2), \sigma_{\mathcal{C}}) = \begin{cases} (\mathcal{T}_1(q_1, \sigma_{\mathcal{C}}), \mathcal{T}_2(q_2, \sigma_{\mathcal{C}})) & \text{if } \sigma_{\mathcal{C}} \in \Gamma_1(q_1) \cap \Gamma_2(q_2) \\ (\mathcal{T}_1(q_1, \sigma_{\mathcal{C}}), q_2) & \text{if } \sigma_{\mathcal{C}} \in \Gamma_1(q_1) \setminus \Sigma_2 \\ (q_1, \mathcal{T}_2(q_2, \sigma_{\mathcal{C}})) & \text{if } \sigma_{\mathcal{C}} \in \Gamma_2(q_2) \setminus \Sigma_1 \\ \text{undefined} & \text{otherwise} \end{cases} \quad (10.7)$$

where $Ac(G)$ is a unary operator that involves taking the accessible part of G from its initial state.

On the other hand, the system model is given by the sets of equations describing the component behaviors and their interconnections. The component equations are

¹It is assumed that there is only one transition from a given mode q_i to a given mode q_j .

Table 10.1 Transition function defined for the *HA*

		Destination modes		
		$\mathcal{Q}_{\mathcal{N}}$	$\mathcal{Q}_{\mathcal{F}_s}$	$\mathcal{Q}_{\mathcal{F}_{ns}}$
Source modes	$\mathcal{Q}_{\mathcal{N}}$	$\Sigma_s \cup \Sigma_c$	$\Sigma_{\mathcal{F}_s}$	$\Sigma_{\mathcal{F}_{ns}}$
	$\mathcal{Q}_{\mathcal{F}_s}$	–	–	$\Sigma_{\mathcal{F}_{ns}}$
	$\mathcal{Q}_{\mathcal{F}_{ns}}$	–	–	–

parametrized with the operation mode. The state space model of each mode in the hybrid model is hence represented by (10.1)–(10.2), where state space matrices are instantiated depending on the modes obtained through the composition [1, 7].

Table 10.1 summarizes when the transition function in *HA* is possibly defined. The symbol ‘–’ indicates that the transition is not possible. Notice that transitions between nominal modes and transitions from structural faulty modes to non-structural faulty modes are possible. Nevertheless, transitions from faulty modes to nominal modes are not possible neither transitions from non-structural faulty modes.

Another aspect to consider is that the composition of component automata is done for operation modes that belong to $\mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$. Non-structural faulty modes are added a posteriori to the resulting hybrid automaton. Thus, the number of non-structural modes associated with each mode in $\mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$ equals $|\mathcal{F}_{ns}|$.

10.3.2 Consistency Indicators

In the hybrid framework, diagnosis is achieved both from reported observable discrete events Σ_o and continuous measurements $(\mathbf{y}(k), \mathbf{u}(k))$. Referring to the latter, we adopt the common view of model-based diagnosis [9] and generate residuals for each mode associated with a dynamic model. These residuals are used to obtain consistency indicators.

Consider a mode $q_i \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$ with dynamic model of the form (10.1)–(10.2), then the set of residuals is given by:

$$\mathbf{r}_i(k) = \mathbf{y}(k) - \mathbf{G}_i(p^{-1})\mathbf{u}(k) - \mathbf{H}_i(p^{-1})\mathbf{y}(k) - \mathbf{E}_i(p^{-1}) \quad (10.8)$$

where $\mathbf{G}_i(p^{-1})$, $\mathbf{H}_i(p^{-1})$ and $\mathbf{E}_i(p^{-1})$ represent the input–output dynamic model for mode q_i . These transfer functions can be calculated using observers [23], for instance. Alternatively, the parity-space approach can also be used²[13]. In fact, the equivalence between the two approaches has been proved under certain conditions [17]. The observer model is given by:

$$\mathbf{G}_i(p^{-1}) = \mathbf{C}_i(p\mathbf{I} - \mathbf{A}_{oi})^{-1}\mathbf{B}_i + \mathbf{D}_i \quad (10.9)$$

$$\mathbf{H}_i(p^{-1}) = \mathbf{C}_i(p\mathbf{I} - \mathbf{A}_{oi})^{-1}\mathbf{L}_{oi} \quad (10.10)$$

$$\mathbf{E}_i(p^{-1}) = (\mathbf{C}_i(p\mathbf{I} - \mathbf{A}_{oi})^{-1}\mathbf{E}_{xi} + \mathbf{E}_{yi}) \frac{p}{p-1} \quad (10.11)$$

²Any residual generation method available in the literature could be used. See, for example, [9, 20].

where $\mathbf{A}_{oi} = \mathbf{A}_i - \mathbf{L}_{oi}\mathbf{C}_i$ and \mathbf{L}_{oi} is the observer gain.

Once the residuals have been generated, they are evaluated with the measurements against a threshold, providing consistency indicators of the following form:

$$\varphi_i^l(k) = \begin{cases} 0 & \text{if } |r_i^l(k)| \leq \tau_i^l \\ 1 & \text{if } |r_i^l(k)| > \tau_i^l \end{cases} \quad (10.12)$$

where $l \in \{1, \dots, n_{r_i}\}$, n_{r_i} is the number of residuals for mode q_i and τ_i^l is the threshold³ associated with residual $r_i^l(k)$. Consistency indicators are then gathered in a vector $\Phi_i(k) = [\varphi_i^1(k), \dots, \varphi_i^{n_{r_i}}(k)]$.

To detect and isolate non-structural faults, a theoretical fault signature matrix \mathbf{FS}_i for mode q_i is generated using the concept of fault sensitivity, which is determined by the expression:

$$\mathbf{\Lambda}_i(p^{-1}) = (\mathbf{I} - \mathbf{H}_i(p^{-1}))\mathbf{\Upsilon}_i(p^{-1}) \quad (10.13)$$

where $\mathbf{\Upsilon}_i$ is given by Eq. (10.5). Given the fault sensitivity of the j th residual with respect to the l th non-structural fault denoted as $\Lambda_i(j, l)$ (i.e., the element (j, l) of the sensitivity matrix $\mathbf{\Lambda}_i$), the element (j, l) of \mathbf{FS}_i is determined as follows:

$$\mathbf{FS}_i(j, l) = \begin{cases} 1 & \text{if } \Lambda_i(j, l) \neq 0 \\ 0 & \text{if } \Lambda_i(j, l) = 0 \end{cases} \quad (10.14)$$

$\mathbf{FS}_i(j, l)$ is 1 if the j th residual of mode q_i is sensitive to the l th fault, otherwise it is 0. For completeness one more column with zero signature is added representing the non-structural fault free case. If f_l is the l th non-structural fault, the theoretical fault signature of f_l , denoted as \mathbf{FS}_i^f , is then given by $\mathbf{FS}_i(\bullet, l)$.

10.3.3 Mode Discernibility Analysis

Discernibility of two modes assesses whether these modes can be distinguished based on continuous measurements. This property is key for hybrid system mode tracking. In this section, we analyze discernibility for the general situation in which modes may be nominal or faulty, structurally or non-structurally. Starting with the definition proposed by Cocquempot et al. [14], we derive operational conditions based on the continuous dynamic models of the modes or on the deviations that they imply on the continuous dynamics of the hybrid system. A formal proof of all the propositions stated in this section can be found in [34].

³The thresholds can be decided using any of the standard FDI threshold generation approaches [9].

Definition 1 Two modes q_i and q_j are discernible iff there exists at least a couple of signals $(\mathbf{u}(k), \mathbf{y}(k))$ consistent with mode q_i that are not consistent with mode q_j and vice versa.

From the properties of residuals, we have the following result.

Proposition 1 Two modes q_i and q_j are non-discernible iff the consistency indicators of the two modes satisfy $\Phi_i(k) = \Phi_j(k)$ for any $(\mathbf{u}(k), \mathbf{y}(k))$ and any time instant k .

We define the following function:

$$f_{disc} : \mathcal{Q} \times \mathcal{Q} \rightarrow \{0, 1\} \quad (10.15)$$

where $f_{disc}(q_i, q_j) = 1$ iff the two modes q_i and q_j are discernible, and $f_{disc}(q_i, q_j) = 0$ otherwise. If two modes q_i, q_j are discernible, we also say that the pair of modes (q_i, q_j) is discernible.

The following definitions are related to discernibility.

Definition 2 Considering HA , a mode change $q_i \rightarrow q_j$ is detectable at time instant k if q_i and q_j are discernible according to Definition 1.

Definition 3 Considering HA , two mode changes, $q_i \rightarrow q_j$ and $q_i \rightarrow q_l$, are isolable if the following conditions are satisfied at time instant k .

1. Both mode changes are detectable according to Definition 2, or equivalently both (q_i, q_j) and (q_i, q_l) are discernible.
2. The pair of modes (q_l, q_j) is discernible according to Definition 1.

The conditions guarantying discernibility depend on the pair of modes considered in HA . Three cases can be outlined.

10.3.3.1 Case 1

Let us consider a pair of modes that have an associated continuous dynamic model of the form (10.1)–(10.2), represented in input–output form (10.3). We have the following result.

Proposition 2 Two modes $\{q_i, q_j\} \subseteq \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$ are non-discernible if the following conditions are fulfilled:

$$\mathbf{M}_i(p^{-1}) = \mathbf{M}_j(p^{-1}) \quad (10.16)$$

$$\mathbf{E}_{m_i}(p^{-1}) = \mathbf{E}_{m_j}(p^{-1}) \quad (10.17)$$

where \mathbf{M}_i , \mathbf{E}_{m_i} , \mathbf{M}_j and \mathbf{E}_{m_j} correspond to the input/output model matrices given by (10.4) and (10.6), respectively.

Conditions (10.16) and (10.17) guarantee that consistency indicators of the two modes satisfy $\Phi_i(k) = 0$ and $\Phi_j(k) = 0$ for any $(\mathbf{u}(k), \mathbf{y}(k))$ and any time instant k , hence proving non-discernibility of the two modes with reference to Proposition 1.

The discernibility function can be evaluated using conditions (10.16) and (10.17), which rely on the system model (10.1) and (10.2) represented in input–output form (10.3).

10.3.3.2 Case 2

Let us consider a pair of modes corresponding to non-structural faults, that have a common predecessor mode. These modes do not have a continuous dynamic model but faults have a signature in the fault signature matrix.

The discernibility property involves comparing their corresponding fault signatures.

Proposition 3 *Two modes $\{q_{i_1}, q_{i_2}\} \subseteq \mathcal{Q}_{\mathcal{F}_{ns}}$ associated to non-structural faults f_{ns_1} and f_{ns_2} respectively, such that $\mathcal{T}(q_i, \sigma_{f_{ns_1}}) = q_{i_1}$ and $\mathcal{T}(q_i, \sigma_{f_{ns_2}}) = q_{i_2}$ for a given mode $q_i \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$ and $\{\sigma_{f_{ns_1}}, \sigma_{f_{ns_2}}\} \subseteq \Sigma_{\mathcal{F}_{ns}}$, are non-discernible if their residual fault sensitivities satisfy*

$$\Lambda_i^{f_{ns_1}}(p^{-1}) = \Lambda_i^{f_{ns_2}}(p^{-1}) \neq \mathbf{0} \quad (10.18)$$

10.3.3.3 Case 3

Let us consider a mode that has a continuous dynamic model and another one which has not, with a common predecessor mode. We have the following result.

Proposition 4 *A mode $q_j \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$ and a mode $q_{i_\alpha} \in \mathcal{Q}_{\mathcal{F}_{ns}}$ associated with non-structural fault f_{ns_α} , such that $\mathcal{T}(q_i, \sigma) = q_j$ and $\mathcal{T}(q_i, \sigma_{f_{ns_\alpha}}) = q_{i_\alpha}$ for a given mode $q_i \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$, $\sigma \in \Sigma_s \cup \Sigma_c \cup \Sigma_{\mathcal{F}_s}$ and $\sigma_{f_{ns_\alpha}} \in \Sigma_{\mathcal{F}_{ns}}$, are non-discernible if the following conditions are fulfilled:*

$$\mathbf{M}_j(p^{-1}) - \mathbf{M}_i(p^{-1}) = \Lambda_i^{f_{ns_\alpha}}(p^{-1}) \quad (10.19)$$

$$\mathbf{E}_{m_i}(p^{-1}) = \mathbf{E}_{m_j}(p^{-1}) \quad (10.20)$$

$$\mathbf{u}(k) = \mathbf{f}_{ns_\alpha}(k) \quad (10.21)$$

Notice that the discernibility condition makes use of the sensitivity function of the non-structural faulty mode calculated through the dynamic model of its predecessor mode.

10.3.4 The Behaviour Automaton Abstraction

The behavior automaton is a finite state generator of the language $L(HA)$ resulting from abstracting the continuous dynamics in terms of discrete signature-events [2, 6]. The behavior automaton is defined by $B = \langle \overline{\mathcal{Q}}, \overline{\Sigma}, \overline{\mathcal{T}}, \overline{q}_0 \rangle$.

- $\overline{\mathcal{Q}} = \mathcal{Q} \cup \mathcal{Q}^t$ is a set of discrete states where:
 - \mathcal{Q} is the set of modes of HA;
 - \mathcal{Q}^t is a set of transient modes.
- \overline{q}_0 is the initial state.
- $\overline{\Sigma} = \Sigma \cup \Sigma^{Sig}$ is the set of events where:
 - Σ is the set of events of HA;
 - Σ^{Sig} is a set of signature-events generated when two modes are discernible according to (10.15).
- $\overline{\mathcal{T}} : \overline{\mathcal{Q}} \times \overline{\Sigma} \mapsto \overline{\mathcal{Q}}$ is the partial transition function of the behavior automaton.

B is built following Algorithm 1 based on the discernibility properties presented in Sect. 10.3.3. The algorithm previously requires the set of modes of HA to be partitioned into subsets of non-discernible modes, i.e. $\mathcal{Q}_{disc} = \{\mathcal{Q}_{v_1}, \dots, \mathcal{Q}_{v_N}\}$.

To explore every mode $q_i \in \mathcal{Q}$ in HA the set $Succs_{HA}(q_i) = \{q_j \in \mathcal{Q} : \exists \sigma \in \Sigma, T(q_i, \sigma) = q_j\}$ is defined. The transitions of HA are integrated into B and the discernibility between the source and destination modes is studied whenever necessary (see Sect. 10.3.3). If a transition in HA is labelled by an observable event the transition is kept in B (see line 14). Otherwise, the discernibility property is evaluated between the pair of modes (q_i, q_j) (see line 16). If the two modes are discernible then a *transient mode*⁴ is added between these modes (see lines 17–23). The outgoing transition of the transient mode is associated with a signature-event δ (see line 18) indicating that the mode change can be observed by means of consistency indicators. This signature-event is indexed according to the case of discernibility of the two modes. Otherwise, if the two modes are non-discernible the original transition is kept in B labeled with its corresponding unobservable event (see line 28).

$$\delta = f_{Sig_ev} : \overline{\mathcal{Q}} \times \overline{\mathcal{Q}} \rightarrow \Sigma^{Sig} \quad (10.22)$$

⁴The transient mode is the way to account for the hybrid automaton HA dwell time requirement [7].

Algorithm 1: B_Builder(BA)

```

1:  $\mathcal{L}_h = \emptyset$ .
2: for all  $q_i \in \mathcal{Q}$  do
3:    $\mathcal{L}_h = \mathcal{L}_h \cup \{q_i\}$ 
4: end for
5: while  $\mathcal{L}_h \neq \emptyset$  do
6:    $\mathcal{L}_h = \mathcal{L}_h \setminus \{q_i\}$ 
7:   for all  $q_j \in Succ_{HA}(q_i)$  do
8:     if  $q_j \notin \mathcal{Q} \cap \mathcal{Q}$  then
9:        $\overline{\mathcal{Q}} = \{q_j\} \cup \overline{\mathcal{Q}}$ 
10:    end if
11:    Let  $\sigma$  is such as  $\mathcal{T}(q_i, \sigma) = q_j$  :
12:    switch ( $\sigma$ )
13:    case  $\sigma \in \Sigma_o$ :
14:       $\overline{\mathcal{T}}(q_i, \sigma) := q_j$ .
15:    case  $\sigma \in \Sigma_{uo}$ :
16:      if  $q_i$  and  $q_j$  are discernible according to (10.15) then
17:         $\mathcal{Q}' = \{q_{i-j}\} \cup \mathcal{Q}'$ .
18:         $\delta := f_{Sig\_ev}(q_i, q_j)$  according to (10.22).
19:        if  $\delta \notin \overline{\Sigma}$  then
20:           $\overline{\Sigma} = \{\delta\} \cup \overline{\Sigma}$ 
21:        end if
22:         $\overline{\mathcal{T}}(q_i, \sigma) := q_{i-j}$ .
23:         $\overline{\mathcal{T}}(q_{i-j}, \delta) := q_j$ .
24:      else
25:        if  $q_j \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$  then
26:           $\mathcal{L}_h = \mathcal{L}_h \cup \{q_j\}$ 
27:        end if
28:         $\overline{\mathcal{T}}(q_i, \sigma) := q_j$ .
29:      end if
30:    end switch
31:  end for
32: end while

```

$$f_{Sig_ev} \mapsto \left\{ \begin{array}{l} \delta_{v_i-v_j} \text{ if } f_{disc}(q_i, q_j) = 1 \text{ according to} \\ \text{Proposition 2, where} \\ q_i \in \mathcal{Q}_{v_i} \text{ and } q_j \in \mathcal{Q}_{v_j} \\ \text{with } \{\mathcal{Q}_{v_i}, \mathcal{Q}_{v_j}\} \subseteq \mathcal{Q}_{disc}, \\ \delta_{\mathcal{F}_{v_i}^\iota} \text{ if } f_{disc}(q_i, q_j) = 1 \text{ according to} \\ \text{Proposition 3, where } \delta_{\mathcal{F}_v^\iota} \text{ is associated} \\ \text{to a non-structural fault } f_l \text{ belonging} \\ \text{to a subset } \mathcal{F}_{v_i}^\iota \text{ with } \iota \in \mathcal{Z}^+, \\ \delta \\ \text{if } f_{disc}(q_i, q_j) = 1 \text{ according to} \\ \text{Proposition 4.} \end{array} \right. \quad (10.23)$$

The event label allows for distinguishing between the discernibility cases analyzed in Sect. 10.3.3, so that the diagnoser can be properly built.

10.3.5 The Hybrid Diagnoser

The hybrid diagnoser is a finite state machine $D = \langle \mathcal{Q}_D, \Sigma_D, T_D, q_{D_0} \rangle$, where:

- $q_{D_0} = \{q_0, \emptyset\}$ is the initial state of the hybrid diagnoser, which is assumed to correspond to a nominal system mode;
- \mathcal{Q}_D is a set of the hybrid diagnoser states. An element $q_D \in \mathcal{Q}_D$ is a set of the form $q_D = \{(q_1, l_1), (q_2, l_2), \dots, (q_n, l_n)\}$, where $q_i \in \mathcal{Q}$ and $l_i \in \Delta_{\mathcal{F}}$ where $\Delta_{\mathcal{F}}$ defines the power set of fault labels $\Delta_{\mathcal{F}} = \Delta_{\mathcal{F}_s} \cup \Delta_{\mathcal{F}_{ns}}$ with $\Delta_{\mathcal{F}_s} = \{f_1, \dots, f_\gamma, \emptyset\}$, and $\Delta_{\mathcal{F}_{ns}} = \{f_1^*, \dots, f_\mu^*\}$ respectively, $\gamma + \mu$ is the total number of fault combinations and $\gamma, \mu \in \mathcal{Z}^+$. In $\Delta_{\mathcal{F}}$, \emptyset represents the nominal behavior;
- $\Sigma_D = \bar{\Sigma}_o$ is the set of all observable events in B ;
- $\mathcal{T}_D : \mathcal{Q}_D \times \Sigma_o \mapsto \mathcal{Q}_D$ is a partial transition function of the hybrid diagnoser.

The transition function \mathcal{T}_D can be calculated according to the procedure described in [12, 27], from the behavior automaton B . According to this procedure, a hybrid diagnoser is built like an observer automaton with the difference that labels reporting whether fault events have occurred are attached to the hybrid diagnoser states.

10.3.6 Mode Tracking Logic

At any time instant k , the current hybrid diagnoser state provides the set called the *belief mode* and denoted by $q_D(k)$. The hybrid system can be operating in any of the modes in the current *belief mode*. Given a set of observations of the system, a mode change can be expected if consistency indicators of the current mode have changed. The minimal time to observe that change is given by the dwell time requirement, which guarantees that residuals, and hence consistency indicators, can be properly computed [3].

The following results provide conditions for transition detection and transition identification. A formal proof of all the propositions stated in this section can be found in [34].

Proposition 5 *If $\Phi_i(k-1) = 0$ and $\Phi_i(k) \neq 0$, then a transition from $q_i \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$ to another mode is suspected at time instant k .*

Proposition 5 is used to decide whether a mode change has occurred by monitoring the set of consistency indicators of the possible current modes, i.e. modes in the belief mode.

Proposition 6 *Assuming that HA is in mode q_i and a transition has been suspected at time instant k according to Proposition 5, then:*

1. if $\Phi_i(k) = \mathbf{FS}_i(\bullet, f_j)$, then a transition to $q_j \in \mathcal{Q}_{\mathcal{F}_{ns}}$ is detected at time instant k .
2. if $\Phi_j(k) = 0$ and $\mathcal{T}(q_i, \tau_{ij}) = q_j$, then a transition to $q_j \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$ is detected at time instant k .

Let us notice that Proposition 6 does not necessarily identify a unique mode q_j . In particular, conditions (1) or (2) of Proposition 6 may be satisfied for more than one index, which corresponds to the cases of ambiguous non-structural faulty modes and ambiguous structural faulty modes, respectively. This logic is used to identify the set of possible mode changes.

10.4 Incremental Hybrid System Diagnosis

10.4.1 Incremental Diagnosis Architecture

The methodology described in Sect. 10.3 requires building and storing offline the entire hybrid automaton model, behavior automaton and hybrid diagnoser. This section proposes an enhanced diagnoser that is built incrementally online. Indeed, diagnosis is directly performed by interpreting the events and measurements issued by the physical system on the hybrid automaton model. This interpretation allows us to incrementally build the useful parts of the diagnoser, developing only the branches that are required to explain the occurrence of incoming events. Generally, a hybrid system operates in a small region compared to the entire behavioral space defined by the hybrid automaton states. A significant gain can hence be expected from the proposed approach. See [34] for an extended description of this methodology, including its complexity analysis that proves its benefits under low memory usage requirement at the expense of a negligible execution time penalty.

Mode tracking and diagnoser building are carried out synchronously, considering the possible current modes of the system and their successors. The original idea is to incrementally build the hybrid diagnoser when events occur. This includes building the hybrid model incrementally through the composition of the automata describing the system component behaviors. The set of linear equations constituting the continuous model of the components are parameterized as a function of the mode.

The hybrid model and the behavior automaton are recalculated whenever the system reaches a new operation mode. Then, the diagnoser is updated and the set of events linking the current diagnoser state with their successors are taken into account to track the system mode. Assuming that the current mode is known, the set of residuals for the current mode and their successors are generated. Next, observable events (i.e. input events and signature-events) are detected and processed by the hybrid diagnoser, that reports the current diagnoser state and the possible occurrence of a fault.

10.4.2 Incremental Hybrid System Diagnosis Framework

An incremental version of HA , B and D are defined and labelled as HA^k , B^k and D^k . The dependence on time is captured by indexing with the time instant k .

HA^k is built by the parallel composition of component automata from (10.7) along with parametrized equations which allow one to obtain the model equations in (10.1) and (10.2) for the modes that are introduced. At any time instant k , the system can be operating in one of the modes of the set called the *belief mode* and denoted by $q_D(k)$. Algorithm 2 takes $q_D(k)$ as input and incrementally builds the hybrid model whenever there is a change in the system, i.e. when the consistency indicators of one of the modes in the belief state change value or when an observable event occurs.

Algorithm 2: Incremental_HA_Builder($q_D(k)$)

```

1:  $\mathcal{L}_h = \emptyset$ 
2: for all  $q_i \in q_D(k)$  such that  $q_i \in \mathcal{Q}_{\mathcal{N}}^k \cup \mathcal{Q}_{\mathcal{F}_s}^k$  do
3:    $\mathcal{L}_h = \mathcal{L}_h \cup \{q_i\}$ 
4: end for
5: while  $\mathcal{L}_h \neq \emptyset$  do
6:    $\mathcal{L}_h = \mathcal{L}_h \setminus \{q_i\}$ 
7:   for all  $f_w \in \mathcal{F}_{ns}$  do
8:      $\mathcal{Q}^k := \{q_{f_w i}\} \cup \mathcal{Q}^{k-1}$ .
9:      $\mathcal{T}(q_i, \sigma_{f_w}) = q_{f_w i}$ .
10:  end for
11:  Update the model by incremental parallel composition.
12:  for all  $\sigma_{\mathcal{M}} \in \Gamma_{Ac}(q_i)$  do
13:     $\mathcal{T}^k(q_i, \sigma_{\mathcal{M}}) := \mathcal{T}_{Ac}(q_i, \sigma_{\mathcal{M}})$ .
14:    if  $\sigma_{\mathcal{M}} \notin \Sigma^{k-1}$  then
15:       $\Sigma^k := \sigma_{\mathcal{M}} \cup \Sigma^{k-1}$ .
16:    end if
17:    if  $q_j \notin \mathcal{Q}^k$  then
18:       $\mathcal{Q}^k := \{q_j\} \cup \mathcal{Q}^{k-1}$ .
19:      Instantiate equations for this mode.
20:      Compute residual expression for  $\mathbf{r}_j(\bullet)$ .
21:      Classify  $q_j$  into  $\mathcal{Q}_{disc}$ .
22:      if  $q_j$  creates a new element  $Q_{v_j}$  in  $\mathcal{Q}_{disc}$  then
23:        Compute  $\mathbf{FS}_{v_j}(\bullet)$ .
24:        Update and store in knowledge-base.
25:      end if
26:      if  $\sigma_{\mathcal{M}} \in \Sigma_{uo}$  then
27:        if  $(q_i, q_j)$  are non-discernible according to (10.15) then
28:           $\mathcal{L}_h = \mathcal{L}_h \cup \{q_j\}$ 
29:        end if
30:      end if
31:    end if
32:  end for
33: end while

```

As can be seen in Algorithm 2, the branches generation of HA^k depends on the discernibility property between the current mode and its successors. If some of them are non-discernible it implies that the event labeling the transition is unobservable. The iterations of the algorithm stop when HA^k is such that all branches end with an observable event avoiding uncertainty in the model. In the first iteration, HA^k initially must contain at least the initial mode and its successors, assuming they are discernible.

Line 11 of Algorithm 2 updates the discrete part of HA^k using parallel composition. The parallel composition given by (10.7) is adapted to generate only the successor modes of a given mode q_i . The function provides the set of successor modes, the set of events and the transition function of this iteration. The elements generated in every parallel composition are gathered in HA^k . It is assumed that the incremental initial mode (HA_{init}) is known and it is generated before the diagnosis process starts.

In Algorithm 2, lines 7–10 add the successor non-structural faulty modes, whereas lines 13–16 add the successor nominal and structural faulty modes using the information provided by the incremental parallel composition. Lines 17–25 update the knowledge-base whenever a new mode is generated. In order to verify whether the branches of HA^k should be extended one more level further, the discernibility concerning the current mode and its successors is analyzed (see lines 26–30).

Algorithm 2 also examines conditions to recognize whether the current node has been previously considered (see line 17). Since the states of the hybrid automaton have a finite number of successor states, this algorithm is guaranteed to terminate in a finite number of steps.

The system model parameterized as a function of the operation mode is composed from the whole set of equations of the components and their interconnections (see line 19 of Algorithm 2). The state space model of each mode can be represented by (10.24) and (10.25). State space matrices depend on system parameters and they are instantiated for the modes obtained in the incremental composition.

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}_i \mathbf{x}(k) + \mathbf{B}_i \mathbf{u}(k) + \mathbf{F}_{x_i} \mathbf{f}(k) + \mathbf{E}_{x_i} \\ &\quad + \sum_{j=1}^{n_{S_i}} \mu_{x_i}^j S_i^j(\mathbf{x}(k), \mathbf{u}(k)) + \sum_{j=1}^{n_{D_i}} \psi_{x_i}^j D_i^j(\mathbf{x}(k), \mathbf{u}(k)) \end{aligned} \quad (10.24)$$

$$\begin{aligned} \mathbf{y}(k) &= \mathbf{C}_i \mathbf{x}(k) + \mathbf{D}_i \mathbf{u}(k) + \mathbf{F}_{y_i} \mathbf{f}(k) + \mathbf{E}_{y_i} \\ &\quad + \sum_{j=1}^{n_{S_i}} \mu_{y_i}^j S_i^j(\mathbf{x}(k), \mathbf{u}(k)) + \sum_{j=1}^{n_{D_i}} \psi_{y_i}^j D_i^j(\mathbf{x}(k), \mathbf{u}(k)) \end{aligned} \quad (10.25)$$

The S_i^j and D_i^j functions model the saturation and dead zone non-linearities that appear in the evolution and observation equations following the methodology in [7] (see Fig. 10.3). n_{S_i} and n_{D_i} denote the number of saturation and dead zone non-linearities introduced by a subset of components, $\mu_{x_i}^j$ and $\psi_{y_i}^j \in \mathcal{R}^{n_y} \times \mathcal{R}$, $\mu_{x_i}^j$ and $\psi_{x_i}^j \in \mathcal{R}^{n_x} \times \mathcal{R}$.

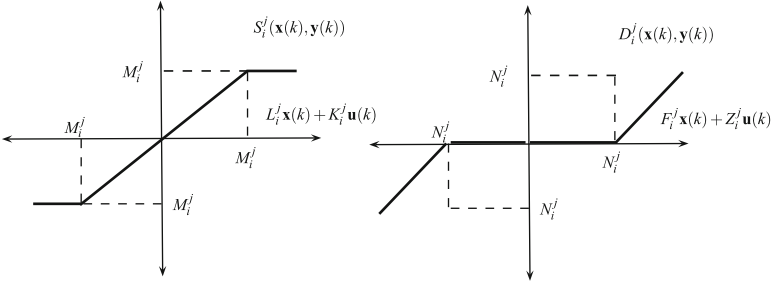


Fig. 10.3 Saturation and dead zone representation

$$S_i^j(\mathbf{x}(k), \mathbf{u}(k)) = \begin{cases} -M_i^j & \text{if } L_i^j \mathbf{x}(k) + K_i^j \mathbf{u}(k) < -M_i^j \\ L_i^j \mathbf{x}(k) + K_i^j \mathbf{u}(k) & \text{if } |L_i^j \mathbf{x}(k) + K_i^j \mathbf{u}(k)| \leq M_i^j \\ M_i^j & \text{if } L_i^j \mathbf{x}(k) + K_i^j \mathbf{u}(k) > M_i^j \end{cases} \quad (10.26)$$

where $M_i^j \in \mathcal{R}$ is a threshold, $L_i^j \in \mathcal{R} \times \mathcal{R}^{n_x}$ and $K_i^j \in \mathcal{R} \times \mathcal{R}^{n_u}$ are constant matrices.

$$D_i^j(\mathbf{x}(k), \mathbf{u}(k)) = \begin{cases} F_i^j \mathbf{x}(k) + Z_i^j \mathbf{u}(k) & \text{if } |F_i^j \mathbf{x}(k) + Z_i^j \mathbf{u}(k)| \leq N_i^j \\ 0 & \text{otherwise} \end{cases} \quad (10.27)$$

where $N_i^j \in \mathcal{R}$ is a threshold, $F_i^j \in \mathcal{R} \times \mathcal{R}^{n_x}$ and $Z_i^j \in \mathcal{R} \times \mathcal{R}^{n_u}$ are constant matrices.

The incremental behavior automaton B^k is built following an incremental implementation of the method explained in Sect. 10.3.4. The new version of Algorithm 1 explores HA^k assuming that the system can be operating in one of the modes q_i of the belief mode $q_D(k)$. Then, an exploration of each successor mode $q_j \in Succ_{SHA}(q_i)$, $q_i \in q_D$ is carried out.

The incremental hybrid diagnoser D^k is also built from the incremental behavior automaton B^k . D^k is updated after the occurrence of an observable event whenever there are behavior automaton states to be introduced that have not been previously visited. The part of the hybrid diagnoser obtained takes into account only the possible successor states and transitions that may occur next.

10.5 Application Case Study

10.5.1 Barcelona Sewer Network

To illustrate the method, a representative part of the Barcelona sewer network presented in [22] is used. Sewer networks present several elements exhibiting numerous operating modes depending on the sewer flows. Sewer networks may be

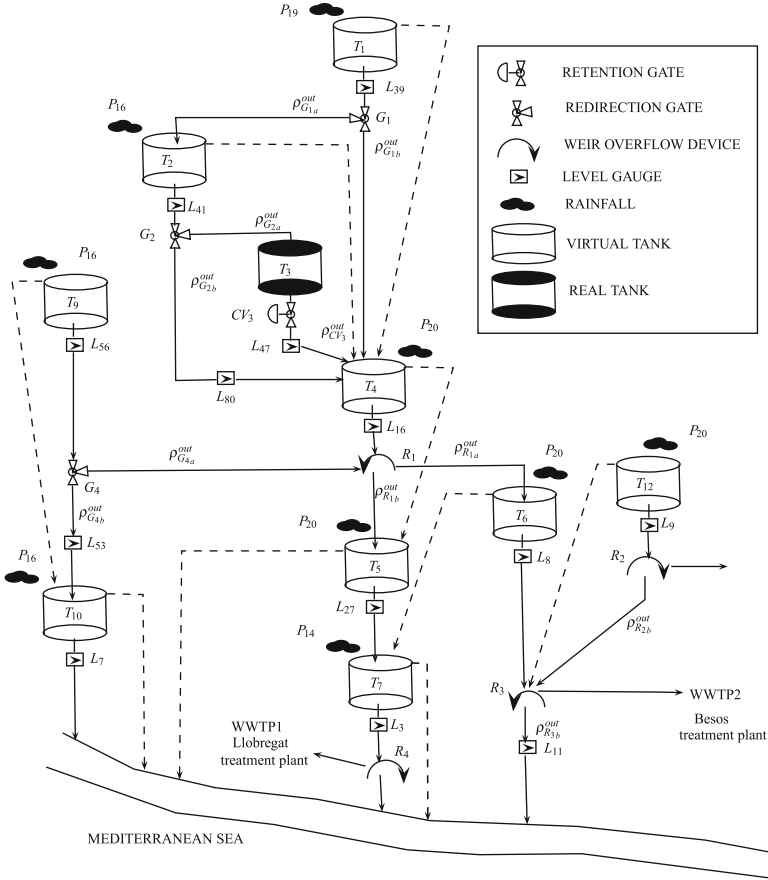


Fig. 10.4 A representative part of the sewer network

modeled using the virtual tank modeling approach. Therefore, the decomposition of the sewer network in catchments looks like what is shown in Fig. 10.4. The elements that appear in the sewer are: nine virtual tanks, one real tank, three redirection gates, one retention gate, one four rain gauges to measure the rain intensity and ten limnimeters to measure the sewer level. The control gates are commanded by a controller where actions are open gate or close depending on the flow in the sewer.

10.5.2 Hybrid Modeling

A hybrid automaton model can be obtained to represent the hybrid phenomena present in the network associated with the virtual tanks and the control gates. As proposed by our incremental method, the hybrid model is obtained incrementally

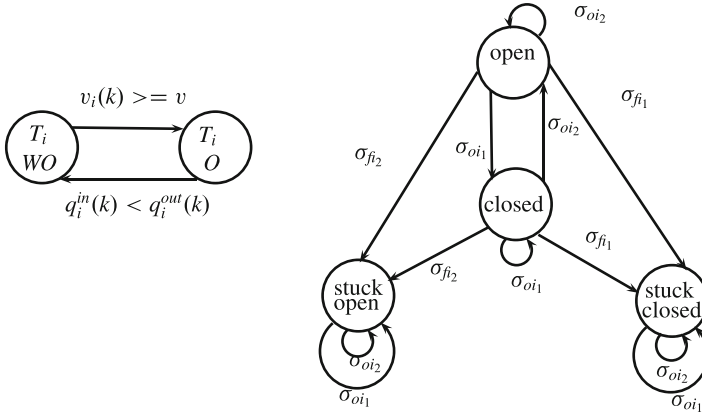


Fig. 10.5 Component Automata

from the automata for each component. The general automaton for a virtual tank is given by two discrete states: overflow (*o*) and non-overflow (*wo*) as is shown in Fig. 10.5 (left side). Regarding the control gates, there are four discrete states, the nominal behaviors (open or closed) and the faulty behaviors (stuck open (*so*) or stuck closed *sc*) such as shown in Fig. 10.5 (right side).

The elements of the sewer can be described by the set of equations below according to the component configuration. The dynamic model of the virtual tank is given by the following discrete-time equation representing the water volume:

$$T_i : \quad v_i(k + 1) = v_i(k) + \Delta t(\varrho_i^{in}(k) - \varrho_i^{out}(k) - \varrho_i^{des}(k))$$

with $i \in \{0, 1\}$. The overflow is given by:

$$\varrho_i^{des}(k) = \begin{cases} \varrho_i^{in}(k) - \varrho_i^{out}(k) & \text{if } v_i(k) \geq \bar{v}_i \\ 0 & \text{otherwise} \end{cases} \quad (10.28)$$

The input flow associated with a virtual tank is given by:

$$\varrho_i^{in} = \varrho_i^{pluv}(k) + \sum_{h=1}^H \varrho_i^{outh}(k) + \sum_{l=1}^L \varrho_i^{desl}(k) \quad (10.29)$$

where $\varrho_i^{pluv}(k) = S_i \phi_i u_i(k)$ is associated with the rain intensity, $\varrho_i^{outh}(k)$ corresponds to all the output flows of the other tanks pouring into the tank T_i and $\varrho_i^{desl}(k)$ corresponds to all overflows pouring into the tank T_i and $h, l \in \mathcal{Z}^+$.

The output flow for every tank is given by:

$$Q_i^{out}(k) = \begin{cases} \beta_i v_i(k) & \text{if } Q_i^{in}(k) < Q_i^{out}(k) \\ \beta_i \bar{v}_i & \text{if } v_i(k) \geq \bar{v}_i \end{cases} \quad (10.30)$$

The relation between level and volume and the measurements provided by the sensors are described by the equations below:

$$L_i(k) = \frac{\beta_i}{M_i} v_i(k) \quad (10.31)$$

The input flow to a control gate is divided into two output flows where the values depend on the position: open ($\alpha_j = 0$) or closed ($\alpha_j = 1$).

$$Q_{G_j}^{out}(k) = \begin{cases} Q_{aG_j}(k) = (1 - \alpha_j) Q_{G_j}^{in}(k) \\ Q_{bG_j}(k) = \alpha_j Q_{G_j}^{in}(k) \end{cases} \quad (10.32)$$

The composition is based on the automata of virtual tanks and control gates.

10.5.3 Hybrid System Diagnosis

10.5.3.1 Design

A small part of the sewer network considered as case study is used for illustrating the proposed hybrid diagnosis approach in detail (see Fig. 10.6).

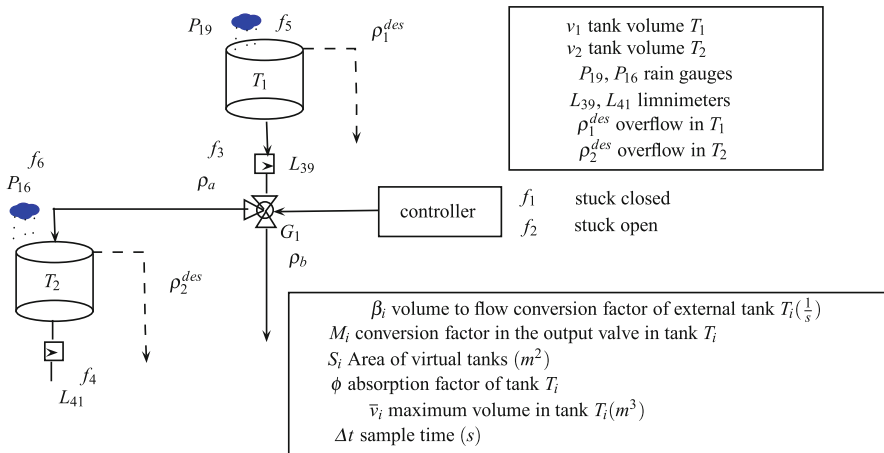


Fig. 10.6 Small part of the sewer network

Table 10.2 Type of events in HA

Event	Action	Observable	Type	Code
$uo1$	$v_1 \geq \bar{v}_2$	Not	Spontaneous	1
$uo2$	$\varrho_1^{in} < \varrho_1^{out}$	Not	Spontaneous	2
$uo3$	$v_2 \geq \bar{v}_2$	Not	Spontaneous	3
$uo4$	$\varrho_2^{in} < \varrho_2^{out}$	Not	Spontaneous	4
$o1$	Close redirection gate	Yes	Controlled	5
$o2$	Open redirection gate	Yes	Controlled	6
$f1$	Stuck closed	Not	Structural fault event	7
$f2$	Stuck open	Not	Structural fault event	8
$f3$	Fault in sensor L_{39}	Not	Non-structural fault event	9
$f4$	Fault in sensor L_{47}	Not	Non-structural fault event	10
$f5$	Fault in sensor P_{19}	Not	Non-structural fault event	11
$f6$	Fault in sensor P_{16}	Not	Non-structural fault event	12

In all, there are three components that can be described by an automaton: the two virtual tanks and the redirection gate. Structural faults are associated with faults in the redirection gate (stuck open and stuck closed). Non-structural faults are associated with faults in output and input sensors ($L_{39}, L_{41}, P_{19}, P_{16}$).

Events associated to the component automata and non-structural faults are detailed in Table 10.2.

The entire hybrid automaton (see Fig. 10.7) is obtained from parallel composition of the component automata. The hybrid automaton is composed by eight nominal modes and eight faulty modes (related to structural faults).

The continuous dynamical model for each mode $q_i \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$ is provided in Table 10.3. Notice that modes q_1 and q_9 have an equivalent dynamical model as well as modes q_5 and q_{10} . In the three last rows, when an overflow is present in any of both virtual tanks, model equations are equivalent even if the control gate is open or closed.

The output function is given by

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} \frac{\beta_1}{M_{39}} & 0 \\ 0 & \frac{\beta_2}{M_{41}} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \tag{10.33}$$

where the matrix \mathbf{C}_i is the same for all modes and $\mathbf{D}_i = \mathbf{0}$.

The sets of residuals for all modes are given in Table 10.4. There are two residuals per operation mode and five non-discernible mode sets as shown in Table 10.5.

For online diagnosis, only the set of residuals corresponding to active sets are computed. The active sets include modes in the belief mode and their successors belong to.

The following fault distribution matrices are defined:

$$\mathbf{F}_{y_i} = [\mathbf{0} \ \mathbf{I}] \qquad \mathbf{F}_{x_i} = [-\mathbf{B}_i \ \mathbf{0}]$$

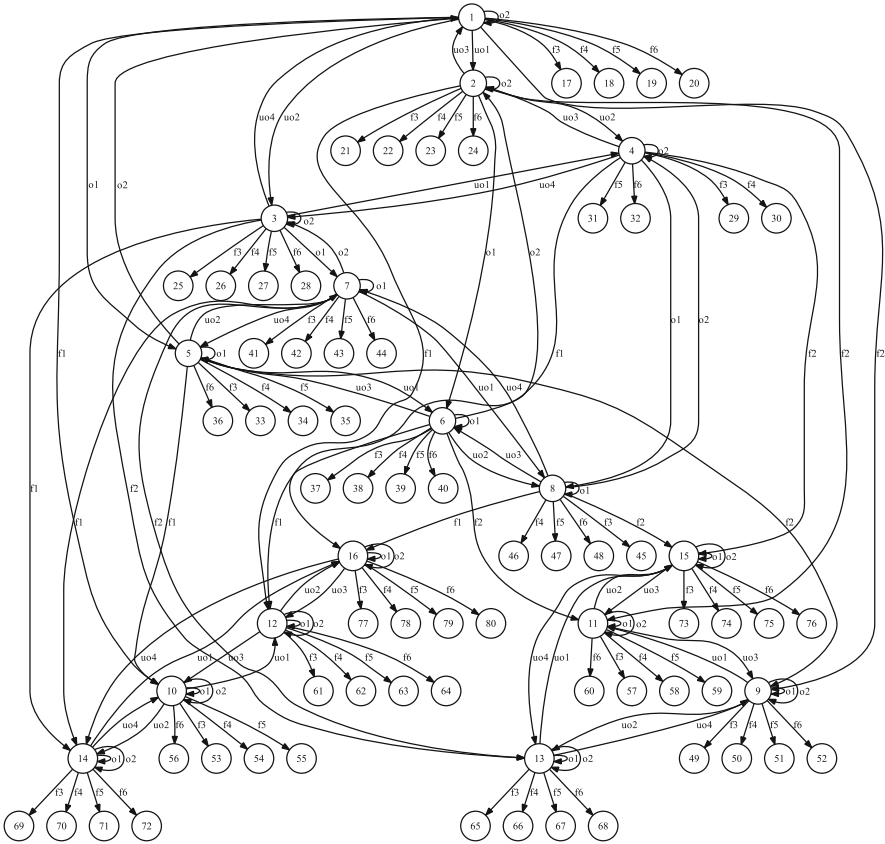


Fig. 10.7 Hybrid automaton model obtained using the component automata composition

These matrices are used to generate a fault signature matrix FS_{v_i} for every non-discernible mode set, applying Eq. (10.13).

Following Algorithm 1 the behavior automaton B is obtained. The corresponding automaton diagram has been omitted since it is too large: the number of modes is $|\mathcal{Q}| = 130$ and the number of explored transitions is 194.

The diagnoser without silent closure is shown in Fig. 10.8. The number of generated states is $|\mathcal{D}| = 59$ and the number of generated transitions is 188. The diagnoser was generated using DIADES tool [25]. The initial state is assumed known and nominal.

Table 10.3 State space matrices for each mode $q_i \in \mathcal{L}_{\mathcal{N}} \cup \mathcal{L}_{\mathcal{F}_s}$

q_i	Gate open and stuck open ($\alpha_1 = 1$)				Gate closed and stuck closed $\alpha_1 = 0$			
	\mathbf{A}_i	\mathbf{B}_i	\mathbf{E}_{α_i}	q_i	\mathbf{A}_i	\mathbf{B}_i	\mathbf{E}_{α_i}	q_i
1,9	$\begin{bmatrix} 1 - \Delta t \beta_1 & 0 \\ \alpha_1 \Delta t \beta_1 & 1 - \Delta t \beta_2 \end{bmatrix}$	$\begin{bmatrix} \Delta t S_1 \varphi_{19} & 0 \\ 0 & \Delta t S_2 \varphi_{16} \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	5,10	$\begin{bmatrix} 1 - \Delta t \beta_0 & 0 \\ 0 & 1 - \Delta t \beta_2 \end{bmatrix}$	$\begin{bmatrix} \Delta t S_1 \varphi_{19} & 0 \\ 0 & \Delta t S_2 \varphi_{16} \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	5,10
2,11	$\begin{bmatrix} 0 & 0 \\ 0 & 1 - \Delta t \beta_2 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & \Delta t S_2 \varphi_{16} \end{bmatrix}$	$\begin{bmatrix} \bar{v}_1 \\ \alpha_1 \Delta t \beta_0 \bar{v}_0 \end{bmatrix}$	6,12	$\begin{bmatrix} 0 & 0 \\ 0 & 1 - \Delta t \beta_2 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & \Delta t S_2 \varphi_{16} \end{bmatrix}$	$\begin{bmatrix} \bar{v}_1 \\ 0 \end{bmatrix}$	6,12
3,13	$\begin{bmatrix} 1 - \Delta t \beta_1 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \Delta t S_1 \varphi_{19} \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ \bar{v}_2 \end{bmatrix}$	7,14	$\begin{bmatrix} 1 - \Delta t \beta_1 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \Delta t S_1 \varphi_{19} \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ \bar{v}_2 \end{bmatrix}$	7,14
4,15	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \bar{v}_1 \\ \bar{v}_2 \end{bmatrix}$	8,16	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \bar{v}_1 \\ \bar{v}_2 \end{bmatrix}$	8,16

Table 10.4 Residual generation for all modes using input-output models

q_i	Gate open and stuck open ($\alpha = 1$)			q_i	Gate closed and stuck closed $\alpha = 0$		
	\mathbf{G}_i	\mathbf{H}_i	\mathbf{E}_i		\mathbf{G}_i	\mathbf{H}_i	\mathbf{E}_i
1,9	$\begin{bmatrix} \frac{\Delta t \beta_1 S_1 \phi_1}{M_{59} p} & 0 \\ 0 & \frac{\Delta t \beta_1 S_2 \phi_2}{M_{41} p} \end{bmatrix}$	$\begin{bmatrix} \frac{1 - \Delta t \beta_1}{p} & 0 \\ \frac{\Delta t \beta_2 \alpha_1 M_{59}}{M_{41} p} & \frac{1 - \Delta t \beta_2}{p} \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	5,10	$\begin{bmatrix} \frac{\Delta t \beta_1 S_1 \phi_1}{M_{59} p} & 0 \\ 0 & \frac{\Delta t \beta_2 S_2 \phi_2}{M_{41} p} \end{bmatrix}$	$\begin{bmatrix} \frac{1 - \Delta t \beta_1}{p} & 0 \\ 0 & \frac{1 - \Delta t \beta_2}{p} \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$
2,11	$\begin{bmatrix} 0 & 0 \\ 0 & \frac{\Delta t \beta_1 S_{16} \phi_1}{M_{41} p} \end{bmatrix}$	$\begin{bmatrix} \frac{1 - \Delta t \beta_1}{p} & 0 \\ \frac{\Delta t \beta_1 \alpha_1 M_{59}}{M_{41} p} & \frac{1 - \Delta t \beta_2}{p} \end{bmatrix}$	$\begin{bmatrix} \frac{\overline{v}_0 \beta_1}{M_{59} p} \\ \frac{\Delta t \beta_2 \alpha \beta_0 \overline{v}_1}{M_{41} (p - 1 + \Delta t \beta_1)} \end{bmatrix}$	6,12	$\begin{bmatrix} 0 & 0 \\ 0 & \frac{\Delta t \beta_2 S_2 \phi_2}{M_{41} p} \end{bmatrix}$	$\begin{bmatrix} \frac{1 - \Delta t \beta_1}{p} & 0 \\ 0 & \frac{1 - \Delta t \beta_2}{p} \end{bmatrix}$	$\begin{bmatrix} \frac{\overline{v}_1 \beta_1}{M_{59} p} \\ 0 \end{bmatrix}$
3,13	$\begin{bmatrix} \frac{\Delta t \beta_1 S_1 \phi_1}{M_{59} p} & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \frac{1 - \Delta t \beta_1}{p} & 0 \\ \frac{\Delta t \beta_1 \alpha M_{59}}{M_{41} p} & \frac{1 - \Delta t \beta_2}{p} \end{bmatrix}$	$\begin{bmatrix} 0 \\ \frac{\overline{v}_2 \beta_2}{M_{41} p} \end{bmatrix}$	7,14	$\begin{bmatrix} \frac{\Delta t \beta_0 S_1 \phi_1}{M_{59} p} & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \frac{1 - \Delta t \beta_1}{p} & 0 \\ 0 & \frac{1 - \Delta t \beta_2}{p} \end{bmatrix}$	$\begin{bmatrix} 0 \\ \frac{\overline{v}_1 \beta_2}{M_{41} p} \end{bmatrix}$
4,15	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \frac{\overline{v}_1 \beta_1}{M_{59} p} \\ \frac{\overline{v}_2 \beta_2}{M_{41} p} \end{bmatrix}$	8,16	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} \frac{\overline{v}_1 \beta_1}{M_{59} p} \\ \frac{\overline{v}_2 \beta_2}{M_{41} p} \end{bmatrix}$

Table 10.5 Non-discernible mode sets, \mathcal{Q}_{disc}

Groups	Non-discernible modes
\mathcal{Q}_{v_1}	{1, 9}
\mathcal{Q}_{v_2}	{5, 10}
\mathcal{Q}_{v_3}	{2, 6, 11, 12}
\mathcal{Q}_{v_4}	{3, 7, 13, 14}
\mathcal{Q}_{v_5}	{4, 8, 15, 16}

10.5.4 Incremental Hybrid System Diagnosis

Applying Algorithm 2 for the first time, the initial incremental hybrid model HA_{init} is obtained.

Next, applying the incremental implementation of Algorithm 1 to HA_{init} , the initial incremental behavior automaton B_{init} is obtained (see Fig. 10.9). Modes in dashed line correspond to the transient modes generated evaluating the discernability property. The generated signature-events are δ_{13} , δ_{14} , δ_{12} , $\delta_{\mathcal{F}_{v_1}^1}$, $\delta_{\mathcal{F}_{v_1}^2}$, $\delta_{\mathcal{F}_{v_1}^3}$. Transitions in dashed line show that the destination mode is a faulty mode. Modes q_1 and q_5 are linked by an observable event. Modes q_1 and q_9 do not have a transient mode between them because they are non-discernible. Modes labeled as q_i^j correspond to those non-structural faulty modes where i represents its path in HA^k and j is associated with the considered non-structural fault.

Note that B_{init} includes the events that may occur. The initial diagnoser (see Fig. 10.10) is obtained applying the procedure mentioned in Sect. 10.3.4 to B_{init} .

10.5.5 Results

Considering the whole sewer, assume that system follows the mode sequence $q_1 \rightarrow q_3 \rightarrow q_{214} \rightarrow q_{140} \rightarrow q_{211} \rightarrow q_5 \rightarrow q_{885}$ with a sample time of $\Delta t = 300$ s.

Mode q_1 refers to the situation in which no tank is being overflowing. Mode q_3 refers to T_1 being overflowing. q_{214} refers to T_2, T_4, T_5 and T_{12} being overflowing. q_{140} refers to T_2, T_4 and T_5 being overflowing. Mode q_{211} refers to T_5 and T_4 being overflowing and mode q_5 refers to T_5 being overflowing. The diagnoser must track the right mode sequence and detect and isolate the possible faults from an incrementally built behavior automaton B^k .

The set of residuals are only generated for modes that are visited in HA^k . In this way, the efficient use of memory is guaranteed. There is a set of ten residuals per group using the expression given by (10.8).

Figure 10.11 shows the set of residuals for the concerned modes in the sequence. Remark that the residuals of a given mode are consistent with measurements whenever system remains in this mode. The signature-events identified during the simulation are shown in black vertical dashed lines in Fig. 10.11. Events are such that a virtual tank reaching an overflow situation, a virtual tank leaving an

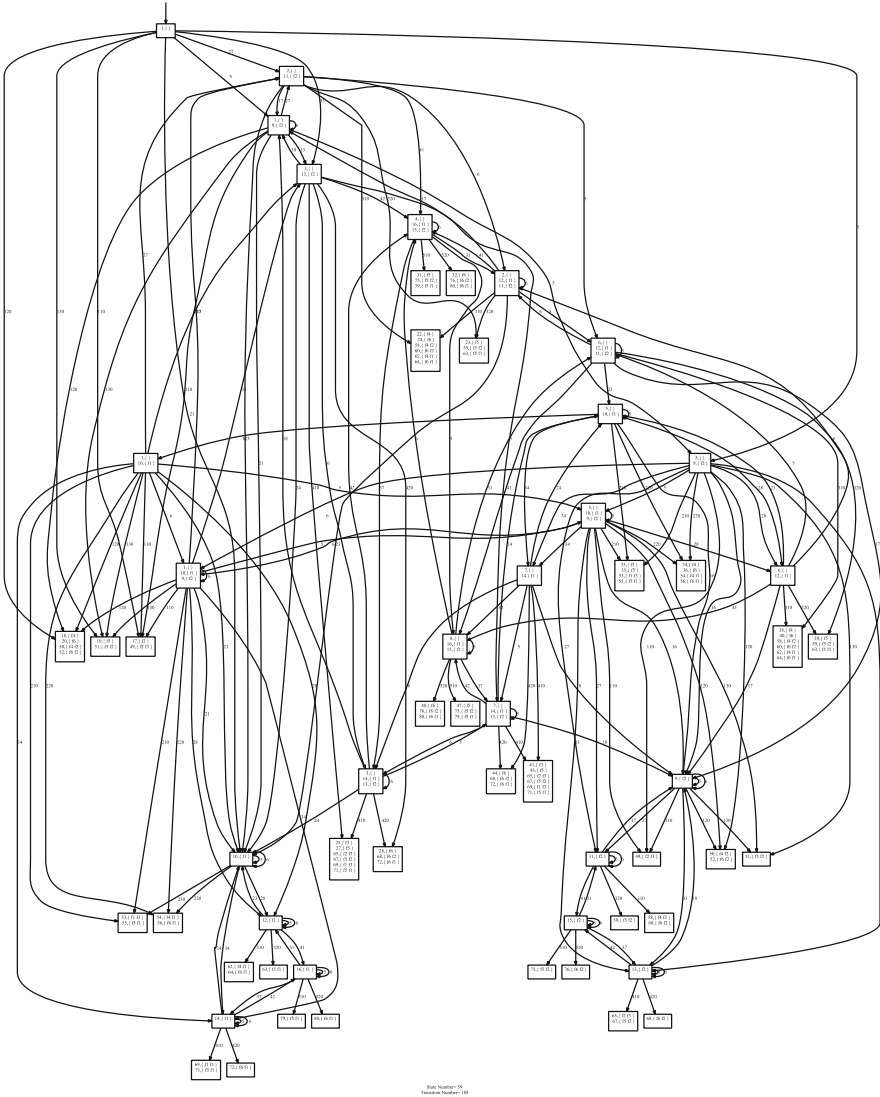


Fig. 10.8 Diagnoser without silent closure obtained using DIADES tool

overflow situation and a non-structural fault in a sensor. These events are reported in Table 10.6. Notice, for instance, that when the system is in mode q_3 , $\Phi_{67}(k) = \mathbf{0}$ during the time interval [3600 s, 3900 s] whereas the remaining consistency relations differ from zero.

Next, a non-structural fault occurs at 7800 s, that is detected by the diagnoser. The set of consistency indicators of mode q_5 are used to isolate the fault. The observed

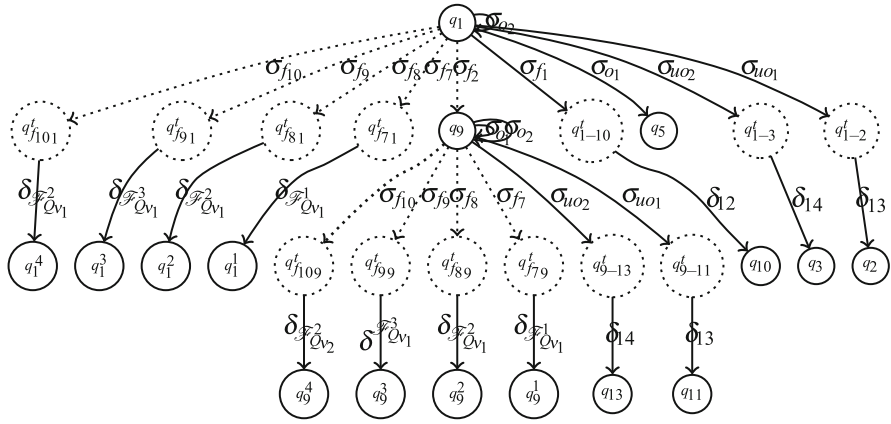


Fig. 10.9 Initial incremental behavior automaton B_{mit}

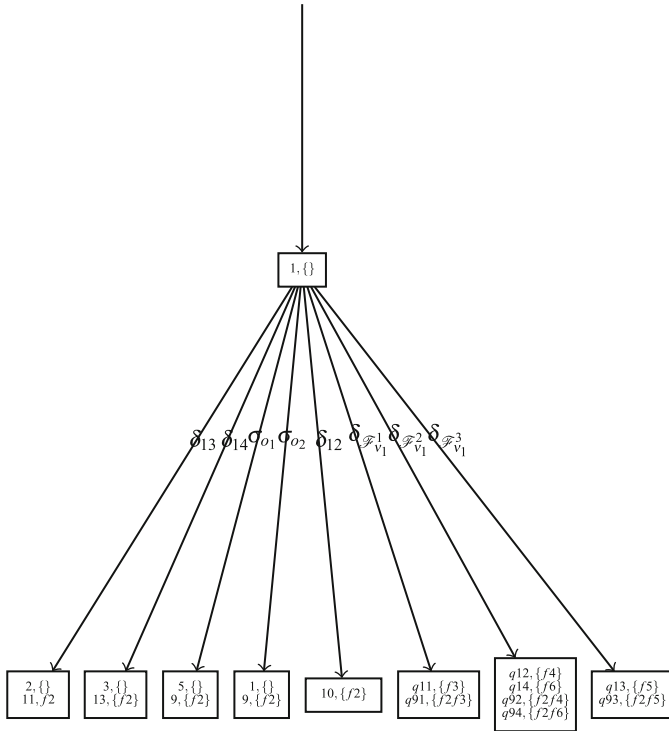


Fig. 10.10 Initial incremental diagnoser D_{mit}

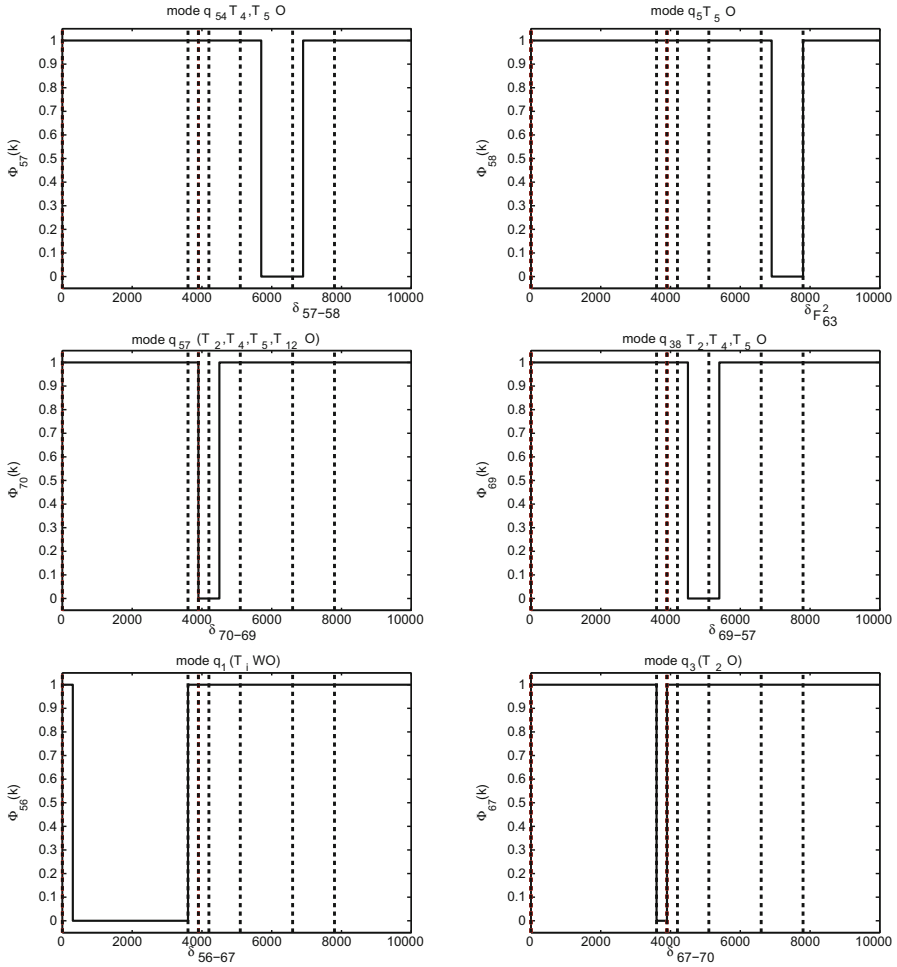


Fig. 10.11 Binary residuals

signature is $[0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]^T$ which, according to \mathbf{FS}_{58} , corresponds to a fault in sensor L_{41} (see Fig. 10.12). Finally, the hybrid diagnoser stops and reports the diagnosis. Indeed, a non-structural faults needs to be repaired before the diagnoser can resume.

The report given by the hybrid diagnoser is shown in Table 10.6. The first column represents mode changes in HA^k , the second one, the identified events. The third column corresponds to the diagnoser state information and total number of states generated, the fourth one shows the total number of residuals generated. The last two columns show the occurrence time and the detection time of the identified events.

Table 10.7 shows in detail how the incremental automata HA^k , B^k and D^k are built when an incoming event is observed and identified. The first column shows

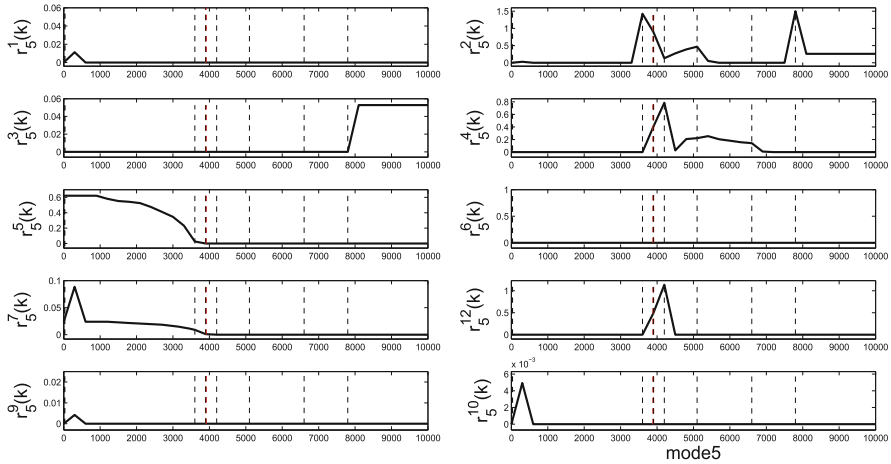


Fig. 10.12 Residuals of mode q_{23} when f_{L41} is detected

Table 10.6 Hybrid diagnoser report for the simulation scenario

Mode change	Reported event	Current diagnoser state	Occurrence time (s)	Detection time (s)
Initial mode q_1	—	$(q_1, \{\})$	—	—
$q_1 \rightarrow q_3$	δ_{56-67}	$q_3, \{q_{21}, \{f_{1b}\}$ $q_{36}, \{f_{2b}\}q_{50}, \{f_{3b}\}$	3600	3600
$q_3 \rightarrow q_{214}$	δ_{67-70}	$q_{214}, \{q_{238}, \{f_{1a}\}$ $q_{251}, \{f_{1b}\}q_{264}, \{f_{2b}\}$ $q_{274}, \{f_{3a}\}$	3900	3900
$q_{214} \rightarrow q_{140}$	σ_{70-69}	$q_{140}, \{q_{166}, \{f_{1a}\}$ $q_{179}, \{f_{1b}\}$	4200	4500
$q_{140} \rightarrow q_{211}$	σ_{69-57}	$q_{211}, \{q_{248}, \{f_{1b}\}$ $q_{261}, \{f_{2b}\}q_{271}, \{f_{3a}\}$	5100	5400
$q_{211} \rightarrow q_5$	σ_{57-58}	$q_5, \{q_{23}, \{f_{1b}\}$ $q_{38}, \{f_{2b}\}q_{52}, \{f_{3a}\}$	6600	6900
$q_5 \rightarrow q_{885}$ fault in $L_{41} \in \mathcal{F}_{ns}$	$\delta_{\mathcal{F}_{58}^2}$	$q_{885}, \{f_8\}q_{899}, \{f_{1bf8}\}$ $q_{913}, \{f_{2bf8}\}q_{927}, \{f_{3af8}\}$	7800	7800

the transitions that occur during the simulation scenario, which is described in Table 10.2. The second column shows how the number of generated residuals increases with every mode change. Notice that the average number of residuals is 130 per iteration. The remaining columns provide the average number of modes and states for each automaton. Regarding B^k , the fourth column highlights the number of transitions that the Sampath’s algorithm has to explore when building the diagnoser. Between 10 and 61 modes belonging to \mathcal{Q}_N and $\mathcal{Q}_{\mathcal{F}_s}$ are explored in HA^k per iteration. 140 modes belonging to $\mathcal{Q}_{\mathcal{F}_{ns}}$ are generated in HA^k per iteration. Additionally, between 81 and 140 diagnoser states are computed per iteration.

Table 10.7 Sewer network complexity for the considered scenario

Mode change	$ \Phi^k , \mathcal{Q}_v^k $	HA^k $ \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s} + \mathcal{Q}_{\mathcal{F}_{ns}} $	B^k $ \mathcal{Q} (\bar{T})$	D^k $ \mathcal{Q}_D $
Initial mode q_1	130, 13	61 +42	182 (199)	13
$q_1 \rightarrow q_3$	240, 24	137+70	437 (495)	32+13
$q_3 \rightarrow q_{214}$	340, 34	209+70	686 (791)	29+45 =74
$q_{214} \rightarrow q_{140}$	540, 54	347+70	1180 (1383)	74+28=102
$q_{140} \rightarrow q_{211}$	610, 61	386+42	1340 (1582)	102+28=130
$q_{211} \rightarrow q_5$	690, 69	431 +42	1506 (1781)	130+26=156
$q_5 \rightarrow q_{885}$ fault in $L_{41} \in \mathcal{F}_{ns}$	690, 69	0	0	156
Total	690, 69	431 + 266 = 697	1506 (1781)	156

Table 10.8 Comparison between incremental and non-incremental method for the simulation scenario

	Non-incremental method	Incremental method
Number of modes to be explored	32,768	697
Number of non-structural faulty modes to be generated	458,752	266
Number of diagnoser states to be computed	2^{32768}	156
Number of residuals to be computed	$10 * 32768 = 327,680$	690
Spatial computational complexity	Exponential ($O(2^{n_q})$) n_q : number of global behavior automaton states	Exponential ($O(2^{n_q^{nom}})$)/Linear ($O(n_q^{nom})$) n_q^{nom} : number of nominal modes

Table 10.8 provides a comparison of the results obtained with the proposed method and those obtained according to the non-incremental method of [6, 31], standing out the benefits of the proposed method. As can be seen, the process complexity increases with the number of operation modes. Hence, the non-incremental method could have a very high cost.

As can be seen in Table 10.8, the complexity of the incremental method is much lower than the complexity of the standard method. The number of explored and generated modes remains quite tractable.

10.6 Conclusions

This chapter has presented a hybrid system diagnosis approach based on the behavior automaton framework and the algorithms used to track the system mode.

The proposed diagnosis approach is able to detect and isolate two types of faults: structural and non-structural faults. A diagnoser executes the tasks of mode recognition and identification using mode discernibility provided by consistency indicators generated from a set of residuals for every mode. It has been shown that the proposed hybrid diagnosis approach can operate in a non-incremental and an incremental manner. In the non-incremental form, algorithms are executed taking into account global models. In the incremental form, only the useful parts of the diagnoser are built, developing the branches that are needed to explain the occurrence of incoming events. Thus, the resulting diagnoser adapts to the system operation mode and is less demanding in terms of memory storage than building the full diagnoser offline. The incremental method is illustrated by the application to a case study based on a representative part of the Barcelona sewer network and its complexity is compared to the non-incremental method.

The proposed incremental approach could be accommodated by computing off-line the part of the diagnoser corresponding to the modes with highest probability, in particular the nominal modes, and building the rest of the diagnoser as proposed whenever it is necessary. This would achieve even better space/time complexity.

The implementation used in the application presented in the paper is totally software since the sampling time was large enough to allow real-time operation. In case of a shorter sampling time, the use of a hardware or mixed hardware/software implementation would be necessary. These alternative implementation architectures will be part of future research work.

Acknowledgements This work has been partially funded by the Spanish Government (MINECO) through the project ECOCIS (ref. DPI2013-48243-C2-1-R), by MINECO and FEDER through the project HARCRICS (ref. DPI2014-58104-R).

References

1. Arogeti, A., Wang, D., & Low, C. B. (2010). Mode identification of hybrid systems in the presence of fault. *IEEE Transactions on Industrial Electronics*, 57(4), 1452–1467.
2. Bayouhd, M. (2009). *Active Diagnosis of Hybrid Systems Guided by Diagnosability Properties- Application to Autonomous Satellites*. PhD thesis, l'Université de Toulouse, Institut National Polytechnique, Toulouse, France.
3. Bayouhd, M., & Travé-Massuyès, L. (2014). Diagnosability analysis of hybrid systems cast in a discrete-event framework. *Discrete Event Dynamic Systems*, 24(3), 309–338.
4. Bayouhd, M., Travé-Massuyès, L., & Olive, X. (2007). State tracking in the hybrid space. In *18th International Workshop on Principles of Diagnosis (DX-07)*, May 2007 (pp. 221–228).
5. Bayouhd, M., Travé-Massuyès, L., & Olive, X. (2008). Coupling continuous and discrete event system techniques for hybrid system diagnosability analysis. In *18th European Conference on Artificial Intelligence (ECAI 2008)*, July 2008 (pp. 219–223).
6. Bayouhd, M., Travé-Massuyès, L., & Olive, X. (2008). Hybrid systems diagnosis by coupling continuous and discrete event techniques. In *17th IFAC World Congress* (Vol. 41, pp. 7265–7270).
7. Bayouhd, M., Travé-Massuyès, L., & Olive, X. (2009). On-line analytic redundancy relations instantiation guided by component discrete-dynamics for a class of non-linear hybrid systems.

- In *Proceedings of the Decision and Control Conference CDC/CCC 2009*, Shanghai (China) (pp. 6970–6975).
8. Benazera, E., & Travé-Massuyès, L. (2009). Set-theoretic estimation of hybrid system configurations. *IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics*, 39(5), 1277–1291.
 9. Blanke, M., Kinnaert, M., Lunze, J., & Staroswiecki, M. (2006). *Diagnosis and fault tolerant control* (2nd ed.). New York: Springer.
 10. Blom, H. A. P., & Bar-Shalom, Y. (1988) The interacting multiple model algorithm for systems with Markovian switching coefficients. *IEEE Transactions on Automatic Control*, 33, 780–783 (1988)
 11. Bregon, A., Alonso, C., Biswas, G., Pulido, B., & Moya, N. (2012). Fault diagnosis in hybrid systems using possible conflicts. In *8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes* (pp. 132–137).
 12. Cassandras, C., & Lafortune, S. (2008). *Introduction to discrete event systems*. New York: Springer.
 13. Chow, E., & Willsky, A. (1984). Analytical redundancy and the design of robust failure detection systems. *IEEE Transactions on Automatic Control*, 29(7), 603–614.
 14. Cocquempot, V., Staroswiecki, M., & El Mezyani, T. (2003). Switching time estimation and fault detection for hybrid systems using structured parity residuals. In *Proceedings of the 15th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes* (pp. 681–686).
 15. Daigle, M. (2008). *A Qualitative Event-Based Approach to Fault Diagnosis of Hybrid Systems*. PhD thesis, Faculty of the Graduate School of Vanderbilt University, Nashville, TN.
 16. de Freitas, N. (2002). Rao-Blackwellised particle filtering for fault diagnosis. In *Proceedings of the IEEE Aerospace Conference 2002* (Vol. 4, pp. 1767–1772).
 17. Ding, X., Kinnaert, M., Lunze, J., & Staroswiecki, M. (2008). *Model based fault diagnosis techniques*. Berlin: Springer.
 18. Georges, J.-P., Theilliol, D., Cocquempot, V., Ponsart, J.-C., & Aubrun, C. (2011). Fault tolerance in networked control systems under intermittent observations. *International Journal of Applied Mathematics and Computer Science*, 21(4), 639–648.
 19. Hofbaur, M., & Williams, B. (2004). Hybrid estimation of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 34(5), 2178–2191.
 20. Krysander, M., Åslund, J., & Nyberg, M. (2008). An efficient algorithm for finding minimal over-constrained sub-systems for model-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 38(1), 197–206.
 21. Lygeros, J., Henrik, K., & Zhang, J. (2003). Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48, 2–17.
 22. Meseguer, J., Puig, V., & Escobet, T. (2010). Fault diagnosis using a timed discrete-event approach based on interval observers: Application to sewer networks. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 40(5), 900–916.
 23. Meseguer, J., Puig, V., & Escobet, T. (2010). Observer gain effect in linear interval observer-based fault detection. *Journal of Process Control*, 20(8), 944–956.
 24. Narasimhan, S., & Biswas, G. (2007). Model-based diagnosis of hybrid systems. *IEEE Transactions on Systems, Man and Cybernetics*, 37(3), 348–361.
 25. Pencilé, Y. (2006–2015). *Diades: Diagnosis of discrete-event systems*. <http://homepages.laas.fr/yencilé/DiaDes/>
 26. Rienmuller, T., Hofbaur, M.W., Travé-Massuyès, L., & Bayouhd, M. (2013). Mode set focused hybrid estimation. *International Journal of Applied Mathematics and Computer Science*, 23(1), 13 pp.
 27. Sampath, M., Sengupta, R., & Lafortune, S. (1995). Diagnosability of discrete-event system. *IEEE Transactions on Automatic Control*, 40(9), 1555–1575.
 28. Travé-Massuyès, L., Bayouhd, M., & Olive, X. (2008). Hybrid systems diagnosis by coupling continuous and discrete event techniques. In *Proceedings of the 17th World Congress*, Seoul, Korea, July 2008 (pp. 7265–7270).

29. Travé-Massuyès, L., Bayouhd, M., & Olive, X. (2009). On-line analytic redundancy relations instantiation guided by component discrete-dynamics for a class of non-linear hybrid systems. In *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, December 2009, Shanghai, P.R. China (pp. 6970–6975).
30. Vento, J., Puig, V., & Sarrate, R. (2010). Fault detection and isolation of hybrid system using diagnosers that combine discrete and continuous dynamics. In *2010 Conference on Control and Fault-Tolerant Systems (SysTol)*, October 2010 (pp. 149–154).
31. Vento, J., Puig, V., & Sarrate, R. (2011). A methodology for building a fault diagnoser for hybrid systems. In *9th European Workshop on Advance Control and Diagnosis*, Budapest, Hungary, November 2011.
32. Vento, J., Puig, V., & Sarrate, R. (2012). Parity space hybrid system diagnosis under model uncertainty. In *2012 20th Mediterranean Conference on Control Automation (MED)*, July 2012 (pp. 685–690).
33. Vento, J., Puig, V., Sarrate, R., & Travé-Massuyès, L. (2012). Fault detection and isolation of hybrid systems using diagnosers that reason on components. *IFAC Proceedings Volumes*, 45(20), 1250–1255. 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes.
34. Vento, J., Travé-Massuyès, L., Puig, V., & Sarrate, R. (2015). An incremental hybrid system diagnoser automaton enhanced by discernibility properties. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(5), 788–804.
35. Vento, J., Travé-Massuyès, L., Sarrate, R., & Puig, V. (2013). Hybrid automaton incremental construction for online diagnosis. In *International Workshop on Principles of Diagnosis*, October 2013 (pp. 186–191).

Index

A

- Absolute residual, 103, 104
- ADAPT, *see* Advanced Diagnostics and Prognostics Testbed (ADAPT)
- Adaptive resonance theory, 25
- Adaptive state estimation, 154
- ADAPT-Lite
 - component fault modes, 198
 - diagnosability, 202–203
 - discrete and parametric faults, 205
 - fault signatures, 202, 203
 - measured and predicted values, 204
 - structural model decomposition, 201–202
 - system modeling, 198
 - system schematics, 198
 - true fault, 204–205
- Advanced Diagnostics and Prognostics Testbed (ADAPT), 11, 181, 197, 198
- AMESim™ model, 170, 172
- Analytical redundancy relations (ARRs), 46, 247
 - DBG technique, 57–59
 - FDI community, 180
 - two-tank benchmark, 61–68
- Anti-lock braking system (ABS), 123
- Artificial Intelligence Diagnosis (DX) communities, 180
- Artificial neural networks (NNs), 18
- Automated fault diagnosis, 179
- Automotive engines, *see* Spark ignition gasoline engine system
- Autonomous discrete events, 52, 65

B

- Barcelona sewer network, 262–263
- Behavior automaton (BA)
 - ARRs, 247
 - Barcelona sewer network, 262–263
 - binary residuals, 270, 273
 - COMP, 249–251
 - consistency indicators, 252–253
 - diagnoser without silent closure, 270, 271
 - diagnosis architecture, 245–247
 - discernibility
 - definition, 246
 - event label, 257
 - mode discernibility analysis, 253–255
 - transient mode, 256
 - hybrid diagnoser, 258
 - report, 273, 274
 - incremental hybrid system diagnosis, 275
 - belief mode, 260
 - consistency indicators, 260
 - current mode, 259
 - incremental behavior automaton (B^k), 249, 262
 - incremental hybrid diagnoser (D^k), 262
 - incremental initial mode (HA_{mit}), 261
 - initial incremental behavior automaton (B_{mit}), 270, 272
 - initial incremental diagnoser, 270, 272 (D_{mir})
 - non-structural faulty modes, 261
 - observable events, 259
 - parallel composition, 261

- Behavior automaton (BA) (*cont.*)
- saturation and dead zone representation, 261, 262
 - state space matrices, 261
 - mode tracking logic, 258–259
 - non-incremental form, 244, 275
 - non-structural faults, 247–248, 252
 - parallel composition, 251
 - parameter uncertainty, 248
 - parity-space approach, 247
 - residuals of mode, 273, 274
 - sensitivity concept, 248
 - sewer network
 - continuous dynamical model, 266
 - DIADES tool, 267
 - fault distribution matrices, 266–267
 - input–output models, 269
 - network complexity, 273, 275
 - non-discernible mode sets, 270
 - non-structural faults, 266
 - output function, 266
 - state space matrices, 268
 - structural faults, 266
 - type of events, 266
 - signature-events, 248
 - state space model, 252
 - transition function, 252
 - virtual tanks and control gates, 263–265
- Bond graph (BG) modelling
- BG-LFT model, 47
 - causal and structural properties, 50
 - causality assignment, 50, 51
 - definition, 49–50, 127
 - HBG technique, 52–53
 - hydraulic tank and valve, 50, 52
 - modelling approach, 124
 - power variables, 50
- C**
- Causal assignments, 186
- Chemical process
- detection, 116–117
 - diagnosis, 118–119
 - flowsheet, 114, 115
 - production, 115
 - reference model, 115–116
- Coherence vector (CV), 59–60, 74
- Component automata, 264, 267
- Component-based modelling, 182–185
- Conflict-driven diagnosis, *see* Possible conflicts (PCs)
- Consistency indicators, 252–253
- Consistent causal assignments, 186–188
- Continuous Petri nets (CPN), 211
- Controlled discrete events, 52, 65
- Controlled junctions (CJs), 3
- Covariance estimation, 164–165, 169
- Cylinder valve faults, 162
- D**
- Degenerated junctions, 128–129
- DES, *see* Discrete-event system (DES)
- Detectability index (D_b), 60
- DFNN algorithms, 40
- DIADES tool, 267, 271
- Diagnostic Bond Graph (DBG) technique
- ARR/GARR and adaptive threshold, 57–59
 - coherence vector, 59–60
 - DHBG-LFT form model, 54
 - fault signatures, 59–60
 - measurement uncertainty, 56–57
 - optimal threshold and mode fault detection
 - definition, 61
 - discrete mode fault, 60, 61
 - GFSSM, 62
 - Hilbert transform, 62–63
 - initial relative deviation, 63
 - MCSSM, 62, 63
 - parameter estimation, 64
 - residual evaluation, 62
 - SBG approach, 64
 - parameter uncertainty, 54–56
- Diagnostic HBG (DHBG), 54
- Discernibility
- definition, 246
 - mode discernibility analysis, 253–255
- Discrete-event system (DES), 12–13, 79
- advanced diagnosis, 212
 - changes of modes, 210
 - continuous state, 210
 - degradation state, 210
 - discrete state, 210
 - HPPN (*see* Hybrid particle petri nets (HPPN))
- Discrete faults, 4, 7–8
- ADAPT-Lite, 205
 - fault modeling, 185
 - four-tank hybrid system, 145–148
 - HBG-PC, 124, 125
 - SHBG-PC, 137–138
- Discretely controlled continuous systems (DCCSs), 1, 2, 17
- Discrete mode fault, 60, 61
- Discrete nonlinear observers, 154
- Doubly fed induction generator (DFIG), 17

E

- EKF, *see* Extended Kalman filter (EKF)
- Electrical power systems, *see* ADAPT-Lite
- Evolving type-2 random vector functional link network (ϵ T2RVFLN), 7, 19, 20
 - classification, 39–40
 - cognitive architecture
 - Chebyshev polynomial, 27
 - classification decision, 28
 - design coefficient, 27–28
 - fuzzy rule, 25–26
 - interval firing strength, 27
 - Mahalanobis distance, 26
 - multivariate Gaussian function, 26
 - TSK model, 26–27
 - complexity, 41
 - DFNN and FAOSPFNN, 40
 - issues, 37
 - meta-cognitive learning (*see* Meta-cognitive learning policy)
- Exhaust manifold leak, 162
- Extended Kalman filter (EKF), 10, 11, 103–104, 154, 164, 170, 175, 212

F

- FAOSPFNN algorithms, 40
- Fast Fourier transform (FFT), 24
- Fault detection
 - SHBG-PCs approach
 - assumptions, 137–138
 - complexity, 142–143
 - diagnosis framework, 140–142
 - fault signature matrices, 138–140
 - four-tank hybrid system, 143–148
 - spark ignition gasoline engine system
 - diagnosis results, 172–175
 - estimation results, 170–172
 - isolation, 169–170
 - process, 168–169
 - thresholds, 176
- Fault detection and isolation (FDI)
 - adaptive thresholds, 46
 - ARR, 46
 - BG modelling
 - BG-LFT model, 47
 - causal and structural properties, 50
 - causality assignment, 50, 51
 - definition, 49–50
 - HBG technique, 52–53
 - hydraulic tank and valve, 50, 52
 - power variables, 50

- DBG technique (*see* Diagnostic Bond Graph (DBG) technique)
- model-based quantitative, 45, 46
- SSF, 47
- two-tank benchmark
 - ARRs/GARRs, 66–68
 - autonomous discrete event, 65
 - coherence vector, 74
 - controlled discrete event, 65
 - GFSSM and MCSSM, 70, 71, 74
 - HBG model, 66
 - optimum adaptive threshold, 68–70
 - parametric fault and discrete mode fault, 70
 - PI-controller, 65
 - response of, 71–73
 - schematic diagram, 64, 65
 - SSF, 74–75

Fault incidence matrix (FIM), 172, 173, 175

Fault isolation

- computational complexity, 93
- decision making, 113–114
- distance, 111–113
- HPPN, 212
- principle, 108–110
- SMPL automata, 91–92
- Fault signatures
 - ADAPT-Lite, 202, 203
 - qualitative fault isolation approach, 194–195
- FDI, *see* Fault detection and isolation (FDI)
- Feed-forward NN, 24
- FIM, *see* Fault incidence matrix (FIM)
- Four-tank hybrid system, 143–148
- Fuzzily weighted generalized recursive least squares (FWGRLS), 35–36
- Fuzzy logic systems, 18

G

- Generalized Likelihood Ratio Test (GLRT)
 - based fault detection
 - FIM, 173
 - IM leak fault, 175
 - residual prediction, 166, 168
- Global-ARRs (GARRs), 46
 - DBG technique, 57–59
 - two-tank benchmark, 61–68
- Global Fault Sensitivity Signature Matrix (GFSSM), 59
- Grid side converters, 17

H

- Hamming distance, 110–111
- HBGs, *see* Hybrid bond graphs (HBGs)
- Heating, ventilation, and air conditioning (HVAC) systems, 123
- Hidden node growing mechanism, 31–32
- Hidden node pruning mechanism, 33–34
- HNS modeling, *see* Hybrid nonlinear system (HNS) modeling
- How-to-learn method, 19–20
 - hidden node growing mechanism, 31–32
 - hidden node pruning mechanism, 33–34
 - online feature selection mechanism, 34
 - RVFLN, 35–36
- HPPN, *see* Hybrid particle petri nets (HPPN)
- HQFSM, *see* Hybrid qualitative fault signature matrix (HQFSM)
- Hybrid automation, 3
- Hybrid bond graphs (HBGs), 10, 48, 52–53, 212
 - advantage, 124
 - hybrid systems modelling, 125
 - observation delays, 181
 - PCs
 - assumptions, 126
 - bond graph model, 127, 128
 - characterization, 130–132
 - degenerated junctions, 128–129
 - RBG, 129–130
 - VCA, 131
- Hybrid diagnosis engine (HyDE), 181, 213
- Hybrid dynamic systems (HDS), 79
 - chemical process
 - detection, 116–117
 - diagnosis, 118–119
 - flowsheet, 114, 115
 - production, 115
 - reference model, 115–116
 - classes, 1
 - controlled junctions, 3
 - DCCS, 1, 2, 17
 - definition, 1
 - fault diagnosis
 - challenge of, 6
 - definition, 3
 - discrete faults, 4
 - external methods, 6
 - internal methods, 5
 - offline diagnosis, 5
 - parametric faults, 4–5
 - fault isolation
 - decision making, 113–114
 - distance, 111–113
 - principle, 108–110
 - FDI (*see* Fault detection and isolation (FDI))
 - hybrid automation, 3
 - hybrid bond graph, 3
 - hybrid function, 2–3
 - hybrid Petri nets, 2
 - incidence matrix, 106–109
 - MCCS, 17–18
 - residual estimation, 105–106
 - residual generation, 103–104
 - spark ignition gasoline engine (*see* Spark ignition gasoline engine system)
- Hybrid fault signature matrix (HFSM), 139
- Hybrid function, 2–3
- Hybrid hydraulic system, 51–53
- Hybrid nonlinear system (HNS) modeling
 - mode changes, 153
 - spark ignition gasoline engine system, 155–158
 - fault modelling, 162
 - model equations, 159–161
 - model parameters and tuning, 161–162
 - state estimation, 154
- Hybrid particle Petri net (HPPN), 12
 - CPN, 211
 - definition, 216
 - degradation places P^D model, 217–218
 - diagnoser generation
 - behavioral level, 225, 226
 - degradation conditions removal, 226–227
 - degradation level, 225, 226
 - discrete, continuous and degradation state spaces, 225
 - mergeable transitions, 227
 - merging of two transitions, 227–228
 - symbolic conditions, 226
 - diagnoser process, 229–231
 - fault isolation, 212
 - fireable transition, 222
 - fluid stochastic Petri nets, 211
 - HPN, 211
 - hypothesis, 218–219
- K11 rover, health monitoring and diagnosis
 - continuous dynamics, 234
 - degraded mode, 213
 - description, 231–233
 - failure mode, 213
 - fault f_2 occurrence, 234
 - health modes, 213
 - HPPN-based diagnoser $HPPN_{\Delta}$, 214
 - HPPN-based prognoser $HPPN_{\Pi}$, 214
 - hybrid system model $HPPN_{\Phi}$, 214
 - HyDE, 213

- mobile robot description, 214–215, 219–220
 - nominal mode, 213
 - overview, 214
 - QED, 213
 - simulation results, 235–238
 - temperature model, 234
 - marking evolution rules, 220–222
 - MPPN, 211
 - numerical places P^N model, 217
 - particle cluster, 219
 - PPN, 211
 - symbolic places P^S model, 216–217
 - transition firing, 222
 - uncertainty, 223–225
 - Hybrid PCs, 126
 - Hybrid Petri nets (HPN), 2, 211
 - Hybrid qualitative fault signature matrix (HQFSM), 139–140
 - Hybrid system (HS), 79
 - definition, 81, 209–210
 - descriptions, 82–83
 - equivalence, 96
 - finite SMPL system, 96
 - max-plus algebraic methods, 81
 - polyhedral partition, 96
 - Hybrid systems modeling
 - causality, 186–188
 - compositional modeling, 182–185
 - diagnosis
 - architecture, 137
 - problem, 191–193
 - qualitative fault isolation, 195–197
 - fault modeling, 185
 - HBGs, 125
 - structural model decomposition, 188–191
 - Hypothesized State*, 141, 142
- I**
- Incidence matrix, 106–109
 - Incremental hybrid system diagnosis, 275
 - belief mode, 260
 - consistency indicators, 260
 - current mode, 259
 - incremental behavior automaton (B^k), 249, 262
 - incremental hybrid diagnoser (D^k), 262
 - incremental initial mode (HA_{init}), 261
 - initial incremental behavior automaton (B_{init}), 270, 272
 - initial incremental diagnoser (D_{init}), 270, 272
 - non-structural faulty modes, 261
 - observable events, 259
 - parallel composition, 261
 - saturation and dead zone representation, 261, 262
 - state space matrices, 261
 - Induction motors
 - artificial intelligence techniques, 18
 - “big data,” 19
 - computer specifications, 38–39
 - data collection, 37, 38
 - DCCSs, 17
 - direct-current motors, 15–16
 - eT2RVFLN (*see* Evolving type 2 RVFL network (eT2RVFLN))
 - fault detection and diagnosis features
 - events/type of faults, 22, 23
 - motor components, 21, 22
 - physical and electrical measurements, 21, 22
 - single and multiple sources, 24–25
 - types, 21
 - features and class label, 18
 - low-cost and efficient method, 16
 - MCCS, 17, 18
 - MCSA, 16
 - motor operation, 16
 - RNNs, 19
 - statistical techniques, 18
 - training and test, 39
 - Input posterior probability, 20–21
 - Intake manifold leak fault, 159, 167, 174
 - Interactive multiple-model (IMM), 154, 212
 - Interval Extension Functions (IEF) technique, 47
 - Isolatability index (I_b), 60
- K**
- Kalman filter, 9, 154
 - Kalman filter algorithm, 25
 - K11 rover
 - description, 231–233
 - health monitoring and diagnosis
 - continuous dynamics, 234
 - degraded mode, 213
 - failure mode, 213
 - fault f_2 occurrence, 234
 - health modes, 213
 - HPPN-based diagnoser HPPN $_{\Delta}$, 214
 - HPPN-based prognoser HPPN $_{\Pi}$, 214
 - hybrid system model HPPN $_{\Phi}$, 214
 - HyDE, 213
 - mobile robot description, 214–215, 219–220

- K11 rover (*cont.*)
 nominal mode, 213
 overview, 214
 QED, 213
 simulation results, 235–238
 temperature model, 234
- L**
 Linear Fractional Transformation (LFT), 46
- M**
 Mahalanobis distance, 26
 Manhattan distance, 111–113
 Markov switching systems, 97
 Max-min-plus-scaling (MMPS) systems, 82
 Max-plus algebraic methods
 computational complexity
 approximation algorithm, 93
 fault detection, 92
 fault isolation, 93
 DES, 79
 diagnosis scenarios
 T_3 leak, 94
 V_2 blockage, 95
 V_3 blockage, 94–95
 faults model, 88–89
 generality, 86–87
 hybrid system
 definition, 81
 descriptions, 82–83
 equivalence, 96
 finite SMPL system, 96
 polyhedral partition, 96
 Markov switching systems, 97
 MPL systems, 84–85
 nominal model, 87–88
 objectivity, 81
 Petri nets models, 83
 SMPL, 80, 85–86
 SMPL automata
 fault isolation, 91–92
 observers, 90
 stochastic SMPL system, 80
 system architecture, 81–82
 TEG control, 80
 time parameters, 89–90
 Max-plus-linear (MPL) systems, 84–85
 Max-plus state space model, 84–85
 Mean Value Models (MVMs), 153
- Meta-cognitive learning policy
 how-to-learn method
 hidden node growing mechanism, 31–32
 hidden node pruning mechanism, 33–34
 online feature selection mechanism, 34
 RVFLN, 35–36
 overview, 28, 29
 pseudocode, 28, 30
 what-to-learn method, 28–31
 when-to-learn method, 36
- Mode Change Sensitivity Signature Matrix (MCSSM), 59, 70, 71, 74
- Mode discernibility analysis, 253–255
- Model-based fault diagnosis
 discrete and parametric faults, 140
 sensors, 128
 stages, 153
- Model-based online diagnosis, 243
- Model predictive control (MPC), 83
- Modified particle Petri nets (MPPN), 211
- Moore-Penrose generalized inverse matrix, 35
- Motor current signature analysis (MCSA), 16, 24
- Multicellular converter continuous systems (MCCS), 17, 18
- Multimode system, 244
- Multivariate Gaussian function, 26
- N**
 Neuro fuzzy system, 24
 Neyman–Pearson (NP) test, 168
 Nominal EKF, 166, 167
 Nonlinear observers, 154
- O**
 Online feature selection mechanism, 34
 Optimum adaptive threshold, 61, 68–70
 Output posterior probability, 20
- P**
 Parametric faults, 4, 7–8, 148, 180
 See also Discrete faults
 Parity-space approach, 247
 Particle filters (PF), 154, 155, 170
 Particle Petri nets (PPN), 211
 Petri nets models, 83
 Piecewise affine (PWA) systems, 8, 82, 97
 Polyhedral partition, 96

- Possible conflicts (PCs), 124
 - BG models, 126–130
 - HBG models, 130–132
- Power spectral density (PSD), 24, 37
- Predictive filter method, 24
- PrODHyS, 9, 103, 115
- Prognostics and health management (PHM), 209
- Proportional-Integral (PI) controller, 64, 65

- Q**
- Qualitative event-based diagnosis (QED), 213
- Qualitative fault isolation approach
 - electrical power distribution system (*see* ADAPT-Lite)
 - fault signatures, 194–195
 - hybrid systems diagnosis, 195–197
 - mode changes, 205
 - observation delays, 181
 - PCs, 149
 - scalability, 197

- R**
- Randomized NNs (RNNs), 19
- Random vector functional link (RVFL), 19, 20, 35–36
- Rao-Blackwell Particle Filter (RBPf), 154, 170
- Receiver operating characteristics (ROC)
 - graph, 176
- Reduced qualitative fault signature matrix (RQ-FSM), 139
- Redundant structural sub bond graph (RBG), 129–130
- Relative mutual information (RMI), 33
- Residual-based approach, 5
- Residual estimation, 105–106
- Residual generation, 103–104
- Rotor side converters, 17
- Rule extraction techniques, 24

- S**
- Sensitivity BG (SBG) approach, 64
- Sensor bias faults, 163, 172
- Sequential Causal Assignment Process (SCAP)
 - algorithm, 124, 128, 146, 148
- Set-membership approach, 5
- Set of Suspected Faults (SSF), 47, 74–75
- Shapiro-Wilk W test, 113
- SHBG-PCs approach, *see* Structural hybrid Bond-Graph-possible conflicts (SHBG-PCs) approach
- Signature-events, 244
- Signature generation, 105–106
- Simulation Abnormal Event Management (SimAEM), 101–102
- Sliding mode estimators, 154–155
- Spark ignition gasoline engine system
 - block diagram, 156
 - fault detection
 - diagnosis results, 172–175
 - estimation results, 170–172
 - isolation, 169–170
 - process, 168–169
 - special features, 155
 - thresholds, 176
 - HNS modeling, 155–158
 - fault modelling, 162, 163
 - model equations, 159–161
 - model parameters and tuning, 161–162
 - residual prediction stage, 154–156, 167
 - state estimation, 162–164
 - adaptive Q/R, 165–166
 - EKF algorithm, 164, 175
 - variable and subscript notations, 157
- State space modelling, 155, 161
- Stochastic scaling algorithm (SSA), 229
- Structural hybrid Bond-Graph-possible conflicts (SHBG-PCs) approach
 - algorithms, 134–136
 - definition, 131–132
 - electric circuit configurations, 132–133
 - faults
 - assumptions, 137–138
 - complexity, 142–143
 - diagnosis framework, 140–142
 - fault signature matrices, 138–140
 - four-tank hybrid system, 143–148
 - performance, 136–137
- Structural model decomposition
 - ADAPT-Lite, 201–202
 - hybrid systems modeling, 188–191
 - qualitative fault isolation, 181
 - role, 205
- Structural sub bond graph (st-sBG), 129
- Switching junction, 124–126, 148, 149
- Switching max-plus linear (SMPL) systems, 80, 85–86
- Switching probability matrix, 86
- System health management (SHM), 209

T

Takagi Sugeno Kang (TSK) model, 26–27
Temporal Causal Graph (TCG), 124, 139
TentativePath, 134, 136
TentativeSBGPC, 134, 136
Three-tank system, 87, 89
Timed Event Graph (TEG), 79
Torricelli's Law, 88
TRANSCEND approach, 139
Type 2 relative mutual information (T2RMI),
33

U

Unscented Kalman filter (UKF), 154, 165, 170

V

Valid causal assignment (VCA), 124, 126,
129–132, 136, 137, 140, 142, 143,
148

W

What-to-learn phase, 19, 20, 28–31
When-to-learn phase, 19, 20, 36
Within-cycle models (WCMs),
153–156

Z

Z-test, 140