Universitat Politècnica de Catalunya

Programa de Doctorat:

Automàtica, Robòtica i Visió

Tesis Doctoral

# Methodologies for Hybrid Systems Diagnosis based on the Hybrid Automaton Framework

**Jorge Isaac Vento Maldonado**

Directors: Vicenç Puig i Ramon Sarrate

Març 2014

# ABSTRACT

Hybrid systems play an important role in the modeling of complex systems since they take into account the interaction between both continuous dynamics and discrete events. Complex systems are subject to changes in the dynamics due to several factors such as nonlinearities, changes in the parameters, disturbances, faults, discrete events and controller actions among others. These facts lead to the need to develop a diagnostic system for hybrid systems improving the diagnostic precision. Hybrid systems allow to combine the classic fault detection and isolation approaches and a diagnoser based on discrete event models. Hence, a design methodology and implementation architecture for diagnosers in the framework of hybrid systems is proposed.

The design methodology is based on the hybrid automaton model that represents the system behavior by means of the interaction of continuous dynamics and discrete events. The architecture is composed of modules which carry out mode recognition and diagnostic tasks interacting each other, since the diagnosis module adapts accordingly to the current hybrid system mode. The mode recognition task involves detecting and identifying a mode change by determining the set of residuals that are consistent with the current hybrid system mode. On the other hand, the diagnostic task involves detecting and isolating two type of faults: structural and non-structural faults. In the first case, structural faults are represented by a dynamic model as in the case of nominal modes. Hence they are identified by consistency checking through the set of residuals. In the second case, non-structural faults do not change the structure of the model, therefore, they are identified by a proper residual pattern.

Discernibility is the main property used in hybrid systems diagnosis. Through the concept of discernibility it is possible to predict whether modes changes (faulty or nominal) in the hybrid model can be detected and isolated properly. This concept can be applied in practice, evaluating a set of mathematical properties derived from residual expressions, which can be obtained from input-output models or parity space equations. General properties are derived to evaluate the discernibility between modes in the hybrid automaton model.

The diagnoser is built through propagation algorithms developed for discrete models represented by automata. The automaton employed to build the diagnoser for a hybrid system is named behaviour automaton. It gathers all information provided by discernibility properties between modes and observable events in the system, increasing the system diagnosability.

Diagnosis for hybrid systems can be divided in two stages: offline and online. Moreover, it can

be carried out twofold: in a non-incremental and an incremental form. In the non-incremental form, algorithms are executed taking into account global models, unlike incremental form that leads to building the useful parts of the diagnoser, only developing the branches that are needed to explain the occurrence of incoming events. The resulting diagnoser adapts to the system operational life and it is much less demanding in terms of memory storage than 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 by means of a tool implemented in Matlab.

**Keywords:** Fault detection and isolation, mode identification , diagnoser, hybrid systems, sewer networks.

# RESUMEN

La investigación realizada en esta tesis se basa en el diseño de sistemas de diagnóstico de fallos aplicado a sistemas complejos. La mayoría de los sistemas están controlados y supervisados de manera automática mediante el diseño de una interfaz que permite monitorear el sistema a través de la medición de sus variables de estado. Se han propuestos diversas metodologías para diseñar estos sistemas de diagnóstico que varían de acuerdo con las características propias del modelado.

La precisión del sistema de diagnóstico de fallos depende del modelado empleado. En el presente trabajo se propone usar modelos híbridos para representar el comportamiento de los sistemas complejos. Los modelos híbridos permiten combinar comportamientos del sistema gobernados por tiempo y eventos en un solo modelo, permitiendo de esta manera combinar las técnicas ya existentes de diagnóstico de fallos para sistemas continuos y a eventos discretos desarrolladas por separado. El hecho de poder combinar las ventajas que ofrecen dichas técnicas por separado permite obtener un sistema de diagnóstico más preciso.

Los modelos híbridos permiten representar el sistema mediante un conjunto de modos de operación del sistema, que pueden incluir comportamientos nominales y comportamientos de fallo. Los fallos pueden ser de diferente naturaleza: fallos no estructurales y fallos estructurales. Los fallos no estructurales pueden representar fallos en los sensores del sistema y en los actuadores. Los fallos estructurales permiten representar fallos en componentes del sistema como por ejemplo fallos en compuertas que se quedan atascadas, interruptores que no conmutan cuando deberían, entre otros.

La evolución de un sistema híbrido queda representada por el conjunto de eventos que pueden ocurrir en el sistema y el cambio en los modelos dinámicos continuos que varían con el punto de operación del sistema. El mayor problema que presentan los modelos híbridos para sistemas complejos es el número de modos que se deben considerar para tratar de representar los posibles comportamientos reales del sistema lo que en muchos casos hace difícil su implementación. Debido a esta problemática que afecta los sistemas complejos se propone diseñar una metodología para diagnosis en sistemas híbridos que haga posible su implementación de manera automatizada, evitando costo de computación y optimizando el espacio en memoria que pueden generar los modos de operación.

La metodología para la diagnosis de sistemas híbridos se basa en el autómata híbrido. El autómata híbrido representa el sistema por un conjunto de modos (nominal y fallo), donde la dinámica continua en cada modo se representa por un modelo discreto en espacio estado y su equivalente en función de

transferencia. La dinámica gobernada por eventos se representa por un conjunto de eventos discretos que pueden ser de dos tipos: observables y no observables. Los eventos observables comprenden acciones que el operador ejecuta sobre la planta o cuando las variables de estado superan un umbral y que pueden ser medidos. En el caso que no pueden ser medidos se consideran eventos no observables. Para el caso de los fallos, todos son considerados eventos no observables. El sistema de diagnóstico debe ser capaz de hacer un seguimiento de los modos de operación del sistema. Para saber si será posible detectar los cambios de modos en línea se estudia la propiedad de la discriminabilidad. Teóricamente se define la discriminabilidad como la capacidad de poder distinguir mediante el conjunto de medidas el comportamiento dinámico de los modos de operación cuando se evalúan los residuos en línea.

La diagnosis de sistemas híbridos se divide en dos etapas: diseño del diagnosticador y diagnosis en línea. En la primea fase se debe construir un diagnosticador que permita hacer un seguimiento a los modos de operación del sistema. Además, la ejecución de los algoritmos para la diagnosis se puede llevar a cabo de dos maneras: de forma no incremental e incremental. En el caso de la metodología no incremental los algoritmos se ejecutan sobre el modelo global del autómata híbrido obteniéndose de este modo un diagnosticador global. A diferencia del modelo incremental donde sólo se construyen las partes del modelo que son necesarias en la diagnosis en línea optimizando espacio y el coste de computación que generan los modos de operación a medida que ocurren los eventos en el sistema. La metodología se aplica a las redes de alcantarillado de la ciudad de Barcelona.

**Palabras clave:** Detección y aislamiento de fallos, identificación del modo de operación, diagnosticadores, sistemas híbridos, redes de alcantarillado.

# RESUM

La investigació realitzada en aquesta tesi es basa en el disseny de sistemes de diagnòstic de fallades aplicat a sistemes complexos. La majoria dels sistemes estan controlats i supervisats de manera automàtica mitjançant el disseny d'una interfície que permeti la seva monitorització a través de les mesures de les variables d'estats. S'han proposat diverses metodologies per dissenyar aquests sistemes de diagnòstic que varien d'acord amb les característiques pròpies del modelat.

La precisió del sistema de diagnòstic de fallades depèn del model utilitzat. Es proposa fer servir models híbrids per representar el comportament de sistemes complexos. Els models híbrids permeten combinar comportaments del sistema governats per temps i per esdeveniments discrets en un únic model. D'aquesta manera es poden combinar les tècniques existents de diagnòstic de fallades per sistemes continus i amb les d'esdeveniments discrets, que fins al moment s'han desenvolupat per separat. El fet de poder combinar aquestes tècniques per separat en un únic model permet obtenir un model de diagnòstic més precís.

Els models híbrids permeten representar el sistema a través d'un conjunt de modes d'operació, que inclouen comportaments nominals i de fallada. Les fallades poden ser de diferents tipus: fallades no estructurals i fallades estructurals. Les fallades no estructurals representen fallades als sensors i als actuadors del procés. Les fallades estructurals permeten representar fallades als components del sistema, com per exemple comportes que es queden embussades i interruptors que no commuten quan són accionats, entre altres.

L'evolució d'un sistema híbrid queda representada per el conjunt d'esdeveniments que poden acórrer al sistema i el canvi dels models continus, que varien amb el punt d'operació del sistema. El major problema que presenten els models híbrids és el numero de modes que s'han de considerar per representar els possibles comportaments reals del sistema i que a vegades dificulta massa la seva implementació. A causa d'aquesta problemàtica que afecta als sistemes complexos es proposa dissenyar una metodologia per diagnosis de sistemes híbrids que faci possible la seva implementació de manera automatitzada, evitant el cost de computació i optimitzant l'espai en memòria que poden generar els modes de operació.

La metodologia per la diagnosi de sistemes híbrids es basa en l'autòmat híbrid. L'autòmat híbrid representa el sistema per un conjunt de modes (nominal i de fallada), on la dinàmica continua de cada mode es representa per un model a temps discret en espai d'estat o el seu equivalent en funció de transferència. La dinàmica governada per esdeveniments discrets es representa per un conjunt d'esdeveniments que

poden ser de dos tipus: observables i no observables. Els esdeveniments observables comprenen accions que l'operador executa a la planta o quan les variables d'estat superen un llindar i que poden ser mesurats. En cas de no poder ser mesurats es consideren esdeveniments no observables. Pel cas de fallades, totes es consideren esdeveniments no observables. El sistema de diagnòstic hauria de ser capaç de fer un seguiment dels modes d'operació del sistema. Per saber si serà possible detectar els canvis de modes en línia s'estudia la propietat de la discernabilitat. Teòricament es defineix la discernabilitat com la capacitat de poder distingir a través del conjunt de mesures el comportament dinàmic dels modes d'operació quan s'avaluen els residus en línia.

La diagnosi de sistemes híbrids es divideix en dues etapes: disseny del diagnosticador i diagnosi en línia. En la primera fase s'ha de construir un diagnosticador que permeti fer un seguiment dels modes d'operació del sistema. A més, l'execució dels algorismes de diagnosi pot ser de dues maneres: no incremental i incremental. En el primer cas els algorismes s'executen considerant el model global de l'autòmat híbrid obtenint un model global del diagnosticador. En la versió incremental només es construeixen les parts del model que són necessàries, optimitzant l'espai en memòria i el cost de computació que generen els modes d'operació quan ocorren els esdeveniments al sistema. La metodologia s'aplica a les xarxes de aigua clavagueram de la ciutat de Barcelona.

**Paraules clau**: Detecció i aillament de fallades, identificació del mode d'operació, diagnosticadors, sistemes híbrids, xarxes de clavegueram.

# NOTATION

Throughout the thesis and as a general rule, scalars and vectors are denoted with lower case letters (e.g., $r$, $n$, $\mathbf{y}$, $\mathbf{x}$, ...), matrices are denoted with upper case letters (e.g., $\mathbf{A}$, $\mathbf{B}$, ...) and sets are denoted with upper case calligraphic letters (e.g., $\mathcal{X}$, $\mathcal{Y}$, ...). If not otherwise noted, all vectors are column vectors.

| | |
|---|---|
| $HA$ | hybrid automaton model |
| $q_i$ | mode $i$ |
| $\mathcal{Q}$ | set of modes |
| $\mathcal{Q}_{\mathcal{N}}$ | nominal modes |
| $\mathcal{Q}_{\mathcal{F}_s}$ | structural faulty modes |
| $\mathcal{Q}_{\mathcal{F}_{ns}}$ | non-structural faulty modes |
| $n_q$ | number of modes |
| $\mathbf{x}(k)$ | state variables |
| $\mathbf{y}(k)$ | output variables |
| $\mathbf{u}(k)$ | input variables |
| $\Sigma$ | set of events |
| $\mathcal{T}$ | transition function |
| $B$ | behavior automaton |
| $\Sigma^{Sig}$ | set of signature-events |
| $\mathcal{Q}_{disc}$ | set of non-discernible modes |
| $D$ | diagnoser |
| $\Delta$ | set of fault labels |
| $\mathcal{F}$ | set of faults |
| $p$ | delay operator |
| $\mathbf{\Lambda}_i(p^{-1})$ | sensitivity |
| $\mathbf{FS}_i$ | fault signature matrix |
| $\mathbf{r}_i(k)$ | residuals in mode $i$ |
| $\tau_i^l$ | threshold associated to a residual component |
| $\Phi_i(k)$ | consistency indicators |
| $S_i^j(\mathbf{x}, \mathbf{u})$ | saturation function |
| $D_i^j(\mathbf{x}, \mathbf{u})$ | dead zone function |
| $T_i$ | virtual tank |

| | |
|---|---|
| $L_{code}$ | Limnimeter label |
| $P_{code}$ | Pluviometer label |
| $G_{code}$ | Control gate label |
| $\varrho(k)$ | flow |
| $\Delta t$ | sampling time |
| $\mathcal{M}$ | set of components |

# CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

Most complex systems are controlled in real-time and they can be affected by faults in sensors, actuators and plant components. The dynamic behavior of a complex system can present different behaviors according to the system configuration and the plant components. To represent all these behaviors, hybrid system modeling can be used. Hybrid models combine discrete dynamics with continuous dynamics. Thus, this leads to design a diagnosis system adapted to detect and isolate faults in hybrid systems.

Complex systems are modeled using hybrid models that integrate continuous and discrete event dynamics. Then, by means of this model, the system mode is monitored such that fault diagnosis and control are properly performed online. 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 [Bregon et al., 2012]. The hybrid system behavior can be described by a hybrid automaton model [Hofbaur and Williams, 2004] or the hybrid bond graph formalism [Narasimhan and Biswas, 2007, Daigle, 2008].

A hybrid automaton models the real behavior of the system through a set of operation modes and a set of transitions between modes which trigger upon discrete events or events based on continuous state conditions. Continuous dynamics within each mode are described by a set of algebraic differential equations which constrain continuous state, input and output variables. Input and output variables are 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 fault events or other events that cause changes in the system state not directly recorded by sensors.

The need to develop a hybrid diagnoser for hybrid systems arises from the application of classic fault detection and isolation approaches based on models combined with a diagnoser based on discrete event models. For a hybrid system, to detect and isolate faults the diagnosis system should have the information about continuous dynamic and discrete events.

## 1.1 Motivation

The main motivation of this thesis is to develop a diagnosis system methodology for hybrid systems, which integrates the existing techniques so far developed separately for continuous and discrete-event systems. Besides, taking advantage of the previous methodologies general properties for diagnosability study will be developed in a unified way. Properties for diagnosability have been approached separately according to the fault influence in the system. The need to develop a methodology for hybrid systems that is applicable in practice and in an automatic way are additional motivations. This thesis focuses on the hybrid automaton model following the FDI community principles to develop a methodology to track the system mode and diagnose hybrid systems. As real case study to prove the validity and performance of the developed methodology a case study based on Barcelona sewer networks will be considered. It is an example of a complex system which contains continuous and discrete dynamics and it is subject to the influence of faults of different nature.

## 1.2 Contributions

The first contribution concerns the integration in the *hybrid automaton framework* of *nominal*, *structural faulty*, and *non-structural faulty* modes at the same time in order to provide an unified treatment of the different operation modes which are not previously treated in the current existing methodologies at the same time. The concept of discernibility is extended in a general manner for all modes of the hybrid model. So far, *discernibility* has been approached separately using the properties developed in [Cocquempot et al., 2004] or, concerning the case of non-structural faults, based on the concept of sensitivity and generating a fault signature matrix where *detectability* and *isolability* properties are defined [Meseguer et al., 2010a].

The complexity of the hybrid automaton model tends to blow up very fast if the set of modes is defined considering all possible system configurations. Usually the number of diagnoser states grows exponentially with the number of hybrid automaton modes. Thus, too much memory storage may be required and in many cases it might even be impossible to implement. Hence, the second contribution relies on building the hybrid automaton model in an incremental way using the concept of parallel composition of component automata [Cassandras and Lafortune, 2008] and generating a set of parametrized equations instantiated as a mode label function [Travé-Massuyès et al., 2009]. Some of the system components can be represented by an automaton describing faulty and nominal behavior. The *incremental hybrid automaton* model allows to only build the part of the model where the system is possibly operating in. Therefore, it avoids to build offline the entire *diagnoser* and *behavior automaton*. On the other hand, diagnosis is performed by interpreting events and measurements issued by the physical system directly on the hybrid automaton model. This interpretation leads to build the useful parts of the diagnoser incrementally, 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 modes.

The third contribution concerns some extensions in the methodology to improve the online diagnosis. On one hand, the inclusion of uncertainty in the system parameters using a robust strategy is proposed, where an adaptive threshold for residual evaluation is generated using the equivalence between parity space approach and input-output models. On the other hand, an extension of the proposed methodology that allows to diagnose hybrid systems using a diagnoser that reasons on components, which can be extended to nonlinear models and multiple fault detection hypothesis, has also been developed.

## 1.3 Objectives

- The global objective of the thesis is to develop a methodology to detect and isolate faults in hybrid systems that be applicable to large scale systems online.

This global objective will be achieved by accomplishing the following specific objectives:

1. To be aware about the current methodologies by providing a state of the art in hybrid system diagnosis.
2. To characterize a hybrid model, representing the nominal as well as the faulty system behavior, which involves the following issues:
   (a) Studying the dynamic model which characterizes the continuous dynamics.
   (b) Characterizing the kind of the faults present in real systems.
   (c) Characterizing model as well as measurement uncertainty.
   (d) Characterizing the different kinds of observable and unobservable events.
   (e) Considering system nonlinearities, and,
   (f) Considering multiple fault sequences.
3. To develop a methodology for the hybrid systems diagnosis. The methodology involves:
   (a) Defining a conceptual architecture for fault detection and isolation in hybrid systems.
   (b) Developing a method to diagnose hybrid systems by recognizing the current mode.
   (c) Characterizing the detectability and isolability of faults in hybrid systems.
   (d) Studying the viability and applicability of the method when system complexity increases.
4. To develop an algorithm to implement a hybrid diagnoser. The algorithm should:
   (a) Implement the modules designed in the conceptual architecture to recognize the current mode and detect and isolate faults.

(b) Formalize an algorithm for the hybrid diagnoser reasoning.

(c) Parameterize the hybrid diagnoser based on the hybrid model.

5. To prove the validity of the propose methodology by assessing the performance and the applicability in a large scale system as a sewer networks

   (a) Obtain a hybrid model for the sewer network.

   (b) Apply the diagnoser design methodology.

   (c) Implement the hybrid diagnoser for the sewer network.

   (d) Test the diagnosis methodology under several scenarios.

## 1.4 Outline of the thesis

The dissertation is organized as follows:

### Chapter 2: Background

This chapter aims to bring the main ideas about the different topics considered in this thesis and to review the state of the art. The first part introduces concepts and definitions in diagnosis for continuous and discrete-event systems. In the second part, the state of the art is mainly focused on hybrid system diagnosis.

### Chapter 3: Methodology to diagnose in hybrid systems

A design methodology and implementation architecture for diagnosers in the framework of hybrid systems is proposed. The design methodology is based on the hybrid automaton model that represents the system behavior by means of the interaction of continuous dynamics and discrete events. The architecture is composed of modules which carry out mode recognition and diagnostic tasks interacting each other, since the diagnosis module adapts accordingly to the current hybrid system mode. Both tasks interact each other since the diagnosis module adapts accordingly to the current mode of the hybrid system. The mode recognition task involves detecting and identifying a mode change by determining the set of residuals that are consistent with the current hybrid system mode.

The discernibility is the main property used in hybrid systems. Through the concept of discernibility it is possible to predict whether modes changes (faulty or nominal) in the hybrid model can be detected and isolated properly. This concept can be applied in practice, evaluating a set of mathematical properties derived from residual expressions, which can be input-output models or parity space equations. General properties are derived to evaluate the discernibility between modes in the hybrid automaton model.

**Related Publications**

J. VENTO AND V. PUIG AND R. SARRATE. Fault Detection and Isolation of Hybrid System using Diagnosers that combine Discrete and Continuous Dynamics. *Conference on Control and Fault Tolerant System*, Nice (France), 2010.

J. VENTO AND V. PUIG AND R. SARRATE. A Methodology for building a Fault Diagnoser for Hybrid Systems. *9th European Workshop on Advance Control and Diagnosis*, Budapest (Hungry) , 2011.

## Chapter 4: Incremental methodology to hybrid systems diagnosis

A methodology to track the system mode and diagnose a hybrid system without building a full diagnoser offline is presented. The methodology is supported by a hybrid automaton model that represents the hybrid system continuous and discrete behavioral dynamics. Diagnosis is performed by understanding the events and measurements issued by the physical system directly on the hybrid automaton model. This interpretation leads to incrementally build the useful parts of the diagnoser, developing only the traces that are needed to explain the occurrence of the incoming events. The resulting diagnoser adapts to the system operational life and is much less demanding in terms of memory storage.

**Related Publications**

J. VENTO AND V. PUIG AND R. SARRATE AND L. TRAVÉ-MASSUYÈS. Hybrid automaton incremental construction for online diagnosis. *The International Workshop on Principles of Diagnosis*, Jerusalem (Israel), 2013 (Poster).

## Chapter 5: Application case study

This chapter introduces the modeling principles for sewer networks by following a *virtual* tank approach. Indeed, a network can be considered as a set of interconnected tanks, which are represented by a first order model relating inflows and outflows with the tank volume. Besides, the corresponding hybrid automaton model can be obtained based on the automata composition. Once the structure and operation modes of sewer networks are introduced, a validation of the incremental hybrid system diagnosis is presented. The main advantages and disadvantages of the incremental and non-incremental methodologies are analyzed in the application case study.

J. VENTO AND L. TRAVÉ-MASSUYÈS AND V. PUIG AND R. SARRATE. An incremental diagnoser automaton for hybrid systems enhanced by discernability properties. Submitted to *IEEE Transactions on Systems, Man and Cybernetics*. March (2014)

## Chapter 6: Extensions to the fault diagnosis methodologies for hybrid systems

The methodology to detect and isolate faults developed in previous chapters can be improved considering some aspects neglected so far as robustness and nonlinearities are always present in a system and assuming fault models are known.

Regarding uncertainty, a method for hybrid system diagnosis using a parity space approach that considers model uncertainty is proposed. The methodology takes into account the parameter uncertainty using a passive robust strategy. An adaptive threshold for residual evaluation is generated and the parity space approach is used to design a set of residuals for each mode.

In the second case, the design methodology is based on the hybrid automata model that represents the system behavior, in which each mode relates to a set of components. The architecture includes a set of modules which achieve mode recognition and diagnosis tasks both based on residuals generated by structural analysis. Diagnosis involves detecting and isolating faults by interlinking the components underlying the inconsistencies reported by the residuals, following the DX approach. The logic applied to detect and isolate faults allows to make hypothesis regarding multiple fault occurrence and to detect non-modeled faults using a component oriented fault diagnosis approach.

**Related Publications**

J. VENTO AND V. PUIG AND R. SARRATE. Parity Space Hybrid System Diagnosis under Model Uncertainty. *20th Mediterranean Conference on Control and Automation (MED)*, Barcelona (Spain), 2012.

J. VENTO AND V. PUIG AND R. SARRATE AND L. TRAVÉ-MASSUYÈS. Fault Detection and Isolation of Hybrid Systems using Diagnosers that Reason on Components. *8th IFAC Symposium Safeprocess*, Mexic (City, Mexic) , 2012.

J. VENTO AND J. BLESA J AND V. PUIG V. AND R. SARRATE.. Set Membership Parity Space Hybrid System Diagnosis. Submitted to *International Journal of Systems Science*. March (2014)

## Chapter 7: Conclusions

This chapter summarizes the contributions made in this thesis and discusses the ways for future research directions.

# CHAPTER 2

# BACKGROUND ON HYBRID SYSTEMS DIAGNOSIS

## 2.1 Review of background theory

Two different communities (FDI community in the Automatic Control and Statistics area and DX community in the Artificial Intelligence area) have developed their own methodologies for model-based fault diagnosis, one independently of the other.

The FDI community has its roots in the classical theory of systems and automatic control, which develops control and statistic decision theories for model-based diagnosis using analytic models and linear algebra [Chow and Willsky, 1984, Staroswiecki and Comtet-Varga, 2001].

On the other hand, the DX community has its roots in consistency-based diagnosis [Reither, 1987] that uses symbolic and qualitative models with logic for diagnosis tasks. In fact, some equivalences between both approaches have been demonstrated [Biswas et al., 2004].

One of the principal differences among model-based diagnosis approaches is the type of model used to detect and isolate possible faults. Hence, it influences the diagnosis techniques and the diagnostic precision. The system model depends on the nature of the system behavior:

1. *Discrete-event systems*: The system behavior is represented by a set of events. It can be modeled by logic formulas or finite states machines. Moreover, diagnosis schemes are based on analyzing observed event sequence [Cassandras and Lafortune, 2008, Sampath et al., 1995].

2. *Continuous systems*: The type of models used to describe the process are differential or difference equations depending on whether the model is described on continuous or in discrete-time. These equations are derived from the analysis of the physical laws governing the continuous variable behaviors. Then, quantitative and/or qualitative methods for diagnosis can be applied

[Gertler, 1997, Patton and Chen, 1997, Isserman, 1997].

3. *Hybrid systems*: Hybrid systems combine both continuous and discrete event dynamics. In classical FDI methods, diagnosis is performed separately on the continuous dynamics and on the discrete event dynamics. For this reason, these techniques neglect the interaction between both dynamics, resulting in poor diagnosability. Therefore, model-based diagnosis methods can be combined if the system is represented by a hybrid model

Complex systems are subject to faults that can appear in any plant component, sensor or actuator (see Fig. 2.1). Two type of faults are considered: *structural faults* and *non-structural faults*. *Structural faults* refer to faults that can be represented by a dynamical model. In Fig. 2.1, a fault in a component is denoted by $f_{\mathcal{M}_j}$. Examples of these faults include a valve or a switch in a stuck position.

*Non-structural faults* refer to faults that alter parameter values without changing the structure of the model. Additive faults like those concerning sensors and actuators are typical non-structural faults. In Fig. 2.1, these faults are denoted as $f_u$ (input sensor fault), $f_y$ (output sensor fault) and $f_a$ (actuator fault), respectively.

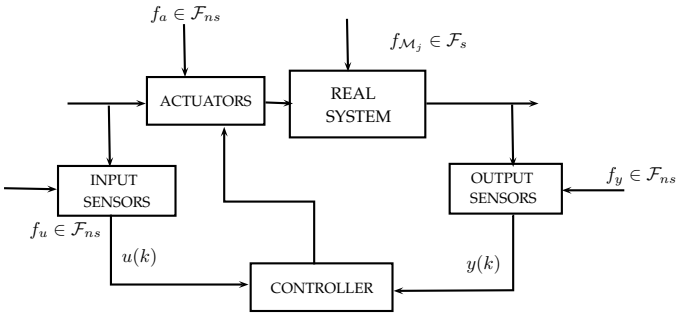


Figure 2.1: Fault classification scheme

## 2.2 Model-based diagnosis based on continuous variable models

Model-based diagnosis in FDI relies on comparing the estimated behavior of the system obtained from a non-faulty model with the real behavior available through sensor measurements [Puig et al., 2004]. FDI embeds three separate tasks:

- Fault detection deals with the generation of a residual

$$\mathbf{r}(k) = \mathbf{y}(k) - \widehat{\mathbf{y}}(k) \tag{2.1}$$

as a means to compare real and predicted output using any of the available model based methods (observers, parity equation, among others) such that in the absence of the fault $\mathbf{r}(k) = 0$ and it deviates from zero when there is a fault. Once the residual has been generated, it is evaluated against a threshold to detect the fault presence

$$\phi^j(k) = \begin{cases} 0 & \text{if } |r^j(k)| \leq \tau^j \quad \text{(no fault)} \\ 1 & \text{if } |r^j(k)| > \tau^j \quad \text{(fault)} \end{cases} \tag{2.2}$$

where $j \in \{1, \cdots, n_r\}$, $\tau^j$ is the threshold associated to the residual $r^j(k)$ generating the observed fault signature $\mathbf{\Phi}(k) = [\phi^1(k), \cdots, \phi^{n_r}(k)]$.

- Fault isolation compares the observed fault signature with the theoretical fault signature, that records the effect of the considered set of faults in each residual, to identify which is the fault that could have lead to the activation of a subset of them.

- Fault estimation consists in determining the fault magnitude and historic evolution through the *sensitivity* concept.

Fig. 2.2 illustrates the conceptual scheme for diagnosis.

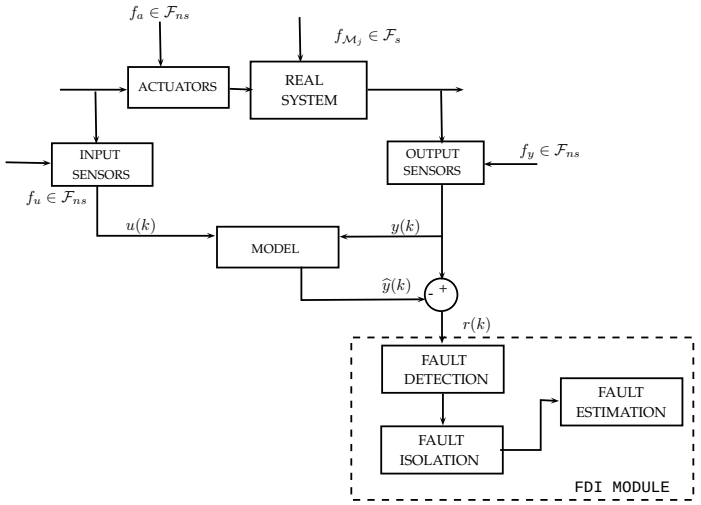


Figure 2.2: FDI scheme for continuous systems

### 2.2.1 Residuals generation methods

Consider the linear system represented by the state space model in discrete-time:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{F}_x\mathbf{f}(k) \quad (2.3)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) + \mathbf{F}_y\mathbf{f}(k) \quad (2.4)$$

where $\mathbf{x}(k)$, $\mathbf{u}(k)$ and $\mathbf{y}(k)$ are the continuous state, input and output vector with dimensions $n_x$, $n_u$ and $n_y$ respectively, $\mathbf{A} \in \mathcal{R}^{n_x \times n_x}$, $\mathbf{B} \in \mathcal{R}^{n_x \times n_u}$, $\mathbf{C} \in \mathcal{R}^{n_y \times n_x}$, $\mathbf{D} \in \mathcal{R}^{n_y \times n_u}$, $\mathbf{F}_x$ and $\mathbf{F}_y$ are the fault distribution matrix.

The predicted output, using the parity space approach [Blanke et al., 2006], corresponding to time instants $1, \cdots, \rho$ in matrix form is represented by:

$$\bar{Y}(k) = \mathbf{O}\mathbf{x}(k-\rho) + \mathbf{T}_{u,\rho}\bar{\mathbf{U}}(k) + \mathbf{T}_{f,\rho}\bar{\mathbf{F}}(k) \quad (2.5)$$

where $\bar{\mathbf{Y}}(k) = \begin{bmatrix} \mathbf{y}(k-\rho) & \mathbf{y}(k-\rho+1) & \cdots & \mathbf{y}(k) \end{bmatrix}^T$ and $\bar{\mathbf{U}}(k)$ and $\bar{\mathbf{F}}(k)$ are similar vectors, and $\rho$ is the parity space order. The parity space matrices are given by:

$$\mathbf{O} = \begin{pmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{\rho} \end{pmatrix} \qquad \mathbf{T}_{u,\rho} = \begin{pmatrix} \mathbf{D} & 0 & \cdots & 0 & 0 \\ \mathbf{CB} & \mathbf{D} & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ \mathbf{CA}^{\rho-1}\mathbf{B} & & \cdots & \mathbf{CB} & \mathbf{D} \end{pmatrix}$$

$$\mathbf{T}_{f,\rho} = \begin{pmatrix} \mathbf{F}_y & 0 & \cdots & 0 & 0 \\ \mathbf{CF}_x & \mathbf{F}_y & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ \mathbf{CA}^{\rho-1}\mathbf{F}_x & & \cdots & \mathbf{CF}_x & \mathbf{F}_y \end{pmatrix}$$

The computational form of the residual is given by:

$$\mathbf{r}(k) = \mathbf{W}\bar{\mathbf{Y}}(k) - \mathbf{W}\mathbf{T}_{u,\rho}\bar{\mathbf{U}}(k) \quad (2.6)$$

where $\mathbf{W}$ is a $n_r \times (\rho+1)n_y$ matrix such that $\mathbf{WO} = 0$ in order to to eliminate the dependence on $\mathbf{x}(k)$.

For input-output models, the predicted output can be expressed under the following general form [Meseguer et al., 2010a]:

$$\widehat{\mathbf{y}}(k) = \mathbf{G}(p^{-1})\mathbf{u}(k) + \mathbf{H}(p^{-1})\mathbf{y}(k) \quad (2.7)$$

where $p^{-1}$ denotes the delay operator, and **G** and **H** are designed to satisfy the condition:

$$\mathbf{G}(p^{-1}) = (\mathbf{I} - \mathbf{H}(p^{-1}))\mathbf{M}(p^{-1}) \tag{2.8}$$

The input-output model can represent a predictor, an observer or a simulator.

## 2.3 Model-based diagnosis techniques based on discrete-event systems

### 2.3.1 System model

The system is composed by a set of components, denoted by $COMP$, connected according to the structure of the system. The discrete-event behavior of a component is represented by an automaton. An automaton is a device capable of representing a language which represents system states and transitions between them triggered by events [Cassandras and Lafortune, 2008] .

Formally, a deterministic automaton is defined as $DA =< \mathcal{Q}_\mathcal{M}, \Sigma_\mathcal{M}, \mathcal{T}_\mathcal{M}, q_{\mathcal{M}_0} >$, where:

- $\mathcal{Q}_\mathcal{M}$ is the set of discrete states. A discrete state can represent a nominal or faulty state of a component.
- $q_{\mathcal{M}0}$ is the initial state.
- $\Sigma_\mathcal{M}$ is the set of events associated to the component automaton which can be unobservable or observable.
- $\mathcal{T}_\mathcal{M} : \mathcal{Q}_\mathcal{M} \times \Sigma_\mathcal{M} \rightarrow \mathcal{Q}_\mathcal{M}$ is the transition function. $\mathcal{T}_\mathcal{M}(q_{\mathcal{M}_i}, \sigma) = q_{\mathcal{M}_j}$ means there is a transition labeled by event $\sigma$ from state $q_{\mathcal{M}_i}$ to state $q_{\mathcal{M}_j}$.

The behavior of a discrete-event system is described by a string of events (called trajectory): $s = s_1 s_2 \cdots s_n$ where $s_j \in \Sigma_\mathcal{M}$ and $j \in \mathcal{N}^+$. The set of all possible trajectories forms a prefix-closed language over the alphabet $\Sigma_\mathcal{M}$, denoted by $L(DA)$. $L(DA)$ is a subset of $\Sigma^*_\mathcal{M}$, where $\Sigma^*_\mathcal{M}$ denotes the set of all finite strings of elements of $\Sigma_\mathcal{M}$ included the empty set $\epsilon$ termed the Kleene-closure of $\Sigma_\mathcal{M}$ [Ramadge and Wonham, 1989].

Through the system component representation using automata, many operations can be performed. One of the common composition operations on automata is the parallel composition, which allows to express the interaction between system components.

Given two automata $DA_1$ and $DA_2$ the parallel composition is formally defined as:

$$DA_1||DA_2 = Ac(\mathcal{Q}_{\mathcal{M}_1} \times \mathcal{Q}_{\mathcal{M}_2}, \Sigma_{\mathcal{M}_1} \cup \Sigma_{\mathcal{M}_2}, \mathcal{T}_{1||2}, (q_{\mathcal{M}_{10}}, q_{\mathcal{M}_{20}})$$

$$\mathcal{T}_{1||2}((q_1.q_2), \sigma_{\mathcal{M}}) = \begin{cases} (\mathcal{T}_1(q_1, \sigma_{\mathcal{M}}), \mathcal{T}_2(q_2, \sigma_{\mathcal{M}})) & \text{if } \sigma_{\mathcal{M}} \in \Gamma_1(q_1) \cap \Gamma_2(q_2) \\ (\mathcal{T}_1(q_1, \sigma_{\mathcal{M}}), q_2) & \text{if } \sigma_{\mathcal{M}} \in \Gamma_1(q_1) \backslash \Sigma_{\mathcal{M}_2} \\ (q_1, \mathcal{T}_2(q_2, \sigma_{\mathcal{M}})) & \text{if } \sigma_{\mathcal{M}} \in \Gamma_2(q_2) \backslash \Sigma_{\mathcal{M}_1} \\ \text{undefined} & \text{otherwise} \end{cases} \tag{2.9}$$

Parallel composition uses the active event function defined by $\Gamma_{\mathcal{M}} : \mathcal{Q}_{\mathcal{M}} \rightarrow 2^{\Sigma_{\mathcal{M}}}$. It contains the set of all possible events $\sigma_{\mathcal{M}} \in \Sigma_{\mathcal{M}}$ such that $\mathcal{T}_{\mathcal{M}}(q_{\mathcal{M}}, \sigma_{\mathcal{M}})$ is defined. The event set of each component includes private events that pertain to its own internal behavior. These events can be observable, unobservable or faulty events. Faulty events represent structural fault occurrences.

Another common operation is computing the successors states of a given automaton state. This operations is very useful to build the global system model and to perform diagnosis as explained later. State successors are denoted by $Succs(q_i) = \{q_j \in \mathcal{Q}_{\mathcal{M}} : \exists \sigma \in \Sigma_{\mathcal{M}} : \mathcal{T}_{\mathcal{M}}(q_{\mathcal{M}_i}, \sigma) = q_{\mathcal{M}_j}\}$.

Other ways to model a discrete-event system is using Petri networks [Zhao et al., 2005] and Semi-Markov models [Dong and He, 2006] among others.

### 2.3.2 Discrete-event diagnosis

Discrete-event system diagnosis consists in detecting and isolating fault events, based on the observation of observable events and the building of a *diagnoser* [Sampath et al., 1995]. The diagnoser performs diagnostics using online observations of the system behavior; it is also used to state and verify offline the necessary and sufficient conditions for diagnosability. Structural and non-structural faults are handled by discrete-event systems as unobservable events in the system model that they are detected through the identified observable events.

A diagnoser is the finite state machine $D =< \mathcal{Q}_D, \Sigma_D, \mathcal{T}_D, q_{D_0} >$, where:

- $q_{D_0} = \{q_{\mathcal{M}_0}, \emptyset\}$ is the initial state of the diagnoser, which is assumed that corresponds to a nominal system state.
- $\mathcal{Q}_D$ is a set of the diagnoser states. An element $q_D \in \mathcal{Q}_D$ is a set of the form $q_D = \{(q_1, l_1), (q_2, l_2), \cdots (q_n, l_n)\}$, where $q_i \in \mathcal{Q}_{\mathcal{M}}$ and $l_i \in \Delta_{\mathcal{F}_s}$ where $\Delta_{\mathcal{F}_s}$ defines the power set of fault labels $\Delta_{\mathcal{F}} = \Delta_{\mathcal{F}_s} \cup \Delta_{\mathcal{F}_{ns}}$ with $\Delta_{\mathcal{F}_s} = \{f_1, \cdots, f_\gamma\}$, and $\Delta_{\mathcal{F}_{ns}} = \{f_1^*, \cdots, f_\mu^*\}$ respectively, $\gamma + \mu$ is the total number of faults in the system and $\gamma, \mu \in \mathcal{Z}^+$. In $\Delta_{\mathcal{F}}$, $\emptyset$ represents nominal behavior,
- $\Sigma_D = \Sigma_{\mathcal{M}_o}$ is a set of all observable events in $DA$,
- $\mathcal{T}_D : \mathcal{Q}_D \times \Sigma_D \mapsto \mathcal{Q}_D$ is a partial transition function of the diagnoser.

The transition function is given by:

$$\mathcal{T}_D(q_D, \sigma) = \bigcup_{\substack{(q,l) \in q_D \\ s \in L_\sigma(DA, q_D)}} \{\mathcal{T}_r(q, s), LP(q, l, s)\}$$

where $L_\sigma(DA, q) = \{s \in L(DA, q) : s = u\sigma, u \in \Sigma^*_{\mathcal{M}uo}, \sigma \in \Sigma_{\mathcal{M}o}\}$ denoting the set of all strings that originates from state $q$ and ends at the first observable event, and $L(DA, q) = \{s \in L_\sigma(DA, q) : s_f = \sigma\}$ denotes those strings in $L_o(DA, q)$ that end at the particular observable event $\sigma$, with $s_f$ denoting the final event of a string $s$. $\mathcal{T}_r$ is the recursive application of $\mathcal{T}_\mathcal{M}$ along string $s$, i.e. considering $s = \sigma_1.\sigma_2.\cdots, \sigma_n\sigma$ then $\mathcal{T}_r(q, s) = \mathcal{T}_\mathcal{M}(\mathcal{T}_\mathcal{M}(...\mathcal{T}_\mathcal{M}(\mathcal{T}_\mathcal{M}(q, \sigma_1), \sigma_2)...), \sigma_n), \sigma)$.

The label propagation function is defined as: $LP : \mathcal{Q}_{\mathcal{M}o} \times \Delta \times \Sigma^*_\mathcal{M} \to \Delta$ where $\mathcal{Q}_{\mathcal{M}o} = \{q_0\} \cup \{q \in \mathcal{Q}_\mathcal{M}, \exists(q', \sigma) \in \mathcal{Q}_\mathcal{M} \times \Sigma_{\mathcal{M}o} : \mathcal{T}(q', \sigma) = q\}$. $LP$ propagates the label $l$ over $s$, starting from $q$ and following the dynamic $DA$, i.e., according to $L(DA, q)$

$$LP(q, l, s) = \begin{cases} \emptyset & \text{if } l = \emptyset \text{ and } \forall i, f_i \notin s \\ \{f_i | f_i \in l\} \cup \{f_i | f_i \in s\} & \text{otherwise} \end{cases}$$

**Example 2.1.** *Consider the control gate model described by the $DA$ in Fig. 2.3. Observable events are open_valve, close_valve and structural faulty events are fail_open and failt_close. These faulty events are related to state events stuck_open and stuck_close in the automaton. Then, the propagation algorithm provides two versions for the diagnoser as is shown in Fig. 2.4 and Fig. 2.5.*

Fig. 2.4 represents a diagnoser without silent closure and Fig. 2.5 represents a diagnoser with silent closure. The difference between both diagnoser versions concerns the way the propagation algorithm is executed. In the first case, a state contains a set of diagnosis candidates, each candidate $q_{\mathcal{M}_i} \in \mathcal{Q}_\mathcal{M}$ contained in $q_{Di}$ asserts that the system may be in state $q_{\mathcal{M}_i}$ and the set of faults $\mathcal{F}$ may have happened before reaching state $q_{\mathcal{M}_i}$ after a given sequence of observations. State $q_{\mathcal{M}_i}$ is the target state of a transition associated to the last observation of the sequence. In the second case, the diagnoser is extended to also perform prediction on the silent part of the system after state $q_{\mathcal{M}_i}$. Thus, the diagnoser states also contain the set of possible states and faults that can occur after the last observable occurrence [Pencolé, 2012]. The silent part means all these states that can be reached through unobservable events.

Notice that in Figs. 2.4 and 2.5, the diagnosers differ in the initial state. The diagnoser in Fig. 2.5 shows all possible states, including faulty states. Remark that the diagnoser with silent closure propagates those branches in the system model that end up with an unobservable event.

The choice of one diagnoser or other depends on the system information the operator wishes to include. One criterion might be the degree of uncertainty in the system. In some cases the initial state of the system is unknown, hence, the diagnoser with silent closure is more appropriate. Some of the faults

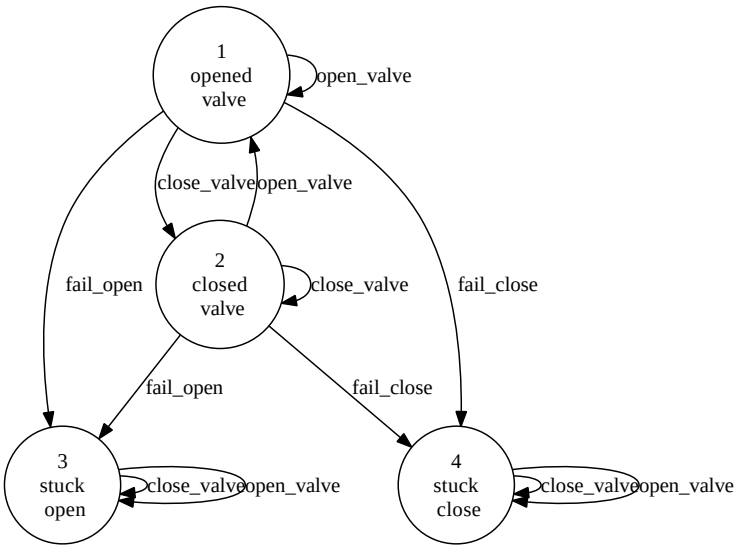


Figure 2.3: Control valve model

in the system can be non-detectable, therefore, the knowledge of which faults belong to this subset can be of interest. In general terms, the diagnoser without silent closure includes a prediction of possible future states. On the contrary, the one with silent closure is only an estimation based on the past.

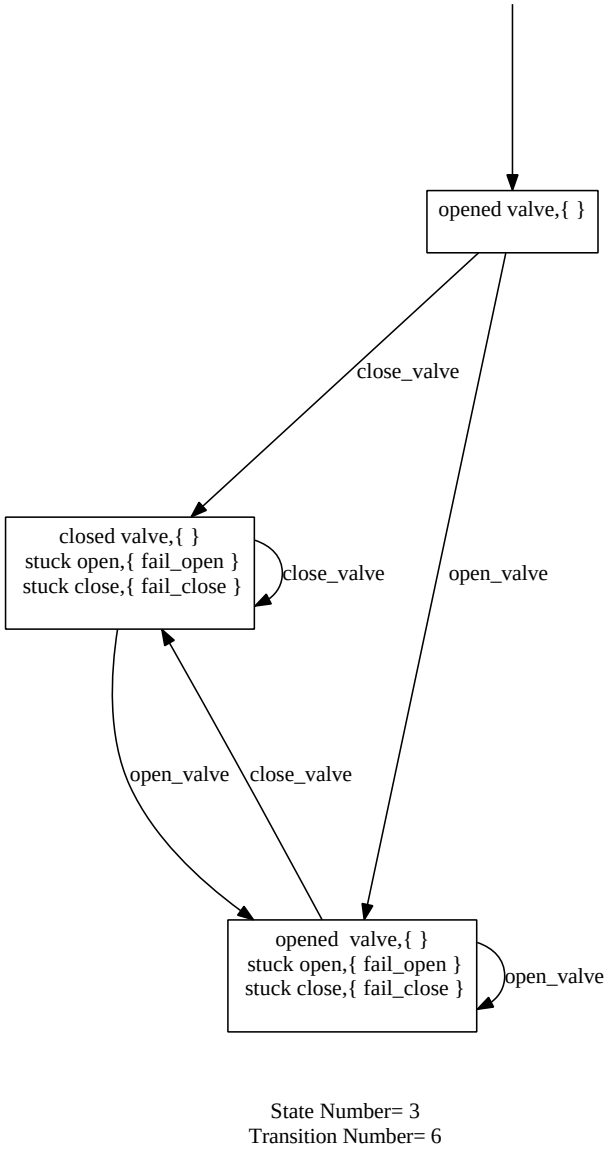


Figure 2.4: Diagnoser without silent closure

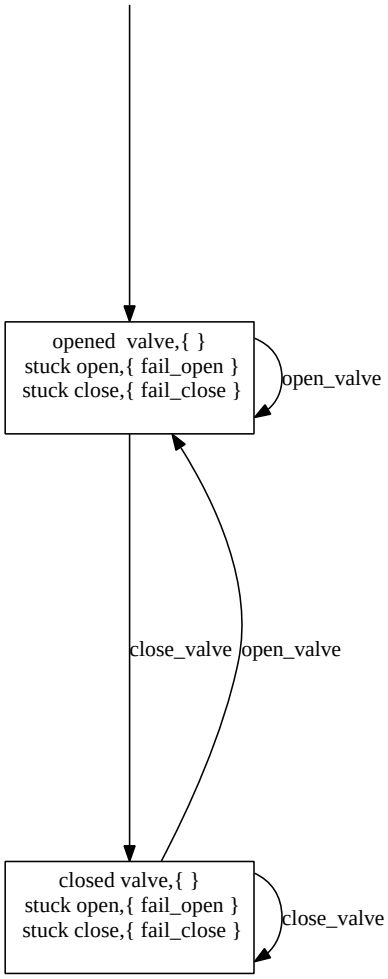


Figure 2.5: Diagnoser with silent closure

## 2.4 Hybrid system diagnosis

### 2.4.1 Hybrid automaton model

Hybrid automaton is a modeling formalism for hybrid systems that results from an extension of finite-state machines by associating with each discrete state a continuous-state model. Conditions on the continuous evolution of the system invoke discrete state transitions.

A hybrid automaton is a dynamical system that describes the evolution in time of the values of a set of discrete and continuous state variables [Lygeros et al., 2003]:

**Definition 2.1.** An hybrid automaton is $HA$ is a collection $HA =< \mathcal{Q}, \mathcal{X}, f, Init, D, E, G, R >$, where

- $\mathcal{Q}$ is a set of discrete states.
- $\mathcal{X}$ is a set of continuous states.
- $f(\cdot,\cdot) : \mathcal{Q} \times \mathcal{X} \rightarrow \mathcal{R}^n$ is a vector field.
- $Init \subseteq \mathcal{Q} \times \mathcal{X}$ is a set of initial states.
- $Dom(\cdot) : \mathcal{Q} \rightarrow P(\mathcal{X})$ is a domain.
- $E \subseteq \mathcal{Q} \times \mathcal{Q}$ is a set of edges.
- $G(\cdot) : E \rightarrow P(\mathcal{X})$ is a guard condition.
- $R(\cdot,\cdot) : E \times \mathcal{X} \rightarrow P(\mathcal{X})$ is a reset map

A hybrid automaton models real behavior of the 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 algebraic differential equations which constrain continuous state, input and output variables. Input and output variables are measured.

The hybrid diagnosis approaches based on the hybrid automaton [Hofbaur and Williams, 2004, Benazera and Travé-Massuyès, 2009, Mezyani, 2007, Vento et al., 2011, Bayoudh et al., 2008] adapts the hybrid automaton introduced in Definition 2.1, taking into account discrete events and fault events in the model. Events may be observable or unobservable. Hence, 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 fault events or other events that cause changes in the system state not directly recorded by sensors.

### 2.4.2 Hybrid systems diagnosis approaches

Hybrid system diagnosis is an extension of the classical approaches developed for fault detection and isolation based on models. Its interest has increased in the last years since the majority of real complex systems are online controlled and supervised by means of automatic computer-based control systems.

The behavior of those systems is composed of continuous plant dynamics described by continuous state variables and a supervisory controller that generates actuator signals at discrete time instants to change regulator set points or the plant configuration. Therefore, the system behavior changes according to the operation mode. Model-based online diagnosis requires quick and robust reconfiguration precess when a mode change occurs, as well as the ability to keep the nominal behavior of the system on track during transient states [Bregon et al., 2012].

In the FDI approach, diagnosis is based on a hybrid automaton model [Hofbaur and Williams, 2004] to track the system mode, such is the case of multiple model filtering methods [Georges et al., 2011, Blom and Bar-Shalom, 1988] and particle filtering methods [de Freitas, 2002], where hybrid automaton models have long been restricted to hybrid estimation schemes exemplified in [Hofbaur and Williams, 2004, Benazera and Travé-Massuyès, 2009]. Only later, hybrid diagnosis approaches combined the discrete part of the hybrid model with parity-space residuals [Mezyani, 2007, Vento et al., 2011, Bayoudh et al., 2008]. The method presented in these works tends towards the building of a finite state machine called a *diagnoser* [Sampath et al., 1995], which is built offline from the hybrid model, and residuals are generated for each mode as explained in [Vento et al., 2011] and [Bayoudh et al., 2008].

The general architecture in hybrid system diagnosis is illustrated in Fig. 2.6. Two stages are considered. In the first stage, the diagnosis system is designed offline based on the hybrid model. The second stage involves the implementation of the diagnosis system based on the online evaluation of the set of residuals. Diagnosis consists of tracking the system mode and detecting and isolating possible faults. The tasks of these modules are based on the residuals analysis and observable discrete-event occurrence.

In [Cocquempot et al., 2004], a hybrid automaton model comprising only nominal operation modes is proposed. The system mode is recognized by checking consistency of the whole set of ARRs generated considering all system modes. The set of ARRs that are consistent with the hybrid system current mode allows to identify it.

The concept of *non-discernibility* is introduced for first time in this work. Necessary and sufficient conditions are provided for the parity space approach in state space representation, guaranteeing the correct mode tracking provided that all system modes are discernable between them. Fault detection and isolation is based on identifying inconsistencies between the measured and estimated system behavior by means of the (ARRs) and the current operation mode.

Later, in [Bayoudh et al., 2008], hybrid automaton include nominal and faulty modes. The type of faults considered are *structural faults*. Besides, every operating mode is indeed characterized by a

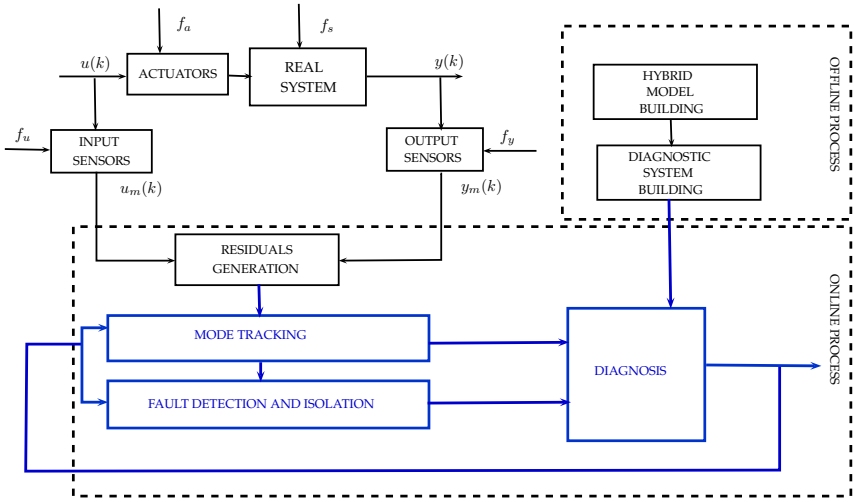


Figure 2.6: General scheme for hybrid system diagnosis

*mode signature*. Signatures are abstracted in terms of discrete events typifying continuous dynamics. *Non-discernibility* is based on the *mode signature* concept. A *mode signature* is built taking into account the whole set of ARRs generated considering all system modes using the parity space approach. Then, a discrete-event automaton called *behavior automaton* is generated, allowing to build a hybrid diagnoser that takes as inputs all observable events and it is built applying the theory presented in [Sampath et al., 1995] for discrete-event systems.

An extension of both methodologies is then proposed in [Vento et al., 2010], considering in the hybrid automaton model only nominal operation modes and the tracking mode through ARRs generated using input-output models. In the *behavior automaton* building process, *non-structural faults* are included as faulty modes, such that each one is associated with a *fault signature* derived from the *sensitivity* concept. *Non-discernibility* conditions are derived from input-output models and an equivalence between the parity space conditions is proved.

On the other hand, in the DX approach, some authors have proposed alternative ways to diagnosing hybrid systems as the hybrid bond graph formalism [Narasimhan and Biswas, 2007, Daigle, 2008]. Unlike hybrid automata models, pre-enumeration of all system modes, is avoided by generating models at runtime as mode switches occur. Hybrid bond graphs (HBGs) are a domain-independent topological-modeling language that capture energy-based interactions among the processes that make up a physical system in a graphic form. Then, an observer is used to track system behavior and provide a nominal reference for diagnosis. The diagnosis algorithms are based on the *TRANSCEND* and *HYBRID TRANSCEND* methodologies to check consistency between measurements and nominal behavior. The hybrid bond graph is built manually through a interface taking into account the global model.

A comparison among the main approaches is shown in Table 2.1, where, advantages and limitations

| Approach (Authors) | | FDI a) [Mezyani, 2007, Cocquempot et al., 2004] | FDI b) [Bayoudh et al., 2008] | DX [Daigle, 2008] |
|---|---|---|---|---|
| Hypothesis | Type of fault | Non-structural (Additives)<br>Discrete faults | Structural | Structural (parametric and discrete) |
| | Models | Linear and continuous | Linear and discrete | Linear |
| | Uncertainty | Not | | |
| | Noise | Not | Not | Yes |
| | Disturbances | Not | Not | Yes |
| | Delays | It is not taken into account | | Yes |
| | Multiples faults | Not | Not | Yes |
| | Nominal mode change and fault do not occur at the same time | | | |
| | Assume known initial state | | | |
| Mode tracking | Mode change | Residual test consistence | Residual test consistence and observable discrete events | Analysis in the transient response of residuals |
| | Identification | Set of residuals consistent with the current mode | Mode signature | Set of residuals consistent with the current mode |
| Fault detection and isolation | Fault detection | Residual test inconsistence | Residual test consistence | Changes in the transient response of residuals |
| | Isolation | Fault signature | Signature of the faulty mode | Analysis of the local diagnosers containing fault information |

Table 2.1: Comparison between the proposed approaches

of the methods are provided. The methods are compared based on the hypothesis made in each approach such as the type of faults treated, the continuous dynamics description, whether diagnosis takes into account uncertainty, noise and disturbances among others. The second part of the table describes how online diagnosis is carried out. The techniques to detect and isolate possible faults and to track the system mode are briefly enumerated.

Other approaches to hybrid system diagnosis are based on *Mixed Logic Dynamical*(MLD) models [Mignone, 2002, Heemels et al., 2001]. The evolution of an MLD model is governed by linear dynamic equations subject to linear mixed integer inequalities, i.e. inequalities involving both continuous and binary variables. Binary variables represent the discrete-valued components and they are introduced according to logical inference techniques used in operations research. The main idea of this method is fault estimation based on an optimization problem MIQP, where the fault effect on the system is modeled as logic propositions and then converted into mixed integer inequalities.

Another approache that can be cited in the hybrid system diagnosis concerns Petri nets [Zhao et al., 2005]. The Petri net has two main functions: detecting faults based on the deviations between observed sensor events and their expected values and providing prior probabilities to the mode estimation algorithm. When a fault occurs, the deviation from the Petri net simulation triggers the decision-tree diagnoser. This task is analogous to residual generation in observer-based diagnosis schemes.

# CHAPTER 3

# METHODOLOGY FOR HYBRID SYSTEM DIAGNOSIS

## 3.1 Principles of the methodology

The architecture to detect and isolate faults in hybrid systems in Fig. 3.1 is an adaptation of the general scheme presented previously in Fig. 2.6. There is a need to include in the classic FDI conceptual block, a model that using system inputs and outputs allows to recognize the system mode and to adapt online the FDI module.

Hybrid diagnosis is based on the hybrid automaton framework. In particular, FDI algorithms take into account which is the current operation mode $q_i$ of the hybrid system to adapt the model used to generate the predicted output. Two separate stages are considered for hybrid systems diagnosis: offline and online processes.

In the offline process, the hybrid automaton model is built through the component parallel composition and the generation of a set of equations which depends on the operation mode. Residuals for each mode are generated and an exploration of the feasible hybrid automaton traces is carried out to study mode discernibility. Therefore, discernibility study and observable events of the system allow to build a behavior automaton ($B$), where diagnosability of the hybrid system is completely contained in $B$. This information is used to predict which mode changes can be detected and isolated. Hence, a diagnoser is built from $B$ applying propagation algorithms described for discrete-event systems to perform online diagnosis.

On the other hand, in the online process, the tasks are carried out by the three blocks highlighted in blue in Fig. 3.1. *Mode recognition* and *fault diagnosis* blocks deal with possible changes in the system operation mode based on consistency indicators and observable event occurrences. Both blocks cooperate together. The *diagnoser decision* block gives a final diagnostic according to information provided by

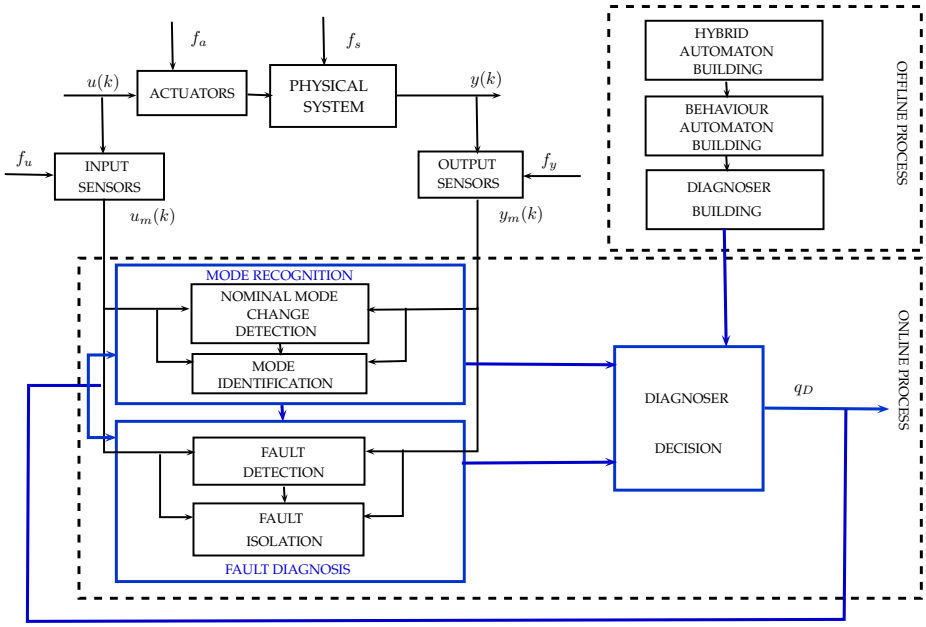


Figure 3.1: Conceptual block diagram for the hybrid diagnosis methodology

*mode recognition* and *fault diagnosis* blocks.

The current diagnoser state $(q_D)$ contains information on all modes the system is possibly operating in. If more than one mode is contained in $q_D$, those modes are non discernible. A mode change in $HA$ implies a nominal, structural faulty or non-structural faulty mode change. In the online diagnosis, a set of events are identified describing a feasible trajectory of the physical system.

*Discernibility* property has been used to predict if a mode change can be detected and identified when the operation mode is described by a dynamic model [Bayoudh et al., 2008, Meseguer et al., 2010b, Cocquempot et al., 2004]. In the case of non-structural faults, discernibility properties are related to *detectability* and *isolability* based on the fault signature matrix [Meseguer et al., 2010b]. 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).

Systematically, the steps followed to design a methodology for hybrid systems is shown in Fig. 3.2.

In online diagnosis, the following assumptions are made:

**Assumption 3.1.** *Two modes changes do not occur at the same time.*

**Assumption 3.2.** *The residual dynamics have time to stabilize between two consecutive mode switchings.*

Assumption 3.2 implies that transitions between modes should be slower than the residual dynamics

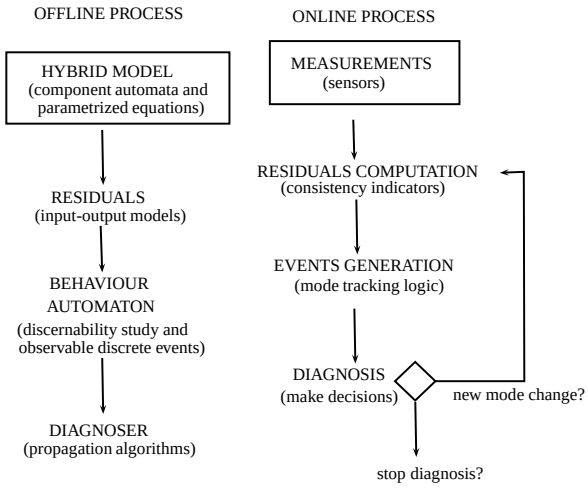


Figure 3.2: Design methodology steps.

generator. This concerns the dwell time requirement, the time elapsed to reach the steady state in a stable way needed by the continuous dynamics of the operation modes before other transitions occur. Otherwise, the transition might not be correctly detected.

**Assumption 3.3.** *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 3.4.** *No mode change will not occur after a non-structural fault has been occurred.*

Once a non-structural fault has been detected, the online diagnosis process stops since it is assumed that the system does not further evolve. Whenever a non-structural fault occurs the set of residuals and models must be adapted to appropriately perform diagnosis. In the case of a structural fault occurrence, the diagnosis tasks can continue even if the system is not repaired.

## 3.2 Hybrid modeling

An hybrid system is a system that combines continuous dynamics with discrete-event dynamics. In general terms, continuous dynamics are described by a set of difference or differential equations, and discrete-event dynamics are described by an automaton.

The behavior of a component $\mathcal{M}_j \in COMP$ in a mode $q_i$ is governed by a linear affine equation (algebraic or differential), which is parametrized with the mode. Examples of these components are sensors, actuators, control gates, switches, valves, tanks and many others that depend on the system nature. More examples are given in [Mezyani, 2007].

The methodology proposed for hybrid system diagnosis relies on the hybrid automaton model. Assume that the model of the hybrid system to be diagnosed is described by the following hybrid automaton:

$$HA =< \mathcal{Q}, \mathcal{X}, \mathcal{U}, \mathcal{Y}, \mathcal{F}, \mathcal{G}, \mathcal{H}, \Sigma, \mathcal{T} > \tag{3.1}$$

where:

- $\mathcal{Q}$ is a set of modes. Each $q_i \in \mathcal{Q}$ with $|\mathcal{Q}| = n_q$ represents a nominal operation, structural faulty mode or non-structural faulty mode of the system i.e. $\mathcal{Q} = \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s} \cup \mathcal{Q}_{\mathcal{F}_{ns}}$.
- $q_0 \in \mathcal{Q}$ is the initial mode.
- $\mathcal{X} \subseteq \mathcal{R}^{n_x}$ 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.
- $\mathcal{U} \subseteq \mathcal{R}^{n_u}$ defines the continuous input space. $\mathbf{u}(k) \in \mathcal{U}$ is the discrete-time input vector.
- $\mathcal{Y} \subseteq \mathcal{R}^{n_y}$ 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 $\mathcal{F} = \mathcal{F}_s \cup \mathcal{F}_{ns}$. Every faulty mode $q_i \in Q_{\mathcal{F}_s}$ or $q_i \in Q_{\mathcal{F}_{ns}}$ has a corresponding fault $f_i \in \mathcal{F}_s$ or $f_i \in \mathcal{F}_{ns}$ as well as a corresponding fault event defined in the set $\Sigma_{\mathcal{F}}$. The mode associated with structural faults have a dynamic model specifying their continuous behavior, whereas those associated with non-structural faults have not. [1]
- $\mathcal{G}$ defines a set of discrete-time state affine functions for each mode $q_i \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$:

  $$\mathbf{x}(k+1) = \mathbf{A}_i\mathbf{x}(k) + \mathbf{B}_i\mathbf{u}(k) + \mathbf{F}_{xi}\mathbf{f}_{ns}(k) + \mathbf{E}_{xi} \tag{3.2}$$

  where $\mathbf{A}_i \in \mathcal{R}^{n_x \times n_x}$, $\mathbf{B}_i \in \mathcal{R}^{n_x \times n_u}$ and $\mathbf{E}_{xi} \in \mathcal{R}^{n_x \times 1}$ are the state matrices in mode $q_i$, $\mathbf{f}_{ns}(k)$ is a vector representing non-structural faults with $\mathbf{F}_{xi}$ 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 $q_i \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$:

  $$\mathbf{y}(k) = \mathbf{C}_i\mathbf{x}(k) + \mathbf{D}_i\mathbf{u}(k) + \mathbf{F}_{y_i}\mathbf{f}_{ns}(k) + \mathbf{E}_{y_i} \tag{3.3}$$

  where $\mathbf{C}_i \in \mathcal{R}^{n_y \times n_x}$, $\mathbf{D}_i \in \mathcal{R}^{n_y \times n_u}$ and $\mathbf{E}_{y_i} \in \mathcal{R}^{n_y \times 1}$ 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 the set of events. Spontaneous mode switching events ($\Sigma_s$), input events ($\Sigma_c$) and fault events ($\Sigma_{\mathcal{F}} = \Sigma_{\mathcal{F}_s} \cup \Sigma_{\mathcal{F}_{ns}}$) are considered. $\Sigma$ can be partitioned into $\Sigma_o \cup \Sigma_{uo}$

[1] The dynamics of the system under a non-structural fault are assumed unknown. These faults are captured just by the modification of the system dynamics they imply. They are modeled by vector $\mathbf{f}_{ns}$.

where $\Sigma_o$ represents the set of observable events and $\Sigma_{uo}$ represents the set of unobservable events. $\Sigma_{\mathcal{F}} \subseteq \Sigma_{uo}$, $\Sigma_c \subseteq \Sigma_o$ and $\Sigma_s \subseteq \Sigma_{uo} \cup \subseteq \Sigma_o$ .

- $\mathcal{T} : \mathcal{Q} \times \Sigma \to \mathcal{Q}$ 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 $\tau_{ij}$ when the event is of no interest. [2].

Alternatively, the model given by equations (3.3) and (3.2) can be expressed in input-output form using the $p$-operator (or delay operator) considering zero initial conditions, as follows

$$\mathbf{y}(k) = \mathbf{M}_i(p^{-1})\mathbf{u}(k) + \mathbf{\Upsilon}_i(q^{-1})\mathbf{f}_{ns}(k) + \mathbf{E}_{mi}(p^{-1}) \tag{3.4}$$

where:

$$\mathbf{M}_i(p^{-1}) = \mathbf{C}_i(p\mathbf{I} - \mathbf{A}_i)^{-1}\mathbf{B}_i + \mathbf{D}_i \tag{3.5}$$

$$\mathbf{\Upsilon}_i(p^{-1}) = \mathbf{C}_i(p\mathbf{I} - \mathbf{A}_i)^{-1}\mathbf{F}_{xi} + \mathbf{F}_{y_i} \tag{3.6}$$

$$\mathbf{E}_{mi}(p^{-1}) = \left(\mathbf{C}_i(p\mathbf{I} - \mathbf{A}_i)^{-1}\mathbf{E}_{xi} + \mathbf{E}_{y_i}\right)\frac{p}{p-1} \tag{3.7}$$

where $\mathbf{M}_i(p^{-1})$ 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 terms $\mathbf{E}_{xi}$ and $\mathbf{E}_{y_i}$ in the state space model.

Table 3.1 summarizes when the transition function in $HA$ is possibly defined. The symbol '$-$' indicates that the transition between the corresponding two modes is not possible. Notice that transitions between nominal modes are possible in any sense, transitions from structural faulty modes to non-structural faulty modes are possible, transitions from faulty modes to nominal modes are not possible neither transitions leading from non-structural faulty modes.

| | | Destination modes | | |
|---|---|---|---|---|
| | | $\mathcal{Q}_{\mathcal{N}}(k)$ | $\mathcal{Q}_{\mathcal{F}_s}(k)$ | $\mathcal{Q}_{\mathcal{F}_{ns}}(k)$ |
| Source modes | $\mathcal{Q}_{\mathcal{N}}(k)$ | $\Sigma_s(k) \cup \Sigma_c(k)$ | $\Sigma_{\mathcal{F}_s}(k)$ | $\Sigma_{\mathcal{F}_{ns}}(k)$ |
| | $\mathcal{Q}_{\mathcal{F}_s}(k)$ | - | - | $\Sigma_{\mathcal{F}_{ns}}(k)$ |
| | $\mathcal{Q}_{\mathcal{F}_{ns}}(k)$ | - | - | - |

Table 3.1: Transition function defined for the $HA$

Another aspect to consider is that the composition of component automata is done for operation modes that belong to $\mathcal{Q}_{\mathcal{N}}(k) \cup \mathcal{Q}_{\mathcal{F}_s}(k)$, whose dynamical behavior is described by equations (3.2)-(3.3). Non-structural faulty modes are added a posteriori to the resulting hybrid automaton. Thus, the number of

[2] It is assumed that there is only one transition from a given mode $q_h$ to a given mode $q_l$.

non-structural modes associated with each mode in $\mathcal{Q}_{\mathcal{N}}(k) \cup \mathcal{Q}_{\mathcal{F}_s}(k)$ equals to $|\mathcal{F}_{ns}|$. This model results from an adaptation of [Lygeros et al., 2003] [Bayoudh et al., 2008, Bayoudh and Travé-Massuyès, 2012] [Vento et al., 2010].

*Remark* 3.1. Structural fault effects are represented through structural changes in state space matrices $\mathbf{A}_i$, $\mathbf{B}_i$, $\mathbf{E}_{xi}$,$\mathbf{C}_i$, $\mathbf{D}_i$ ,$\mathbf{E}_{y_i}$ in Eqs. (3.2)-(3.3).

*Remark* 3.2. Non-structural fault effects are represented as additive terms through matrices $\mathbf{F}_x$ and $\mathbf{F}_y$ in Eqs. (3.2)-(3.3). Moreover, $\mathbf{f}(k)$ is considered an additive signal.

## 3.3 Consistency indicators for hybrid systems

In the hybrid system framework, diagnosis is achieved both from reported observable events $\Sigma_o$ and continuous measurements $(\mathbf{y}(k), \mathbf{u}(k))$. Referring to the later, we adopt the common view of model-based diagnosis [Blanke et al., 2006] 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 a dynamic model of the form described by equations (3.2)-(3.3), then the set of residuals is given by:

$$\mathbf{r}_i(k) = \mathbf{y}(k) - \mathbf{G}_i(p^{-1})\mathbf{u}(k) - \mathbf{H}_i(p^{-1})\mathbf{y}(k) - \mathbf{E}_i(p^{-1}) \tag{3.8}$$

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 [Meseguer et al., 2010b], for instance. Alternatively, the parity space approach can be also used [Chow and Willsky, 1984]. In fact, the equivalence between the two approaches has been proved under certain conditions [Ding et al., 2008]. The observer model is given by:

$$\mathbf{G}_i(p^{-1}) = \mathbf{C}_i(p\mathbf{I} - \mathbf{A}_{oi})^{-1}\mathbf{B}_i + \mathbf{D}_i \tag{3.9}$$

$$\mathbf{H}_i(p^{-1}) = \mathbf{C}_i(p\mathbf{I} - \mathbf{A}_{oi})^{-1}\mathcal{L}_{oi} \tag{3.10}$$

$$\mathbf{E}_i(p^{-1}) = \left(\mathbf{C}_i(p\mathbf{I} - \mathbf{A}_{oi})^{-1}\mathbf{E}_{xi} + \mathbf{E}_{y_i}\right)\frac{p}{p-1} \tag{3.11}$$

where $\mathbf{A}_{oi} = \mathbf{A}_i - \mathcal{L}_{oi}\mathbf{C}_i$ and $\mathcal{L}_{oi}$ is the observer gain.

If $\mathcal{L}_{oi} = \mathbf{0} \rightarrow \mathbf{A}_{oi} = \mathbf{A}_i$ is a simulator, where:

$$\begin{aligned}
\mathbf{G}_i(p^{-1}) &= \mathbf{C}_i(q\mathbf{I}-\mathbf{A}_i)^{-1}\mathbf{B}_i+\mathbf{D}_i \\
\mathbf{H}_i(p^{-1}) &= \mathbf{0} \\
\mathbf{E}_i(p^{-1}) &= \mathbf{C}_i(q\mathbf{I}-\mathbf{A}_i)^{-1}\mathbf{E}_{x_i}\frac{p}{p-1}+\frac{p}{p-1}\mathbf{E}_{y_i}
\end{aligned}$$

If $\mathcal{L}_{oi}(\mathbf{C}_i)=\mathbf{A}_i \rightarrow \mathbf{A}_o^i=\mathbf{0}$ is a predictor, where:

$$\begin{aligned}
\mathbf{G}_i(p^{-1}) &= \mathbf{C}_i(p\mathbf{I})^{-1}\mathbf{B}_i+\mathbf{D}_i \\
\mathbf{H}_i(p^{-1}) &= \mathbf{C}_i(p\mathbf{I})^{-1}\mathcal{L}_{oi} \\
\mathbf{E}_i(p^{-1}) &= \mathbf{C}_i(p\mathbf{I})^{-1}\mathbf{E}_{x_i}\frac{p}{p-1}+\frac{p}{p-1}\mathbf{E}_{y_i}
\end{aligned}$$

Once the residuals have been generated, they are evaluated with the measurements against a threshold, providing one consistency indicator of the following form:

$$\phi_i^l(k)=\begin{cases} 0 & \text{if } |r_i^l(k)|\leq\tau_i^l \\ 1 & \text{if } |r_i^l(k)|>\tau_i^l \end{cases} \tag{3.12}$$

where $l\in\{1,\cdots,n_{r_i}\}$, $n_{r_i}$ is the number of residuals for mode $q_i$ and $\tau_i^l$ is the threshold associated with residual $r_i^l(k)$. Consistency indicators are then gathered in a vector $\mathbf{\Phi_i^j}(k)=[\phi_i^1(k),\cdots,\phi_i^{n_{r_i}}(k)]$. Summarizing, a consistency indicator vector $\mathbf{\Phi}_i^j(k)$ is built from the binarised residual expression given by (3.12) of mode $i$ evaluated with the measurements consistent with mode $j$.

To detect and isolate non-structural faults, a theoretical fault signature matrix in mode $q_i$ is generated using the concept of fault sensitivity, which is determined by the expression:

$$\mathbf{\Lambda}_i(p^{-1}) = (\mathbf{I}-\mathbf{H}_i(p^{-1}))\mathbf{\Upsilon}_i(p^{-1}) \tag{3.13}$$

where $\mathbf{\Upsilon}_i(p^{-1})$ is described by equation (3.6). In particular, 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 $\mathbf{\Lambda}_i(p^{-1})$), the element $(j,l)$ of $\mathbf{FS}_i$ is determined as follows:

$$\mathbf{FS}_i(j,l)=\begin{cases} 1 & \text{if } \mathbf{\Lambda}_i(j,l)\neq 0 \\ 0 & \text{if } \mathbf{\Lambda}_i(j,l)=0 \end{cases} \tag{3.14}$$

i.e., if the $j^{th}$ residual in mode $q_i$ depends on the $l^{th}$ fault, it is coded as a 1 otherwise as a 0. For completeness one more column with zero signature is added representing the non-structural fault free case. If $f_j$ is the $l^{th}$ non-structural fault, the theoretical fault signature of $f_j$, denoted as $\mathbf{FS}_i^{f_j}$, is then given by $\mathbf{FS}_i(\bullet, l)$.

## 3.4 Discernibility properties

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., 2004], 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.

**Definition 3.1.** 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 viceversa.

From the properties of residuals, we have the following result:

**Proposition 3.4.1.** *Two modes $q_i$ and $q_j$ are non discernible iff the consistency indicators of the two modes satisfy $\Phi_i(k) = 0$ and $\Phi_j(k) = 0$ for any $(\boldsymbol{u}(k), \boldsymbol{y}(k))$ and any time instant $k$.*

We define the following function:

$$f_{disc} : \mathcal{Q} \times \mathcal{Q} \rightarrow \{0, 1\} \tag{3.15}$$

where 1 means that the modes are discernible and 0 that they are not. The discernibility function evaluation depends on the class of modes considered in $HA$. Three possible cases will be analyzed further on. Discernibility can be defined as follows:

**Definition 3.2.** Two modes $q_i, q_j$ are non-discernible if and only if $f_{disc}(q_i, q_j) = 0$.

The following definitions are related to discernibility.

**Definition 3.3.** A mode change from mode $q_i$ to mode $q_j$ in $HA$, is detectable at time instant $k$ if both modes are discernible according to function (3.15).

**Definition 3.4.** Two mode changes, $q_i \rightarrow q_j$ and $q_i \rightarrow q_l$ in $HA$, are isolable if the following conditions are satisfied at time instant $k$:

1. Both mode changes are detectable according to Definition 3.3.

2. In the case that some of these mode changes $q_i \rightarrow q_j$ or $q_i \rightarrow q_l$ are detected, the pair of modes $(q_l, q_j)$ is discernible according to function (3.15).

The conditions to evaluate the discernibility function depend on the pair of modes considered in $HA$. Fig. 3.3 summarizes all possible situations where the discernibility property should be analyzed. Moreover, modes $q_a, q_b \in \mathcal{Q}_{\mathcal{N}}$, $q_c \in \mathcal{Q}_{\mathcal{F}_s}$ and $q_{a_1}, q_{a_2} \in \mathcal{Q}_{\mathcal{F}_{ns}}$.

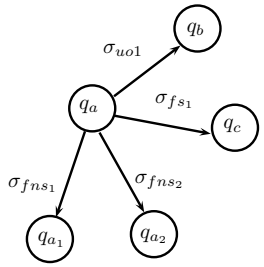


Figure 3.3: Illustration example

The discernibility study comprises three possible cases and the mathematical properties are only evaluated for pair of modes as can be seen in function (3.15).

### 3.4.1 Case 1

Let us consider a pair of modes that have an associated continuous dynamic model of the form (3.2)-(3.3), represented in input-output form (3.4). In Fig. 3.3, it can be seen that the discernibility property must be studied between the pair of modes $(q_a, q_b)$ and $(q_a, q_c)$ to predict if a mode change can be detected and between the pair of modes $(q_c, q_b)$ to know if both mode changes can be isolated. Hence, the following proposition is given:

**Proposition 3.4.2.** *Two modes $\{q_i, q_j\} \subseteq \mathcal{Q}^k_{\mathcal{N}} \cup \mathcal{Q}^k_{\mathcal{F}_s}$ are non-discernable if the following conditions are fulfilled:*

$$\boldsymbol{M}_i(p^{-1}) = \boldsymbol{M}_j(p^{-1}) \tag{3.16}$$

$$\boldsymbol{E}_{mi}(p^{-1}) = \boldsymbol{E}_{mj}(p^{-1}) \tag{3.17}$$

*where $\boldsymbol{M}_i(p^{-1})$, $\boldsymbol{E}_{mi}(p^{-1})$, $\boldsymbol{M}_j(p^{-1})$ and $\boldsymbol{E}_{mj}(p^{-1})$ correspond to input-output model matrices i.e, they guarantee that consistency indicators satisfy $\Phi_i(k) = 0$ and $\Phi_j(k) = 0$ at any time instant.*

The discernibility function can be verified using conditions (3.16)-(3.17), which are derived from the system model equations (3.2)-(3.3) represented in input-output form (3.4) and residual expression (3.8).

**Proof 3.4.1.** *For mode $i$, the residual expression is given by:*

$$\boldsymbol{r}_i(k) = (\boldsymbol{I} - \boldsymbol{H}_i(p^{-1}))\boldsymbol{y}(k) - \boldsymbol{G}_i(p^{-1})\boldsymbol{u}(k) - \boldsymbol{E}_i(p^{-1}) \tag{3.18}$$

*Under no fault*[3] *condition, $\boldsymbol{r}_i(k) = 0$ as long as measurements $\boldsymbol{u}(k), \boldsymbol{y}(k)$ are consistent with mode $i$. Therefore, the following equation holds:*

$$\boldsymbol{y}(k) = \boldsymbol{M}_i(p^{-1})\boldsymbol{u}(k) + \boldsymbol{E}_{mi}(p^{-1}) \tag{3.19}$$

*For mode $j$, the residual expression is given by:*

$$\boldsymbol{r}_j(k) = (\boldsymbol{I} - \boldsymbol{H}_j(p^{-1}))\boldsymbol{y}(k) - \boldsymbol{G}_j(p^{-1})\boldsymbol{u}(k) - \boldsymbol{E}_j(p^{-1}) \tag{3.20}$$

*Replacing equation (3.19) into equation (3.20) leads to:*

$$\boldsymbol{r}_{j/i}(k) = ((\boldsymbol{I} - \boldsymbol{H}_j(p^{-1}))\boldsymbol{M}_i(p^{-1}) - \boldsymbol{G}_j(p^{-1}))\boldsymbol{u}(k) + (\boldsymbol{I} - \boldsymbol{H}_j(p^{-1}))\boldsymbol{E}_{mi}(p^{-1}) - \boldsymbol{E}_j(p^{-1}) \tag{3.21}$$

*which corresponds to the residual expression of mode $j$ evaluated with measurements corresponding to mode $i$. Therefore, the following equalities must be satisfied in order to have a zero residual:*

$$\left(\boldsymbol{I} - \boldsymbol{H}_j(p^{-1})\right)\boldsymbol{M}_i(p^{-1}) = \boldsymbol{G}_j(p^{-1}) \tag{3.22}$$

$$\left(\boldsymbol{I} - \boldsymbol{H}_j(p^{-1})\right)\boldsymbol{E}_{mi}(p^{-1}) = \boldsymbol{E}_j(p^{-1}) \tag{3.23}$$

*Similarly concerning from measurements corresponding to mode $j$, the residual expression of mode $i$ leads to:*

$$\boldsymbol{r}_{i/j}(k) = ((\boldsymbol{I} - \boldsymbol{H}_i(p^{-1}))\boldsymbol{M}_j(p^{-1}) - \boldsymbol{G}_i(p^{-1}))\boldsymbol{u}(k) + (\boldsymbol{I} - \boldsymbol{H}_i(p^{-1}))\boldsymbol{E}_{mj}(p^{-1}) - \boldsymbol{E}_i(p^{-1}) \tag{3.24}$$

*where the following equalities must be satisfied in order to have a zero residual:*

[3] Without the effect of a non-structural fault.

$$\left(\boldsymbol{I} - \boldsymbol{H}_i(p^{-1})\right)\boldsymbol{M}_j(p^{-1}) = \boldsymbol{G}_i(p^{-1}) \quad (3.25)$$

$$\left(\boldsymbol{I} - \boldsymbol{H}_i(p^{-1})\right)\boldsymbol{E}_{mj}(p^{-1}) = \boldsymbol{E}_i(p^{-1}) \quad (3.26)$$

*Therefore, in order to satisfy equalities (3.22), (3.23), (3.25) and (3.26) at the same time the following conditions must also be satisfied:* $\boldsymbol{M}_i(p^{-1}) = \boldsymbol{M}_j(p^{-1})$ *and* $\boldsymbol{E}_{mi}(p^{-1}) = \boldsymbol{E}_{mj}(p^{-1})$.

### 3.4.2 Case 2

Let us consider a pair of modes corresponding to non-structural faults, that have a common predecessor mode. This mode do not have a continuous dynamic model but faults have a signature in the fault signature matrix. According to Fig. 3.3, the discernibility property should be studied between the pair of modes $(q_a, q_{a_1})$ and $(q_a, q_{a_2})$ to predict if the mode change can be detected and between the pair of modes $(q_{a_1}, q_{a_2})$ to know if they can be isolated. Notice that every non-structural faulty mode is associated with a non-structural fault and a signature in the fault signature matrix.

The discernibility property involves comparing the fault signatures between them.

**Proposition 3.4.3.** *Two modes* $\{q_{i_1}, q_{i_2}\} \subseteq \mathcal{Q}^k_{\mathcal{F}_{ns}}$ *associated to non-structural faults* $f_{ns_1}$ *and* $f_{ns_2}$ *respectively, such that* $\mathcal{T}^k(q_i, \sigma_{f_{ns_1}}) = q_{i_1}$ *and* $\mathcal{T}^k(q_i, \sigma_{f_{ns_2}}) = q_{i_2}$ *for a given mode* $q_i \in \mathcal{Q}^k_{\mathcal{N}} \cup \mathcal{Q}^k_{\mathcal{F}_s}$ *and* $\sigma_{f_{ns_1}}, \sigma_{f_{ns_2}} \in \Sigma^k_{\mathcal{F}_{ns}}$*, are non-discernable if their residual fault sensitivities satisfy:*

$$\Lambda_i^{f_l}(p^{-1}) = \Lambda_i^{f_j}(p^{-1}) \neq \boldsymbol{0} \quad (3.27)$$

*where* $i$ *corresponds to the mode* $q_i \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$*,*

**Proof 3.4.2.** *The sensitivity to a non-structural fault applying expression (3.14) corresponds to a binary signature, therefore for this two modes the signatures are* $\boldsymbol{FS}_i^{f_l}$ *and* $\boldsymbol{FS}_i^{f_j}$ *respectively. If their fault sensitivities are equivalent then their signatures are equivalent too. Then, a pair of modes are non-discernible if the following condition holds:*

$$\boldsymbol{FS}_i^{f_l} \neq \boldsymbol{FS}_i^{f_j} \neq \boldsymbol{0} \quad (3.28)$$

Notice that the residual sensitivities in the non-structural fault free case are also included in the fault signature matrix as an additional 0 column vector. Thus, Proposition 3.4.3 can be applied to determine both, non-structural fault detectability and isolability.

### 3.4.3 Case 3

Let us consider a mode that has a continuous dynamic model and another one which has not continuous dynamic model, with a common predecessor mode. In Fig 3.3, this corresponds to the study of the discernibility property between the following pairs of modes $(q_b, q_{a_1})$, $(q_b, q_{a_2})$, $(q_c, q_{a_1})$ or $(q_c, q_{a_2})$.

**Proposition 3.4.4.** *A mode $q_j \in \mathcal{Q}^k_{\mathcal{N}} \cup \mathcal{Q}^k_{\mathcal{F}_s}$ and a mode $q_{i_\alpha} \in \mathcal{Q}^k_{\mathcal{F}_{ns}}$ associated with non-structural fault $f_{ns_\alpha}$, such that $\mathcal{T}^k(q_i, \sigma) = q_j$ and $\mathcal{T}^k(q_i, \sigma_{f_{ns_\alpha}}) = q_{i_\alpha}$ for a given mode $q_i \in \mathcal{Q}^k_{\mathcal{N}} \cup \mathcal{Q}^k_{\mathcal{F}_s}$, $\sigma \in \Sigma^k_s \cup \Sigma^k_c \cup \Sigma^k_{\mathcal{F}_s}$ and $\sigma \in \Sigma^k_{\mathcal{F}_{ns}}$, are non-discernible if the following conditions are fulfilled:*

$$\begin{aligned} \boldsymbol{M}_j(p^{-1}) - \boldsymbol{M}_i(p^{-1}) &= \Lambda_i^{f_{ns_\alpha}}(p^{-1}) && (3.29)\\ \boldsymbol{E}_{mi}(p^{-1}) &= \boldsymbol{E}_{mj}(p^{-1}) && (3.30)\\ \boldsymbol{u}(k) &= \boldsymbol{f}_{ns_\alpha}(k) && (3.31) \end{aligned}$$

Notice that in this case for the non-structural faulty mode, the sensitivity function is calculated through dynamic model of its predecessor mode in order to evaluate the discernibility condition.

**Proof 3.4.3.** *This case can be deduced from case 1 and case 2 respectively. Consider the following residual expressions:*

$$\begin{aligned} \boldsymbol{r}_{i/j}(k) = \quad & ((\boldsymbol{I} - \boldsymbol{H}_i(p^{-1}))\boldsymbol{M}_j(p^{-1}) - \boldsymbol{G}_i(p^{-1}))\boldsymbol{u}(k) \\ & +(\boldsymbol{I} - \boldsymbol{H}_i(p^{-1}))\boldsymbol{E}_{mj}(p^{-1}) - \boldsymbol{E}_i(p^{-1}) \end{aligned} \tag{3.32}$$

*that corresponds to the residual of mode $q_i$ evaluated with measurements corresponding to mode $q_l$ and*

$$\begin{aligned} \boldsymbol{r}_{i/i_\alpha}(k) = & ((\boldsymbol{I} - \boldsymbol{H}_i(p^{-1}))\boldsymbol{M}_i(p^{-1}) - \boldsymbol{G}_i(p^{-1}))\boldsymbol{u}(k) \\ & -\boldsymbol{E}_i(p^{-1}) + (\boldsymbol{I} - \boldsymbol{H}_i(p^{-1}))\boldsymbol{E}_{mi}(p^{-1}) \\ & +(\boldsymbol{I} - \boldsymbol{H}_i(p^{-1}))\boldsymbol{\Upsilon}_i(p^{-1})\boldsymbol{f}_{ns_\alpha}(k) \end{aligned} \tag{3.33}$$

*that corresponds to the residual expression of mode $q_i$ evaluated with measurement corresponding to mode $q_j$ under the non-structural fault effect. Evaluating the difference $\boldsymbol{r}_{i/i_\alpha}(k) - \boldsymbol{r}_{i/j}(k)$, we obtain:*

$$\begin{gathered} (\boldsymbol{I} - \boldsymbol{H}_i(p^{-1}))\boldsymbol{M}_i(p^{-1})\boldsymbol{u}(k) + (\boldsymbol{I} - \boldsymbol{H}_i(p^{-1}))\boldsymbol{E}_{mi}(p^{-1}) \\ (\boldsymbol{I} - \boldsymbol{H}_i(p^{-1}))\boldsymbol{\Upsilon}_i(p^{-1})\boldsymbol{f}_{ns_\alpha}(k) = \\ (\boldsymbol{I} - \boldsymbol{H}_i(p^{-1}))\boldsymbol{M}_j(p^{-1})\boldsymbol{u}(k) + (\boldsymbol{I} - \boldsymbol{H}_i(p^{-1}))\boldsymbol{E}_{mj}(p^{-1}) \end{gathered}$$

*Hence, the pair of modes is non-discernible if the following conditions are satisfied:*

$$\begin{aligned} \boldsymbol{M}_j(p^{-1}) - \boldsymbol{M}_i(p^{-1}) &= \Lambda_i^{f_{ns_\alpha}}(p^{-1}) \\ \boldsymbol{E}_{mj}(p^{-1}) &= \boldsymbol{E}_{mi}(p^{-1}) \end{aligned}$$

*assuming that* $\boldsymbol{u}(k)$ *and* $\boldsymbol{f}_{ns_\alpha}(k)$ *are unitary steps.*

## 3.5 Behavior automaton

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. The behavior automaton is defined by $B =< \overline{\mathcal{Q}}, \overline{\Sigma}, \overline{\mathcal{T}}, \overline{q}_0 >$ where:

- $\overline{\mathcal{Q}} = \mathcal{Q} \cup \mathcal{Q}^t$ is a set of discrete states where:
  - $\mathcal{Q}$ is a set of system modes,
  - $\mathcal{Q}^t$ 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:
  - $\Sigma$ is a set of system events,
  - $\Sigma^{Sig}$ is a set of signature-events generated when two modes are discernible according to function (3.15).
- $\overline{\mathcal{T}} : \overline{\mathcal{Q}} \times \overline{\Sigma} \mapsto \overline{\mathcal{Q}}$ is a partial transition function of the behavior automaton.

The transition function is built following Algorithm 3.1. Unobservable events of the $HA$ may become observable depending on the discernibility properties.

Through the discernibility property, the set of system modes can be partitioned into subsets of non discernible-modes, i.e. $\mathcal{Q}_{disc} = \mathcal{Q}_{\nu_1} \cup \cdots \cup \mathcal{Q}_{\nu_N}$. This information is stored in a knowledge-base used by Algorithm 3.1.

## 3.6 Behavior automaton building

$B$ is built following Algorithm 3.1. In particular, it is shown that the transition function in $B$ is built based on discernibility properties presented in Sections 3.6. The discernibility property is evaluated whenever necessary (see Section 3.6). If a transition in $HA$ involves an observable event, the transition is kept in $B$ (see line 15). Otherwise, the discernibility property is evaluated between the pair of modes. If the two modes are discernible, then a transient mode [4] is added between these modes. The outgoing transition is associated with a signature-event $\delta$, indicating that this mode change can be detected by means of

**Algorithm 3.1** B_Builder()

1: Create a queue $\mathcal{L}$.
2: **for all** $q_i \in \mathcal{Q}$ **do**
3: Enqueue $q_i$ onto $\mathcal{L}$
4: **end for**
5: **while** $\mathcal{L}$ is not empty **do**
6: $q_i$:=dequeue $\mathcal{L}$
7: **for all** $q_j \in Succs_{HA}(q_i)$ **do**
8: **if** $q_j \notin \mathcal{Q} \cap \overline{\mathcal{Q}}$ **then**
9: $\overline{\mathcal{Q}} = \{q_j\} \cup \overline{\mathcal{Q}}$
10: Classify $q_j$ into $\mathcal{Q}_{disc}$.
11: **if** $q_j$ creates a new group $\nu_j$ in $\mathcal{Q}_{disc}$ **then**
12: Compute $\mathbf{FS}_{\nu_j}(\bullet)$.
13: Determine the subsets of detectable faults $\mathcal{F}^*_{\nu_j}$.
14: Determine the set of non-detectable faults $\mathcal{F}^{'}_{\nu_j}$.
15: Update and store in knowledge-base.
16: **end if**
17: **end if**
18: Let $\sigma$ is such as $\mathcal{T}(q_i, \sigma) = q_j$ :
19: **switch** ($\sigma$)
20: **case** $\sigma \in \Sigma_o$**:**
21: $\overline{\mathcal{T}}(q_i, \sigma) := q_j$.
22: **case** $\sigma \in \Sigma_{uo}$**:**
23: **if** $q_i$ and $q_j$ are discernible according to function (3.15) **then**
24: $\mathcal{Q}^t = \{q^t_{i-j}\} \cup \mathcal{Q}^t$.
25: $\delta := f_{Sig\_ev}(q_i, q_j)$ according to function (3.34).
26: **if** $\delta \notin \overline{\Sigma}$ **then**
27: $\overline{\Sigma} = \{\delta\} \cup \overline{\Sigma}$
28: **end if**
29: $\overline{\mathcal{T}}(q_i, \sigma) := q^t_{i-j}$.
30: $\overline{\mathcal{T}}(q^t_{i-j}, \delta) := q_j$.
31: **else**
32: **if** $q_j \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$ **then**
33: Enqueue $q_j$ onto $\mathcal{L}$
34: **end if**
35: $\overline{\mathcal{T}}(q_i, \sigma) := q_j$.
36: **end if**
37: **end switch**
38: **end for**
39: **evaluate_discernability_between_successors**().
40: **end while**

consistency indicators (see lines 17-24). Finally, if the pair of modes $(q_i, q_j)$ is non-discernible, then the transition is kept in $B$ (see line 29).

Given a pair of modes $(q_i, q_j)$, signature-event $\delta$ is properly labeled according the following function:

$$\delta = f_{Sig\_ev} : \overline{\mathcal{Q}} \times \overline{\mathcal{Q}} \rightarrow \Sigma^{Sig} \tag{3.34}$$

[4]The transient mode is the way to account for the hybrid automaton $HA$ dwell time requirement [Bayoudh et al., 2009].

$$f_{Sig\_ev} \mapsto \begin{cases} \delta_{\nu_i-\nu_j} & \text{if } f_{disc}(q_i,q_j)=1 \text{ according to Proposition 3.4.1, where } \delta_{\nu_i-\nu_j} \text{ represents} \\ & \text{that } q_i \in \mathcal{Q}_{\nu_i} \text{ and } q_j \in \mathcal{Q}_{\nu_j} \text{ whith } \mathcal{Q}_{\nu_i}, \mathcal{Q}_{\nu_j} \subseteq \mathcal{Q}_{disc} \\ \delta_{\mathcal{F}^{\iota}_{\nu_i}} & \text{if } f_{disc}(q_i,q_j)=1 \text{ according to Proposition 3.4.2 } \delta_{\mathcal{F}^{\iota}_{\nu}} \text{ is associated to} \\ & \text{a non-structural fault } f_l \text{ belonging to a subset } \mathcal{F}^{\iota}_{\nu_i} \text{ with } \iota \in \mathcal{Z}^+ \\ \delta & \text{if } f_{disc}(q_i,q_j)=1 \text{ according to Proposition 3.4.3 associated with case 3} \end{cases}$$

The event label allow one distinguishing between the discernability cases analyzed in Section , so that the diagnoser can be properly built.

## 3.7 Diagnoser automaton building

$B$ is used to build the diagnoser automaton. The diagnoser automaton is a finite state machine $D =< \mathcal{Q}_D, \Sigma_D, T_D, q_{D_0} >$, where:

- $q_{D_0} = \{q_0, \emptyset\}$ is the initial state of the diagnoser, which is assumed to correspond to a nominal system mode.
- $\mathcal{Q}_D$ is a set of the diagnoser states. An element $q_D \in \mathcal{Q}_D$ is a set of the form $q_D = \{(q_1,l_1),(q_2,l_2),\cdots(q_n,l_n)\}$, where $q_i \in \overline{\mathcal{Q}}$ and $l_i \in \Delta$ where $\Delta$ defines the power set of fault labels $\Delta_{\mathcal{F}} = \Delta_{\mathcal{F}_s} \cup \Delta_{\mathcal{F}_{ns}}$ with $\Delta_{\mathcal{F}_s} = \{f_1,\cdots,f_\gamma\}$, and $\Delta_{\mathcal{F}_{ns}} = \{f_1^*,\cdots,f_\mu^*\}$ respectively, $\gamma+\mu$ is the total number of faults in the system and $\gamma,\mu \in \mathcal{Z}^+$. In $\Delta_{\mathcal{F}}$, $\emptyset$ represents the nominal behavior,
- $\Sigma_D = \overline{\Sigma}_o$ is the set of all observable events in $B$,
- $\mathcal{T}_D : \mathcal{Q}_D \times \overline{\Sigma}_o \mapsto \mathcal{Q}_D$ is a partial transition function of the diagnoser.

The diagnoser is adapted to include the system modes generated for the entire $HA$ representing the system behavior and the interaction of the system components. Transition function $\mathcal{T}_D$ is calculated according to the propagation algorithms without silent-closured explained in [Sampath et al., 1995]. This propagation is more appropriated when the initial mode of the system is assumed as known.

## 3.8 Mode tracking logic

Given a set of observations of the system, a mode change can be detected whenever the consistency indicators corresponding to the current mode change. The minimal time to detect that change is the

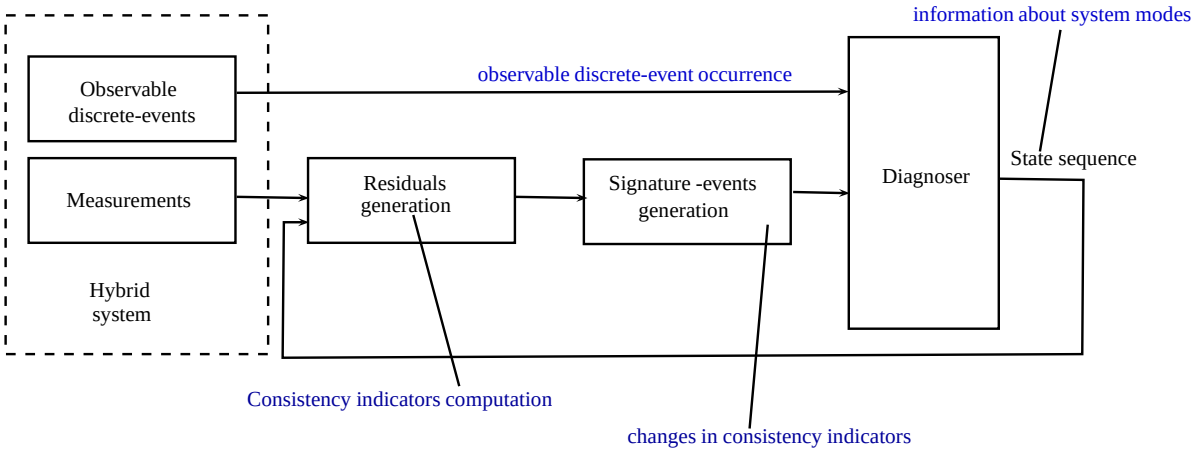


Figure 3.4: Implementation scheme for hybrid systems diagnosis

dwell time requirement (see Assumption 3.2 ), which guarantees that residuals, and hence consistency indicators, can be properly computed. The online diagnosis scheme implementation is shown in Fig. 3.4.

The following results provide conditions for transition detection and transition identification.

**Proposition 3.8.1.** *if $\Phi_i(k-1) = 0$ and $\Phi_i(k) \neq 0$, then a transition from $q_i \in \mathcal{Q}_{\mathcal{N}}(k) \cup \mathcal{Q}_{\mathcal{F}_s}(k)$ to another mode is detected at time instant $k$.*

Proposition 3.8.1 is used to decide if a mode change in the system has occurred by monitoring the set of consistency indicators of the feasible current modes.

**Proposition 3.8.2.** *Assuming that $HA$ is in mode $q_i$ and a transition has been detected at time instant $k$ according to Proposition 3.8.1 and under some discernibility assumptions (see Section 3.6):*

1. *if $\Phi_i(k) = \boldsymbol{FS}_i(\bullet, f_j)$ then a transition to $q_j \in \mathcal{Q}_{\mathcal{F}_{ns}}(k)$ is detected at time instant $k$.*

2. *if $\Phi_j(k) = 0$ then a transition to $q_j \in \mathcal{Q}_{\mathcal{N}}(k) \cup \mathcal{Q}_{\mathcal{F}_s}(k)$ is detected at time instant $k$.*

Notice that Proposition 3.8.2 does not necessarily identify a unique mode $q_j$. In particular, condition 1) or 2) of Proposition 3.8.2 may be satisfied for more than one index, which respectively corresponds to a case of ambiguous non-structural faulty modes and a case of ambiguous structural faulty modes. This logic is used to identify the set of possible events through Algorithm 3.2. A diagnoser state represents the set of modes the system can be possibly operating in. Therefore, the consistency indicator is denoted by $\Phi_{q_{D_i}}(k)$ in the algorithm to indicate the possible ambiguity that may exist in the modes of $HA$.

**Algorithm 3.2** Event_Processing($q_D$)

1: **loop**
2: **wait until** $\Phi_{q_{Di}}(k) \neq 0$ **or** $\sigma_o \in \Sigma_o$ occurs
3: **if** $\sigma_o$ occurs **then**
4: $\sigma_D := \sigma_o$
5: **else**
6: **for all** $q_{Dj} \in Succs(q_{Di})$ **do**
7: **if** $\Phi_{q_{Dj}}(k) =$ **0 then**
8: $COND1 := true$
9: `break`
10: **end if**
11: **end for**
12: **for all** $q_{Dj} \in Succs(q_{Di})$ **do**
13: **if** $\Phi_{q_{D_i}}(k) =$ **FS**$_{\nu_i}(\bullet, \mathcal{F}^{\iota}_{\nu_i})$ **then**
14: $COND2 := true$
15: `break`
16: **end if**
17: **end for**
18: **if** $COND1 = false$ **and** $COND2 = false$ **then**
19: **print** `Unknown event`
20: **else**
21: **if** $COND1$ and $COND2$ **then**
22: $\sigma_D := \delta$
23: **else**
24: **if** $COND1$ **then**
25: $\sigma_D := \delta_{\nu_i - \nu_j}$
26: **else**
27: $\sigma_D := \delta_{\mathcal{F}^{\iota}_{\nu_i}}$
28: **end if**
29: **end if**
30: **end if**
31: **return**
32: **end if**
33: **end loop**

## 3.9 Two-tanks sewer network example

To illustrate the methodology introduced in this chapter, consider here a small part of the sewer network described in more detail in Chapter 5. The elements that appear in Fig. 3.5 are: two virtual tanks ($T_1$ and $T_2$), two limnimeters to measure the sewer levels ($L_{39}$ and $L_{41}$), two rain gauges to measure the input rain intensity in virtual tanks ($P_{19}$ and $P_{16}$), and one redirection gate ($G_1$) placed downstream $T_1$, which allows to change the flow direction.

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 close). 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 3.2.

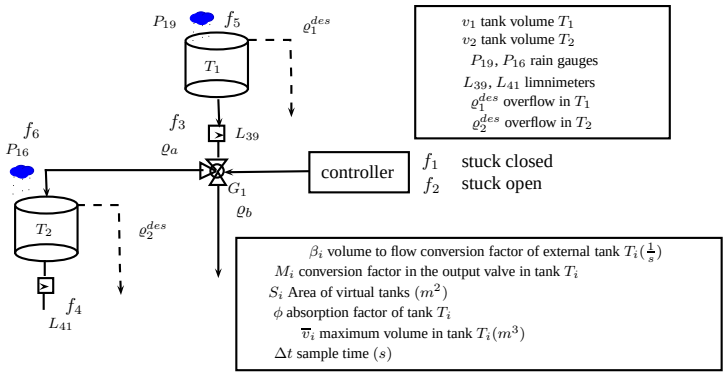


Figure 3.5: Small part of the sewer network

| Event | action | Observable | type | code |
|---|---|---|---|---|
| $uo1$ | $v_1 \geq \overline{v}_2$ | not | spontaneous | 1 |
| $uo2$ | $\varrho_1^{in} < \varrho_1^{out}$ | not | spontaneous | 2 |
| $uo3$ | $v_2 \geq \overline{v}_2$ | 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 |

Table 3.2: Type of events in $HA$

### 3.9.1 Hybrid automaton for two-tanks sewer network

The entire hybrid automaton can be obtained from parallel composition of the component automata. Hence, the global hybrid automaton is given in Fig. 3.6.

As it can be seen, the hybrid automaton is composed by eight nominal modes and eight faulty modes (related to structural faults). Labels for modes belonging to $\mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$ are shown in Table 3.3. Mode $q_1$ means that neither tank is in overflow and $G_1$ is open (nominal mode), mode $q_9$ means that neither tank is in overflow and $G_1$ is stuck open (structural faulty mode), mode $q_{17}$ is similar to $q_1$ but affected by a

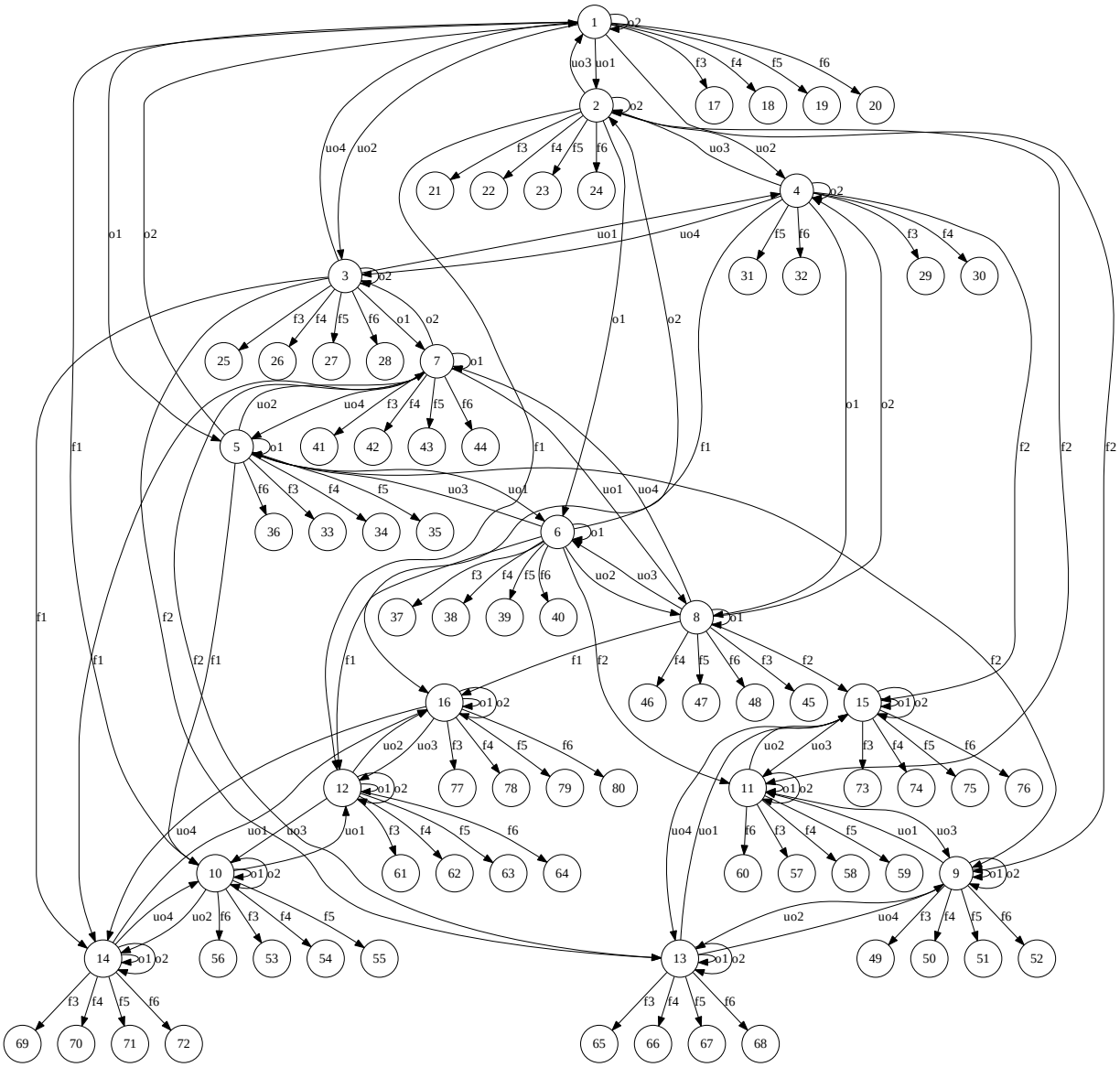


Figure 3.6: Hybrid automaton model obtained using the component automata composition

non-structural fault in sensor $P_{19}$, and so on.

Table 3.4 shows labels for modes belonging to $\mathcal{Q}_{\mathcal{F}_{ns}}$. In total, there are 64 modes representing faults in input and output sensors.

The continuous dynamical model for each mode $q_i \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$ is provided in Table 3.5. 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

| Label | Mode | Label | Mode |
|---|---|---|---|
| $T1wo.open.T2wo$ | $q_1$ | $T1wo.closed.T2wo$ | $q_5$ |
| $T1wo.open.T2o$ | $q_2$ | $T1wo.closed.T2o$ | $q_6$ |
| $T1o.open.T2wo$ | $q_3$ | $T1o.closed.T2wo$ | $q_7$ |
| $T1o.open.T2o$ | $q_4$ | $T1o.closed.T1o$ | $q_8$ |
| $T1wo.So.T2wo$ | $q_9$ | $T1wo.Sc.T2wo$ | $q_{10}$ |
| $T1wo.So.T2o$ | $q_{11}$ | $T1wo.Sc.T2o$ | $q_{12}$ |
| $T1o.So.T2wo$ | $q_{13}$ | $T1o.Sc.T2wo$ | $q_{14}$ |
| $T1o.So.T2o$ | $q_{15}$ | $T1o.Sc.T2o$ | $q_{16}$ |

Table 3.3: Mode labels belonging to $\mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$

| Label | Mode | Label | Mode | Label | Mode | Label | Mode |
|---|---|---|---|---|---|---|---|
| $T1wo.open.T2wo.FP_{19}$ | $q_{17}$ | $T1wo.open.T2wo.FP_{16}$ | $q_{18}$ | $T1wo.open.T2wo.FL_{39}$ | $q_{19}$ | $T1wo.open.T2wo.FL_{47}$ | $q_{20}$ |
| $T1o.open.T2wo.FP_{19}$ | $q_{21}$ | $T1o.open.T2wo.FP_{16}$ | $q_{22}$ | $T1o.open.T2wo.FL_{39}$ | $q_{23}$ | $TT1o.open.T2wo.FL_{47}$ | $q_{24}$ |
| $T1wo.open.T2o.FP_{19}$ | $q_{25}$ | $T1wo.open.T2o.FP_{16}$ | $q_{26}$ | $T1wo.open.T2o.FL_{39}$ | $q_{27}$ | $T1wo.open.T2o.FL_{47}$ | $q_{28}$ |
| $T1o.open.T2o.FP_{19}$ | $q_{29}$ | $T1o.open.T2o.FP_{16}$ | $q_{30}$ | $T1o.open.T2o.FL_{39}$ | $q_{31}$ | $T1o.open.T2o.FL_{47}$ | $q_{32}$ |
| $T1wo.closed.T2wo.FP_{19}$ | $q_{33}$ | $T1wo.closed.T2wo.FP_{16}$ | $q_{34}$ | $T1wo.closed.T2wo.FL_{39}$ | $q_{35}$ | $T1wo.closed.T2wo.FL_{47}$ | $q_{36}$ |
| $T1o.closed.T2wo.FP_{19}$ | $q_{37}$ | $T1o.closed.T2wo.FP_{16}$ | $q_{38}$ | $T1o.closed.T2wo.FL_{39}$ | $q_{39}$ | $T1o.closed.T2wo.FL_{47}$ | $q_{40}$ |
| $T1wo.closed.T2o.FP_{19}$ | $q_{41}$ | $T1wo.closed.T2o.FP_{16}$ | $q_{42}$ | $T1wo.closed.T2o.FL_{39}$ | $q_{43}$ | $T1wo.closed.T2o.FL_{47}$ | $q_{44}$ |
| $T1o.open.T2o.FP_{19}$ | $q_{45}$ | $T1o.open.T2o.FP_{16}$ | $q_{46}$ | $T1o.open.T2o.FL_{39}$ | $q_{47}$ | $T1o.open.T2o.FL_{47}$ | $q_{48}$ |
| $T1wo.So.T2wo.FP_{19}$ | $q_{49}$ | $T1wo.So.T2wo.FP_{16}$ | $q_{50}$ | $T1wo.So.T2wo.FL_{39}$ | $q_{51}$ | $T1wo.So.T2wo.FL_{47}$ | $q_{52}$ |
| $T1o.So.T2wo.FP_{19}$ | $q_{53}$ | $T1o.So.T2wo.FP_{16}$ | $q_{54}$ | $T1o.So.T2wo.FP_{19}.FL_{39}$ | $q_{55}$ | $T1o.So.T2wo.FP_{19}.FL_{47}$ | $q_{56}$ |
| $T1wo.So.T2o.FP_{19}$ | $q_{57}$ | $T1wo.So.T2o.FP_{16}$ | $q_{58}$ | $T1wo.So.T2o.FL_{39}$ | $q_{59}$ | $T1wo.So.T2o.FL_{47}$ | $q_{60}$ |
| $T1o.open.T2o.FP_{19}$ | $q_{61}$ | $T1o.open.T2o.FP_{16}$ | $q_{62}$ | $T1o.open.T2o.FL_{39}$ | $q_{63}$ | $T1o.open.T2o.FL_{47}$ | $q_{64}$ |
| $T1wo.Sc.T2wo.FP_{19}$ | $q_{65}$ | $T1wo.Sc.T2wo.FP_{16}$ | $q_{66}$ | $T1wo.Sc.T2wo.FL_{39}$ | $q_{67}$ | $T1wo.Sc.T2wo.FL_{47}$ | $q_{68}$ |
| $T1o.Sc.T2wo.FP_{19}$ | $q_{69}$ | $T1o.Sc.T2wo.FP_{16}$ | $q_{70}$ | $T1o.Sc.T2wo.FL_{39}$ | $q_{71}$ | $T1o.Sc.T2wo.FL_{47}$ | $q_{72}$ |
| $T1wo.Sc.T2o.FP_{19}$ | $q_{73}$ | $T1wo.Sc.T2o.FP_{16}$ | $q_{74}$ | $T1wo.Sc.T2o.FL_{39}$ | $q_{75}$ | $T1wo.Sc.T2o.FL_{47}$ | $q_{76}$ |
| $T1o.Sc.T2o.FP_{19}$ | $q_{77}$ | $T1o.Sc.T2o.FP_{16}$ | $q_{78}$ | $T1o.Sc.T2o.FL_{39}$ | $q_{79}$ | $T1o.Sc.T2o.FL_{47}$ | $q_{80}$ |

Table 3.4: Mode labels belonging to $\mathcal{Q}_{\mathcal{F}_{ns}}$

control gate is open or closed.

| | Gate open and stuck open ($\alpha_1 = 1$) | | | | Gate closed and stuck close $\alpha_1 = 0$ | | | |
|---|---|---|---|---|---|---|---|---|
| $q_i$ | $\mathbf{A}_i$ | $\mathbf{B}_i$ | $\mathbf{E}_{xi}$ | $q_i$ | $\mathbf{A}_i$ | $\mathbf{B}_i$ | $\mathbf{E}_{xi}$ |
| 1,9 | $\begin{bmatrix} 1-\Delta t\beta_1 & 0 \\ \alpha_1\Delta t\beta_1 & 1-\Delta t\beta_2 \end{bmatrix}$ | $\begin{bmatrix} \Delta tS_1\varphi_{19} & 0 \\ 0 & \Delta tS_2\varphi_{16} \end{bmatrix}$ | $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ | 5,10 | $\begin{bmatrix} 1-\Delta t\beta_0 & 0 \\ 0 & 1-\Delta t\beta_2 \end{bmatrix}$ | $\begin{bmatrix} \Delta tS_1\varphi_{19} & 0 \\ 0 & \Delta tS_2\varphi_{16} \end{bmatrix}$ | $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ |
| 2,11 | $\begin{bmatrix} 0 & 0 \\ 0 & 1-\Delta t\beta_2 \end{bmatrix}$ | $\begin{bmatrix} 0 & 0 \\ 0 & \Delta tS_2\varphi_{16} \end{bmatrix}$ | $\begin{bmatrix} \overline{v_1} \\ \alpha_1\Delta t\beta_0\overline{v_0} \end{bmatrix}$ | 6,12 | $\begin{bmatrix} 0 & 0 \\ 0 & 1-\Delta t\beta_2 \end{bmatrix}$ | $\begin{bmatrix} 0 & 0 \\ 0 & \Delta tS_2\varphi_{16} \end{bmatrix}$ | $\begin{bmatrix} \overline{v_1} \\ 0 \end{bmatrix}$ |
| 3,13 | $\begin{bmatrix} 1-\Delta t\beta_1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} \Delta tS_1\varphi_{19} \\ 0 \quad 0 \end{bmatrix}$ | $\begin{bmatrix} 0 \\ \overline{v_2} \end{bmatrix}$ | 7,14 | $\begin{bmatrix} 1-\Delta t\beta_1 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} \Delta tS_1\varphi_{19} \\ 0 \quad 0 \end{bmatrix}$ | $\begin{bmatrix} 0 \\ \overline{v_2} \end{bmatrix}$ |
| 4,15 | $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} \overline{v_1} \\ \overline{v_2} \end{bmatrix}$ | 8,16 | $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} \overline{v_1} \\ \overline{v_2} \end{bmatrix}$ |

Table 3.5: State space matrices for each mode $q_i \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$

The output function (3.3) is given by equation (3.35).

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} \frac{\beta_1}{M_{39}} & 0 \\ 0 & \frac{\beta_2}{M_{41}} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \tag{3.35}$$

where the same matrix $\mathbf{C}_i$ gain for all modes and $\mathbf{D}_i = \mathbf{0}$.

### 3.9.2 Residual generation

The set of residuals for all modes using the input-output approach described by the residual expression given by Equation (3.8) are given in Table 3.6. There are two residuals per operation mode and five non-discernible mode sets as shown in Table 3.7.

| | Gate open and stuck open ($\alpha = 1$) | | | | Gate closed and stuck close $\alpha = 0$ | | | |
|---|---|---|---|---|---|---|---|---|
| $q_i$ | $\mathbf{G}_i$ | $\mathbf{H}_i$ | $\mathbf{E}_i$ | $q_i$ | $\mathbf{G}_i$ | $\mathbf{H}_i$ | $\mathbf{E}_i$ |
| 1,9 | $\begin{bmatrix} \frac{\Delta t\,\beta_1\,S_1\,\phi_1}{M_{39}\,p} & 0 \\ 0 & \frac{\Delta t\,\beta_2\,S_2\,\phi_2}{M_{41}\,p} \end{bmatrix}$ | $\begin{bmatrix} \frac{1-\Delta t\,\beta_1}{p} & 0 \\ \frac{\Delta t\,\beta_2\,\alpha_1\,M_{39}}{M_{41}\,p} & \frac{1-\Delta t\,\beta_2}{p} \end{bmatrix}$ | $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ | 5,10 | $\begin{bmatrix} \frac{\Delta t\,\beta_1\,S_1\,\phi_1}{M_{39}\,p} & 0 \\ 0 & \frac{\Delta t\,\beta_2\,S_2\,\phi_2}{M_{41}\,p} \end{bmatrix}$ | $\begin{bmatrix} \frac{1-\Delta t\,\beta_1}{p} & 0 \\ 0 & \frac{1-\Delta t\,\beta_2}{p} \end{bmatrix}$ | $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ |
| 2,11 | $\begin{bmatrix} 0 & 0 \\ 0 & \frac{\Delta t\,\beta_1\,S_{16}\,\phi_1}{M_{41}\,p} \end{bmatrix}$ | $\begin{bmatrix} \frac{1-\Delta t\,\beta_1}{p} & 0 \\ \frac{\Delta t\,\beta_1\,\alpha_1\,M_{39}}{M_{41}\,p} & \frac{1-\Delta t\,\beta_2}{p} \end{bmatrix}$ | $\begin{bmatrix} \frac{\overline{v_0}\,\beta_1}{M_{39}\,p} \\ \frac{\Delta t\,\beta_2\,\alpha\,\beta_0\,\overline{v_1}}{M_{41}\,(p-1+\Delta t\,\beta_1)} \end{bmatrix}$ | 6,12 | $\begin{bmatrix} 0 & 0 \\ 0 & \frac{\Delta t\,\beta_2\,S_2\,\phi_2}{M_{41}\,p} \end{bmatrix}$ | $\begin{bmatrix} \frac{1-\Delta t\,\beta_1}{p} & 0 \\ 0 & \frac{1-\Delta t\,\beta_2}{p} \end{bmatrix}$ | $\begin{bmatrix} \frac{\overline{v_1}\,\beta_1}{M_{39}\,p} \\ 0 \end{bmatrix}$ |
| 3,13 | $\begin{bmatrix} \frac{\Delta t\,\beta_1\,S_1\,\phi_1}{M_{39}\,p} & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} \frac{1-\Delta t\,\beta_1}{p} & 0 \\ \frac{\Delta t\,\beta_1\,\alpha\,M_{39}}{M_{41}\,p} & \frac{1-\Delta t\,\beta_2}{p} \end{bmatrix}$ | $\begin{bmatrix} 0 \\ \frac{\overline{v_2}\,\beta_2}{M_{41}\,p} \end{bmatrix}$ | 7,14 | $\begin{bmatrix} \frac{\Delta t\,\beta_0\,S_1\,\phi_1}{M_{39}\,p} & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} \frac{1-\Delta t\,\beta_1}{p} & 0 \\ 0 & \frac{1-\Delta t\,\beta_2}{p} \end{bmatrix}$ | $\begin{bmatrix} 0 \\ \frac{\overline{v_1}\,\beta_2}{M_{41}\,p} \end{bmatrix}$ |
| 4,15 | $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} \frac{\overline{v_1}\,\beta_1}{M_{39}\,p} \\ \frac{\overline{v_2}\,\beta_2}{M_{41}\,p} \end{bmatrix}$ | 8,16 | $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ | $\begin{bmatrix} \frac{\overline{v_1}\,\beta_1}{M_{39}\,p} \\ \frac{\overline{v_2}\,\beta_2}{M_{41}\,p} \end{bmatrix}$ |

Table 3.6: Residual generation for all modes using input-output models

Lines from 8 to 15 in Algorithm 3.1 allow to obtain this information gathering together the set of modes with equivalent residuals. In online diagnosis, only the set of residuals corresponding to active sets are computed. The active sets include the sets the current mode and their successors belong to.

| Groups | Modes non-discernible |
|---|---|
| $\mathcal{Q}_{\nu_1}$ | $[1, 9]$ |
| $\mathcal{Q}_{\nu_2}$ | $[5, 10]$ |
| $\mathcal{Q}_{\nu_3}$ | $[2, 6, 11, 12]$ |
| $\mathcal{Q}_{\nu_4}$ | $[3, 7, 13, 14]$ |
| $\mathcal{Q}_{\nu_5}$ | $[4, 8, 15, 16]$ |

Table 3.7: Non-discernible mode sets ($\mathcal{Q}_{disc}$)

The following fault distribution matrices are defined:

$$\mathbf{F}_{y_i} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \qquad \mathbf{F}_{x_i} = \begin{bmatrix} -\mathbf{B}_i & \mathbf{0} \end{bmatrix}$$

These matrices are used to generate a fault signature matrix $FS_{\nu_i}$ for every non-discernible mode set, applying Equation (3.13).

| | $f_3$ | $f_4$ | $f_5$ | $f_6$ | |
|---|---|---|---|---|---|
| $\mathbf{FS}_{\nu_1}$ | 1 | 0 | 1 | 0 | $\mathcal{F}^1_{\nu_1} = \{f_3\}, \mathcal{F}^3_{\nu_1} = \{f_5\}$ |
| | 1 | 1 | 0 | 1 | $\mathcal{F}^2_{\nu_1} = \{f_4, f_6\}$ |
| $\mathbf{FS}_{\nu_2}$ | 1 | 0 | 1 | 0 | $\mathcal{F}^2_{\nu_2} = \{f_4, f_6\}$ |
| | 0 | 1 | 0 | 1 | $\mathcal{F}^1_{\nu_2} = \{f_3, f_5\}$ |
| $\mathbf{FS}_{\nu_3}$ | 1 | 0 | 0 | 0 | $\mathcal{F}^1_{\nu_3} = \{f_3\}$ |
| | 0 | 1 | 0 | 1 | $\mathcal{F}^2_{\nu_3} = \{f_4, f_6\}$ |
| $\mathbf{FS}_{\nu_4}$ | 1 | 0 | 1 | 0 | $\mathcal{F}^1_{\nu_4} = \{f_3, f_5\}$ |
| | 0 | 1 | 0 | 0 | $\mathcal{F}^2_{\nu_4} = \{f_4\}$ |
| $\mathbf{FS}_{\nu_5}$ | 1 | 0 | 0 | 0 | $\mathcal{F}^1_{\mathcal{Q}_{\nu_5}} = \{f_3\}$ |
| | 0 | 1 | 0 | 0 | $\mathcal{F}^2_{\mathcal{Q}_{\nu_5}} = \{f_4\}$ |

Table 3.8: Non-structural fault signature matrices per each non-discernible mode set

### 3.9.3 Behavior automaton

Following Algorithm 3.1 the behavior automaton $B$ shown in Fig. 3.7 is obtained. The number of modes is $|\overline{\mathcal{Q}}| = 130$ and the number of explored transitions is 194. The signature-events generated by function (3.34) are shown in Table 3.9. The algorithm has been implemented in MATLAB and it assigns a numeric code for each generated event as it can be seen in the first and fourth columns of the table. Signature-events represent changes in the active sets of non-discernible modes. Fig. 3.7 shows in dashed line the transient modes taken into account for transitions that can be detected using residuals.

| Code | Event | Type | Code | Event | Type |
|---|---|---|---|---|---|
| 27 | $\delta_{13}$ | signature-event | 41 | $\delta_{35}$ | signature-event |
| 33 | $\delta_{14}$ | signature-event | 17 | $\delta_{31}$ | signature-event |
| 5 | close | controlled | 310 | $\delta_{\mathcal{F}^1_{\nu_3}}$ | signature-event |
| 6 | open | controlled | 320 | $\delta_{\mathcal{F}^2_{\nu_3}}$ | signature-event |
| 21 | $\delta_{12}$ | signature-event | 42 | $\delta_{45}$ | signature-event |
| 110 | $\delta_{\mathcal{F}^1_{\nu_1}}$ | signature-event | 18 | $\delta_{41}$ | signature-event |
| 120 | $\delta_{\mathcal{F}^2_{\nu_1}}$ | signature-event | 410 | $\delta_{\mathcal{F}^1_{\nu_4}}$ | signature-event |
| 130 | $\delta_{\mathcal{F}^3_{\nu_1}}$ | signature-event | 420 | $\delta_{\mathcal{F}^2_{\nu_4}}$ | signature-event |
| 37 | $\delta_{54}$ | signature-event | 16 | $\delta_{21}$ | signature-event |
| 510 | $\delta_{\mathcal{F}^1_{\nu_5}}$ | signature-event | 210 | $\delta_{\mathcal{F}^1_{\nu_2}}$ | signature-event |
| 520 | $\delta_{\mathcal{F}^1_{\nu_5}}$ | signature-event | 220 | $\delta_{\mathcal{F}^2_{\nu_2}}$ | signature-event |
| 28 | $\delta_{23}$ | signature-event | 23 | $\delta_{32}$ | signature-event |
| 34 | $\delta_{24}$ | signature-event | 24 | $\delta_{42}$ | signature-event |

Table 3.9: Event codification used in the behavior automaton

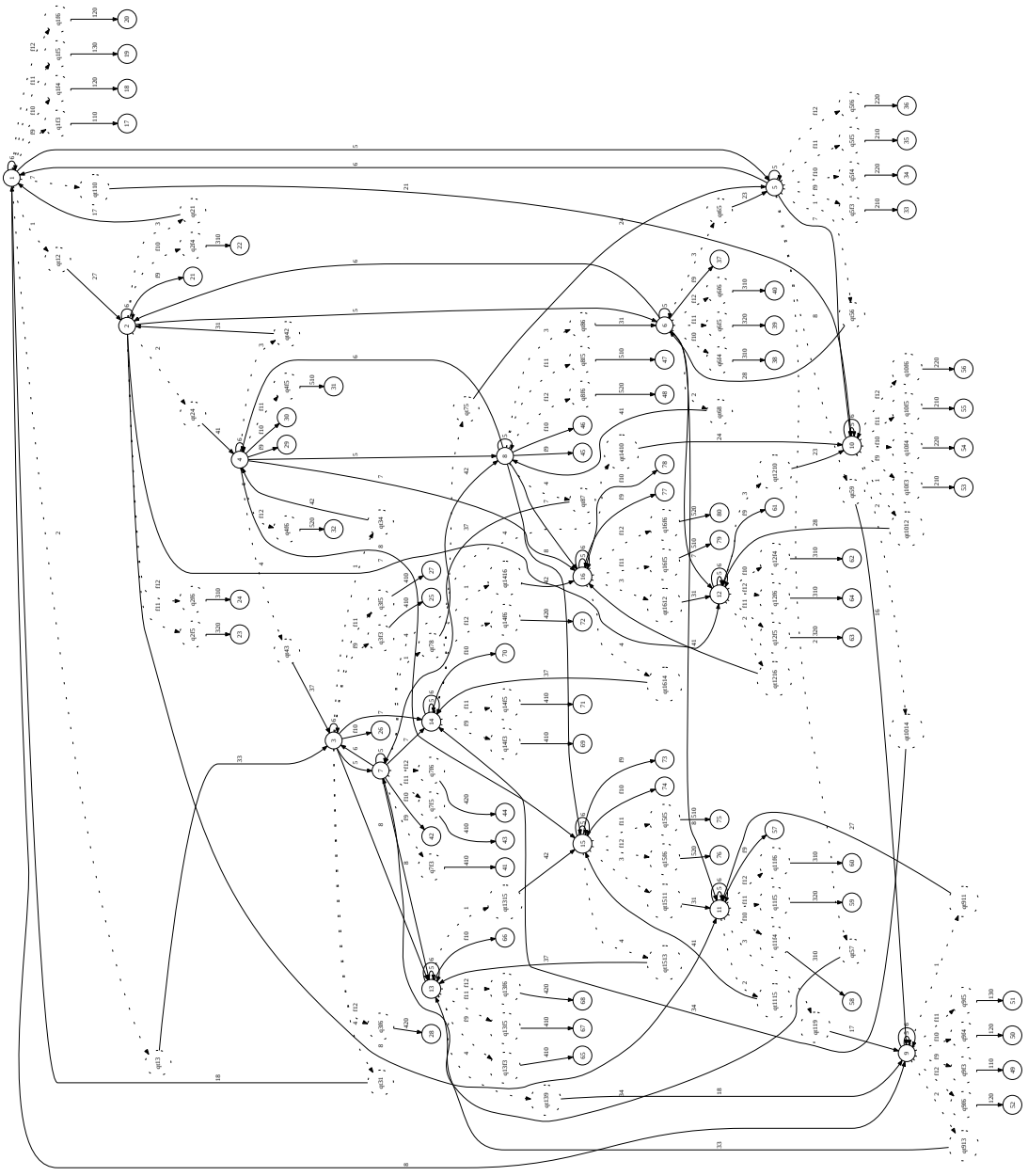

Figure 3.7: Behavior automaton obtained in the MATLAB implementation

### 3.9.4 Diagnoser for two-tanks sewer network

The diagnoser without silent closure is shown in Fig. 3.8. The number of states generated is $|\mathcal{Q}_D| = 59$ and the number of generated transitions is 188. The diagnoser was generated using DIADES tool [Ramadge and Wonham, 1989]. The initial state is assumed known and nominal.

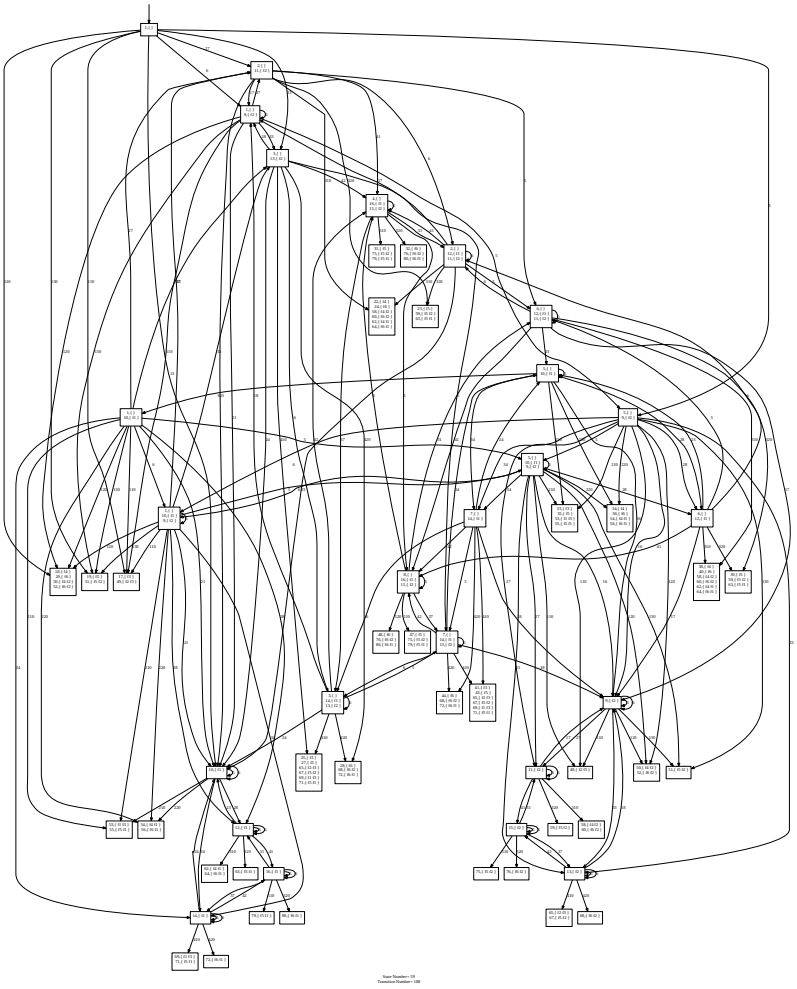

Figure 3.8: Diagnoser without silent closure obtained using DIADES tool

### 3.9.5 Simulation scenario

To validate the methodology, consider the measurements provided by the limnimeters and the rain gauges in the sewer network in Fig. 3.9. These measurements correspond to the following system mode sequence: $q_1 \rightarrow q_3 \rightarrow q_1 \rightarrow q_5$. Mode $q_1$ refers to the situation in which no tank is in overflow. Then, $T_2$ is in overflow during a period of time (mode $q_3$) until it leaves the overflow situation (mode $q_1$). Later, the control gate is closed. Thus, the diagnoser must track the right mode sequence and detect and isolate the possible faults.

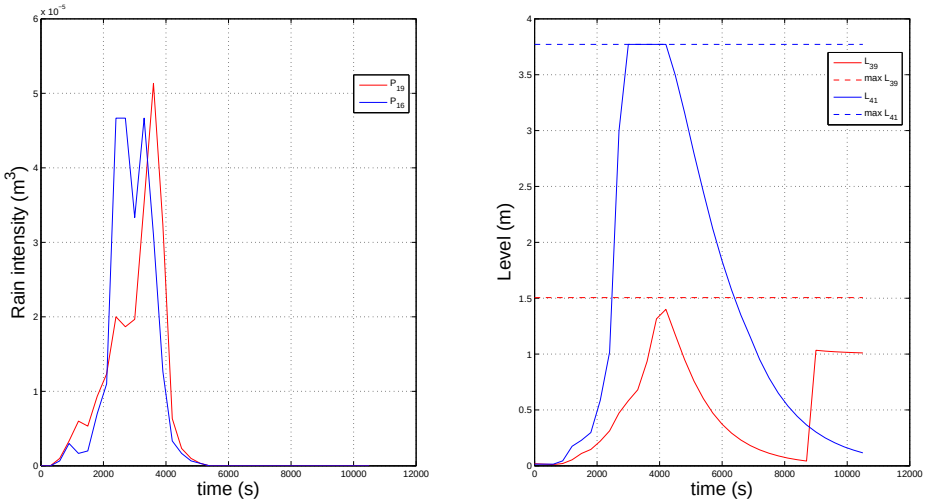


Figure 3.9: Measurements for the simulation scenario with a sampling time of $\Delta t = 300s$

Fig. 3.10 plots the set of residuals for all sets in $\mathcal{Q}_{disc}$. According to the set of residuals for the set of modes of $HA$, two signature-events, $\delta_{14}$ and $\delta_{41}$, were appropriately identified using consistency indicators. These signature-events correspond to transitions $q_1 \rightarrow q_3$ and $q_3 \rightarrow q_1$ with $q_1 \in \mathcal{Q}_{\nu_1}$ and $q_3 \in \mathcal{Q}_{\nu_4}$ detected at $3300s$ and $4500s$ respectively. Notice for instance that when the system is in mode $q_3$, $\Phi_{\nu_1} \neq \mathbf{0}$ and $\Phi_{\nu_4} = \mathbf{0}$. Both modes $q_1, q_3 \in \mathcal{Q}$ represent a nominal behavior. In Fig. 3.10, red vertical dashed lines correspond to mode changes belonging to $\mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$ and the vertical black dashed line corresponds to a mode changes belonging to $\mathcal{Q}_{\mathcal{F}_{ns}}$.

Later, observable event $\sigma_{o1}$ occurs at $6900s$, corresponding to the control gate closing. This event is identified instantaneously and indicates that a mode change from $q_1$ to $q_5$ takes place. It should be noticed in Fig. 3.10 that the residuals in mode $q_5 \in \mathcal{Q}_{\nu_2}$ are consistent with measurements after $\sigma_{o_1}$ occurrence, i.e. $\Phi_{\nu_2}(k) = \mathbf{0}$, and in mode $(q_9, \{f_2\})$, $\Phi_{\nu_2}(k) \neq \mathbf{0}$ which implies that an inconsistency has occurred. Therefore, they are discernible and the diagnoser state contains only mode $q_5$ as a feasible

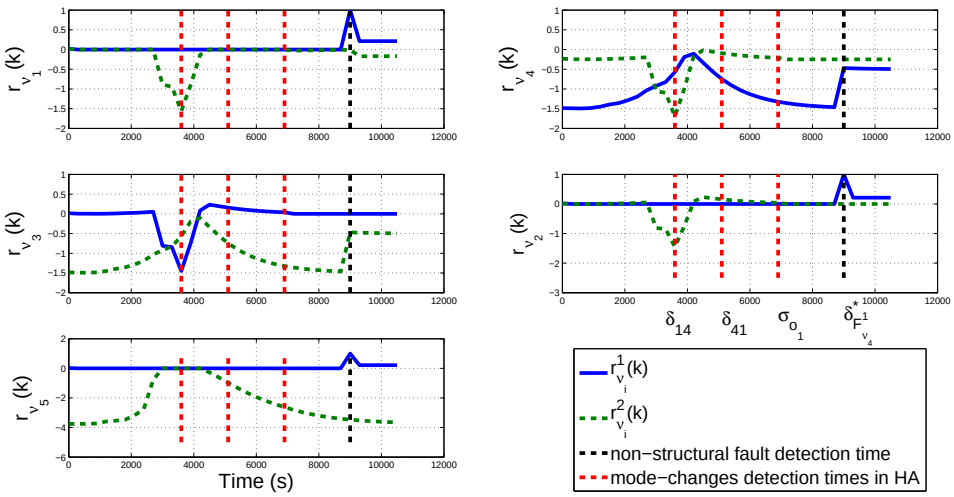


Figure 3.10: Residuals for non-discernible groups

current operation mode.

Later, a non-structural fault occurs at $9000s$. In this case, the diagnoser detects the fault at $9300s$. The set of consistency indicators of mode $q_5$ are used to isolate the fault. The observed signature is $[\ 1 \quad 0\ ]^t$ which, according to $\mathbf{FS}_{\nu_4}$, corresponds to a fault in sensor $L_{39}$. Finally, the hybrid diagnoser stops and reports the diagnosis. Indeed, a non-structural fault needs to be repaired before the diagnoser can resume.

The report given by the hybrid diagnoser is shown in Table 3.10. The first column corresponds to mode changes in $HA$, whereas in the second column, the identified events are collected. The third column presents the diagnoser state information. The last two columns show the occurrence time and detection time of the identified events. It can be seen that there is a maximum delay of two sampling times in the mode change detection. Fig. 3.11 illustrates the mode and diagnoser state sequences.

| Mode change | Reported event | State diagnoser | Occurrence time (s) | Detection time (s) |
|---|---|---|---|---|
| $q_1 \rightarrow q_3$ | $\delta_{14}$ | $(q_3, \{\}), (q_{13}, \{f_2\})$ | 3000 | 3300 |
| $q_3 \rightarrow q_1$ | $\delta_{41}$ | $(q_1, \{\}), (q_9, \{f_2\})$ | 4200 | 4500 |
| $q_1 \rightarrow q_5$ | $\sigma_{o_1}$ | $(q_5, \{\}), (q_9, \{f_2\})$ | 6900 | 6900 |
| $q_5 \rightarrow q_5^3$ fault $f_7 \in \mathcal{F}_{ns}$ in Mode $q_5$ | $\delta_{\mathcal{F}^1_{\nu_4}}$ | $(q_5^1, \{f_3\}), (q_5^2, \{f_5\})$ | 9000 | 9300 |

Table 3.10: Hybrid diagnoser report

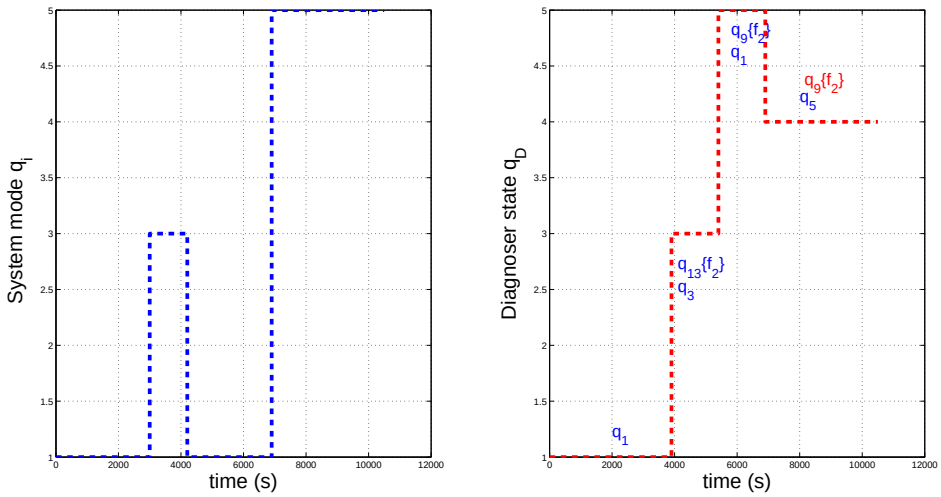


Figure 3.11: Mode sequence vs. diagnoser state sequence

### 3.9.6 Discussion

In general terms, the number of diagnoser states to be generated is at the worst case greater than $|\mathcal{Q}_D| = 2^{n_q}$, which implies exponential complexity. In addition, the total number of residuals to be generated is $|\mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}|n_{ri}$, assuming that the number of residuals does not differ per mode. Thus, the main disadvantage of this approach is the complexity of the system model. Consequently, the number of modes may be a limiting factor, for the online implementation.

The active sets of non-discernible modes allow to reduce the computational cost of the residuals computation, but the searching algorithm to update the active sets after an event occurs increases this computational cost. The major problem of the methodology concerns the online process, because in the offline process time is not a limiting factor even if the number of modes is big.

# CHAPTER 4

# INCREMENTAL METHODOLOGY TO HYBRID SYSTEMS DIAGNOSIS

## 4.1 Principles of the method

The scheme of the incremental hybrid diagnosis architecture is provided in Fig. 4.1. The method consists in incrementally building the hybrid model through the composition of automata describing system component. The set of linear equations concerning all components are parametrized as a mode function. Mode sequence tracking and incremental diagnoser building are synchronously carried out, taking just into account the set of modes the system is possibly operating in and their successors.

The behavior automaton includes the so called signature-events that abstract the residual behaviors. Transitions labeled by unobservable events in the hybrid automaton may hence turn into observable by means of the signature-events according to the discernibility property, as explained in Chapter 3.

Incremental hybrid diagnosis is directly performed by understanding the events and measurements issued by the physical system on the hybrid automaton model. Therefore, just the useful parts of the diagnoser are incrementally built, only developing the traces that are required to explain the occurrence of incoming events. Generally, a hybrid system operates in a small region in comparison to the entire behavioral space defined by the hybrid automaton modes. A significant gain can hence be expected from the proposed approach.

Input events are identified instantaneously and signature-events are determined by looking at those successor modes whose consistency indicators are in agreement with measurements, or checking the consistency indicators against the fault signature matrix. Incremental hybrid diagnosis is based on the same principles and assumptions explained in Chapter 3. The key idea is to reduce the implementation cost of the hybrid diagnosis scheme when dealing with large complex systems by avoiding the offline building of the full diagnoser.

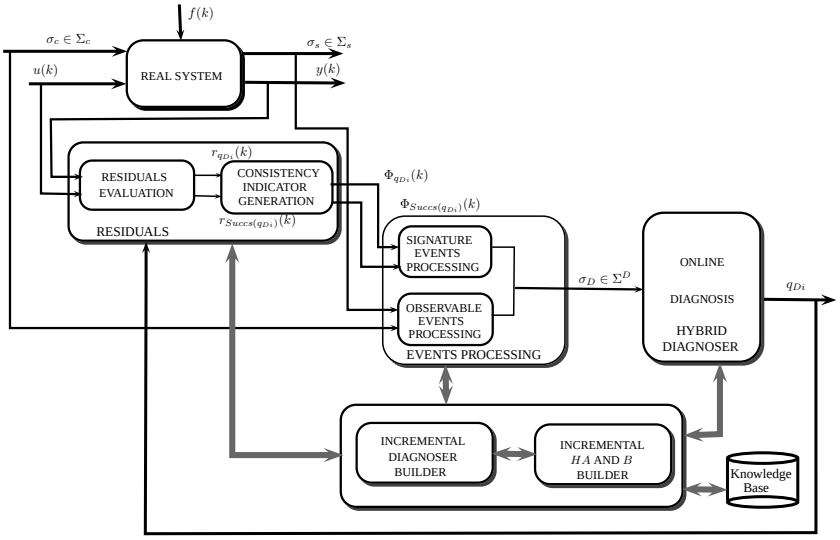


Figure 4.1: Conceptual block diagram for online diagnosis methodology

The hybrid model and the behavior automaton are updated whenever the system reaches a new operation mode as is shown in Fig. 4.1. The *Incremental Diagnoser Builder* block builds then the corresponding piece of the diagnoser. Once a piece of the diagnoser is built, 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 and used in the *Residuals* block. Hence, the *Residuals* block computes the consistency indicators needed by the *Event Processing* block. The *Event Processing* block detects the occurrence of an observable event: either an input event of the system or a signature-event generated by residuals.

The *On-line Diagnosis* block displays messages about the current diagnoser state and the possible occurrence of a fault. The number of system modes associated with a diagnoser state depends on the hybrid system diagnosability. After an event occurrence, the hybrid diagnoser traces feasible mode changes and detects and isolates potential faults.

Systematically, the steps followed into the methodology are schematized in Fig. 4.2.

In the offline process, the initialization involves building initial automata $HA_{init}$, $B_{init}$ and $D_{init}$. These initial automata includes the information of the initial state of the system components assuming that the initial mode is known. The number of modes of the initial $HA_{init}$ varies according to the discernibility property involving the current mode and their successors. The minimal number of modes in $HA_{init}$ is $|Succs(q_i)| + 1$. Otherwise the number of modes increases according to those successor modes that are non-discernible with respect to the current mode. Initial automata are required to start the online diagnosis tasks.

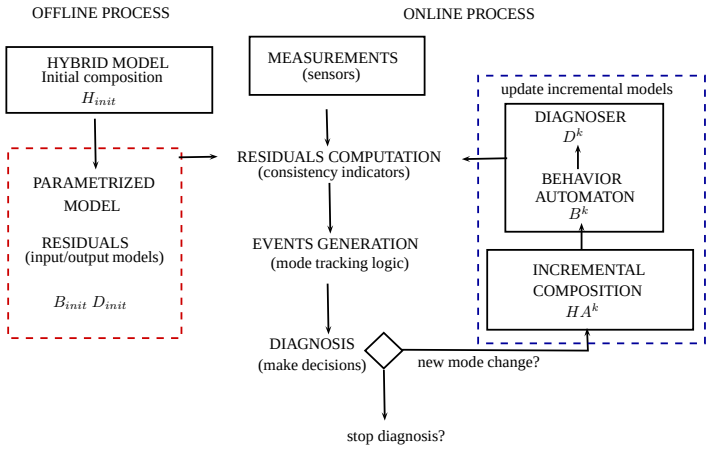


Figure 4.2: Steps to follow in the methodology

On the other hand, a parametrized model is generated in order to automatically obtain the model of the system operation modes involved in the initialization and in the online process. The remaining diagnosis tasks are executed online. These involve updating the model for the new operation modes and building $HA^k$, $B^k$ and $D^k$ whenever a mode change is detected. $HA^k$, $B^k$ and $D^k$ represent the hybrid automaton, the behavior automaton and the diagnoser corresponding to time instant $k$.

The incremental diagnoser building is an adaptation of the algorithm proposed in [Sampath et al., 1995], such that the propagation is carried out taking into account the first occurrence of an observable event, only extending the traces of the diagnoser concerning the current state and their successors.

## 4.2 Incremental hybrid model

The incremental hybrid model $HA^k$ is updated whenever an event is detected. Its initial mode corresponds to the composition of the initial state of all components, and is defined by $q_0$, assuming that the initial states of all components are known and under no failure. The incremental $HA^k$ is defined as:

$$HA^k =< \mathcal{Q}^k\mathcal{X}, \mathcal{U}, \mathcal{Y}, \mathcal{F}, \mathcal{G}^k, \mathcal{H}^k, \Sigma^k, \mathcal{T}^k >$$

Notice that some of the elements vary is size during the online diagnosis process. As mentioned before, $HA_{init}$ is built offline and required to start the online diagnosis process. The initialization process is described in Algorithm 4.1.

Line 2 in Algorithm 4.1 computes $HA_{init}$ through Algorithm 4.2, obtaining $\mathcal{Q}_{init}$, $\Sigma_{init}$ and $\mathcal{T}_{init}$,

**Algorithm 4.1** Initialization()

1: Set initial state for all components and the system mode, i.e. $q_0$.
2: $HA_{init}$:=Incremental_HA_Builder$(q_0)$
3: **for all** $f_w \in \mathcal{F}_{ns}$ **do**
4: $\quad \Sigma_{init} := \{\sigma_{f_w}\} \cup \Sigma_{init}$.
5: **end for**
6: $B_{init}$:=B_Builder($\mathcal{Q}_{init}$).
7: $D_{init}$ is built from $B_{init}$.

which they represent the set of modes, events and transitions generated in the initial composition, respectively. $H_{init}$ must contain at least the initial mode and their successors, assuming they are discernible. Fault events ($\Sigma_{\mathcal{F}}$) are included in $\Sigma_{init}$ in the first iteration. Besides, the set of residuals for the modes in $\mathcal{Q}_{init}$ are computed and the knowledge-base is generated taking into account these modes.

Algorithm 4.2 incrementally builds the hybrid model whenever a change in the system is detected through consistency indicators or an observable event occurs. $HA^k$ is built by the composition of automata ($DA_{Ac}$) along with parametrized equations which allow to obtain the model equations (3.2)-(3.3).

**Algorithm 4.2** Incremental_HA_Builder$(q_D)$

1: Create a queue $\mathcal{L}_h$
2: **for all** $q_i \in q_D$ such that $q_i \in \mathcal{Q}^k_{\mathcal{N}} \cup \mathcal{Q}^k_{\mathcal{F}_s}$ **do**
3: $\quad$ Enqueue $q_i$ onto $\mathcal{L}_h$
4: **end for**
5: **while** $\mathcal{L}_h$ is not empty **do**
6: $\quad q_i$:= dequeue $\mathcal{L}_h$
7: $\quad$ **for all** $f_w \in \mathcal{F}_{ns}$ **do**
8: $\quad\quad \mathcal{Q}^k := \{q_{f_w i}\} \cup \mathcal{Q}^{k-1}$.
9: $\quad\quad \mathcal{T}(q_i, \sigma_{f_w}) = q_{f_w i}$.
10: $\quad$ **end for**
11: $\quad DA_{Ac}$ :=**incremental_parallel_composition(**$q_i$**)**
12: $\quad$ **for all** $\sigma_{\mathcal{M}} \in \Gamma_{Ac}(q_i)$ **do**
13: $\quad\quad \mathcal{T}^k(q_i, \sigma_{\mathcal{M}}) := \mathcal{T}_{Ac}(q_i, \sigma_{\mathcal{M}})$.
14: $\quad\quad$ **if** $\sigma_{\mathcal{M}} \notin \Sigma^{k-1}$ **then**
15: $\quad\quad\quad \Sigma^k := \sigma_{\mathcal{M}} \cup \Sigma^{k-1}$.
16: $\quad\quad$ **end if**
17: $\quad\quad$ **if** $q_j \notin \mathcal{Q}^k$ **then**
18: $\quad\quad\quad \mathcal{Q}^k := \{q_j\} \cup \mathcal{Q}^{k-1}$.
19: $\quad\quad\quad$ Instantiate equations for this mode.
20: $\quad\quad\quad$ Compute residual expression for $\mathbf{r}_j(\bullet)$.
21: $\quad\quad\quad$ **Update_Knowledge_Base(**$q_j$**)**.
22: $\quad\quad\quad$ **if** $\sigma_{\mathcal{M}} \in \Sigma_{uo}$ **then**
23: $\quad\quad\quad\quad$ **if** $(q_i, q_j)$ are non-discernible according to (3.15) **then**
24: $\quad\quad\quad\quad\quad$ Enqueue $q_l$ onto $\mathcal{L}_h$
25: $\quad\quad\quad\quad$ **end if**
26: $\quad\quad\quad$ **end if**
27: $\quad\quad$ **end if**
28: $\quad$ **end for**
29: **end while**

The **incremental_parallel_composition** function updates the discrete part of $HA^k$. The parallel

composition (2.9) is adapted to only generate 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 $HA^k$.

The parallel composition for $N$ components is given by $DA_{Ac}$, where $\mathcal{Q}_{Ac} = \mathcal{Q}_{\mathcal{M}_1} \times \cdots \times \mathcal{Q}_{\mathcal{M}_N}$, $\Sigma_{Ac} = \Sigma_1 \cup \cdots \cup \Sigma_N$, $q_0 = (q_{0_1}, \cdots, q_{0_N})$ and $\mathcal{T}_{Ac}((q_1, \cdots, q_N), \sigma_{\mathcal{M}})$ with $\mathcal{T}_{Ac} = \mathcal{T}_{1||\cdots||N}$.

The knowledge-base is updated whenever a new mode of $HA^k$ is generated. The new mode is classified into the corresponding non-discernible mode sets $\mathcal{Q}^k_{disc}$ as is shown in Algorithm 4.3.

**Algorithm 4.3** Update_Knowledge_Base($q_j$)

```
if q_j ∈ Q^k_N ∪ Q^k_{F_s} then
    Classify q_j into Q^k_disc.
    if q_j creates a new group ν in Q^k_disc then
        Compute FS_ν(•).
        Determine the subsets of isolable faults F*_ν.
        Determine the set of non-isolable faults F'_ν.
        Store in memory FS_ν(•), F*_ν, F'_ν.
    end if
end if
Update and store in memory Q^k_disc
```

The parameterized system model as a function of the operation mode is composed of the whole set of equations concerning the components and their interconnections. The state space model of each mode in the incremental hybrid model can be represented by equations (4.1)-(4.2). State space matrices depend on system parameters and they are instantiated for the modes obtained in the incremental composition.

$$\begin{aligned}\mathbf{x}(k+1) = \mathbf{A}_i\mathbf{x}(k) + \mathbf{B}_i\mathbf{u}(k) + \mathbf{F}_{xi}\mathbf{f}(k) + \mathbf{E}_{xi} + \sum_{j=1}^{n_{S_i}} \mu^j_{x_i} S^j_i(\mathbf{x}(k), \mathbf{u}(k)) \\ + \sum_{j=1}^{n_{D_i}} \psi^j_{x_i} D^j_i(\mathbf{x}(k), \mathbf{u}(k))\end{aligned} \tag{4.1}$$

$$\begin{aligned}\mathbf{y}(k) = \mathbf{C}_i\mathbf{x}(k) + \mathbf{D}_i\mathbf{u}(k) + \mathbf{F}_{y_i}\mathbf{f}(k) + \mathbf{E}_{y_i} + \sum_{j=1}^{n_{S_i}} \mu^j_{y_i} S^j_i(\mathbf{x}(k), \mathbf{u}(k)) \\ + \sum_{j=1}^{n_{D_i}} \psi^j_{y_i} D^j_i(\mathbf{x}(k), \mathbf{u}(k))\end{aligned} \tag{4.2}$$

$S^j_i$ and $D^j_i$ functions model the saturation and dead zone nonlinearities that appear in the evolution and observation equations following the methodology in [Bayoudh et al., 2009] (see Fig. 4.3). $n_{S_i}$ and $n_{D_i}$ denote the number of saturation and dead zone nonlinearities introduced by a subset of components, $\mu^j_{y_i}$ and $\psi^j_{y_i} \in \mathcal{R}^{n_y} \times \mathcal{R}$, $\mu^j_{x_i}$ and $\psi^j_{x_i} \in \mathcal{R}^{n_x} \times \mathcal{R}$.

$$S_i^j(\mathbf{x}(k),\mathbf{u}(k)) = \begin{cases} -M_i^j & \text{if } L_i^j\mathbf{x}(k) + K_i^j\mathbf{u}(k) < -M_i^j \\ L_i^j\mathbf{x}(k) + K_i^j\mathbf{u}(k) & \text{if } |L_i^j\mathbf{x}(k) + K_i^j\mathbf{u}(k)| \leq M_i^j \\ M_i^j & \text{if } L_i^j\mathbf{x}(k) + K_i^j\mathbf{u}(k) > M_i^j \end{cases} \tag{4.3}$$

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.

$$D_i^j(\mathbf{x}(k),\mathbf{u}(k)) = \begin{cases} 0 & \text{if } |F_i^j\mathbf{x}(k) + Z_i^j\mathbf{u}(k)| \leq N_i^j \\ F_i^j\mathbf{x}(k) + Z_i^j\mathbf{u}(k) & \text{otherwise} \end{cases} \tag{4.4}$$

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.

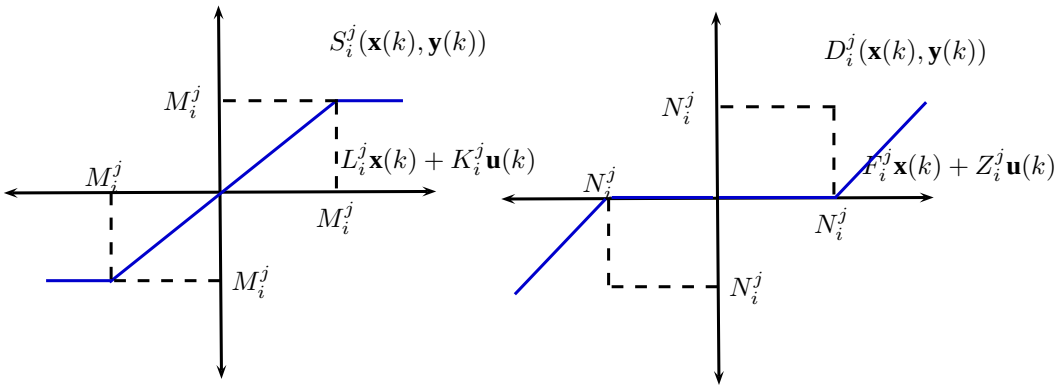


Figure 4.3: Saturation and dead zone representation

Without loss of generality, non-linearities are included in the system equations as a tool to efficiently adapt the system mathematical equations to mode changes needed for the on-line diagnosis. However, the continuous dynamics in a mode are represented by linear equations, as a result of the region combinations of the saturation and dead zone. Therefore, the set of residuals is generated by Eq. (3.8). Discernability properties given in Section 3.6 can be applied to build the behavior automaton ($B^k$). Transitions between modes depend on the non-linearities conditions and observable events of the system.

**Assumption 4.1.** *The parametrized equations using non-linearities is applied to those systems which admit both representations, as a hybrid system and as a non-linear system.*

## 4.3 Incremental behavior automaton

The incremental behavior automaton is defined by $B^k =< \overline{Q}^k, \overline{\Sigma}^k, \overline{T}^k, \overline{q}_0 >$.

Algorithm 4.4 explores $HA^k$ taking into account only the modes in which physical system is possibly operating in at time instant $k$. $B^k$ is built assuming that the system can be operating in a set of belief modes denoted by $q_D$. Then, an exploration of each successor mode $q_j \in Succs_{HA}(q_i), q_i \in q_D$ is carried out.

**Algorithm 4.4** B_Builder($q_D$)

Create a queue $\mathcal{L}$.
**for all** $q_i \in q_D$ **do**
  Enqueue $q_i$ onto $\mathcal{L}$
**end for**
**while** $\mathcal{L}$ is not empty **do**
  $q_i$:=dequeue $\mathcal{L}$
  **for all** $q_j \in Succs_{HA}(q_i)$ **do**
    **if** $q_j \notin \mathcal{Q}^k \cap \overline{\mathcal{Q}}^k$ **then**
      $\overline{\mathcal{Q}}^k = \{q_j\} \cup \overline{\mathcal{Q}}^{k-1}$
    **end if**
    Let $\sigma$ is such as $\mathcal{T}(q_i, \sigma) = q_j$ :
    **switch** ($\sigma$)
    **case** $\sigma \in \Sigma_o^k$**:**
      $\overline{\mathcal{T}}^k(q_i, \sigma) := q_j$.
    **case** $\sigma \in \Sigma_{uo}^k$**:**
      **if** $q_i$ and $q_j$ are discernible according to (3.15) **then**
        $\mathcal{Q}^{t^k} = \{q_{i-j}^t\} \cup \mathcal{Q}^{t^{k-1}}$.
        $\delta := f_{Sig\_ev}(q_i, q_j)$ according to (3.34).
        **if** $\delta \notin \overline{\Sigma}^{k-1}$ **then**
          $\overline{\Sigma}^k = \{\delta\} \cup \overline{\Sigma}^{k-1}$
        **end if**
        $\overline{\mathcal{T}}^k(q_i, \sigma) := q_{i-j}^t$.
        $\overline{\mathcal{T}}^k(q_{i-j}^t, \delta) := q_j$.
      **else**
        **if** $q_j \in \mathcal{Q}_{\mathcal{N}}^k \cup \mathcal{Q}_{\mathcal{F}_s}^k$ **then**
          Enqueue $q_j$ onto $\mathcal{L}$
        **end if**
        $\overline{\mathcal{T}}^k(q_i, \sigma) := q_j$.
      **end if**
    **end switch**
  **end for**
  **evaluate_discernability_between_successors**().
**end while**

## 4.4 Two-tanks sewer network example

### 4.4.1 Initialization

Consider the two-tanks sewer network described in Chapter 3. $HA_{init}$ is incrementally built according to Algorithm 4.2 as shown in Fig. 4.4. The composition includes also successor modes of mode $q_9$, due to $q_1$ and $q_9$ being non-discernible (see Table 3.7).

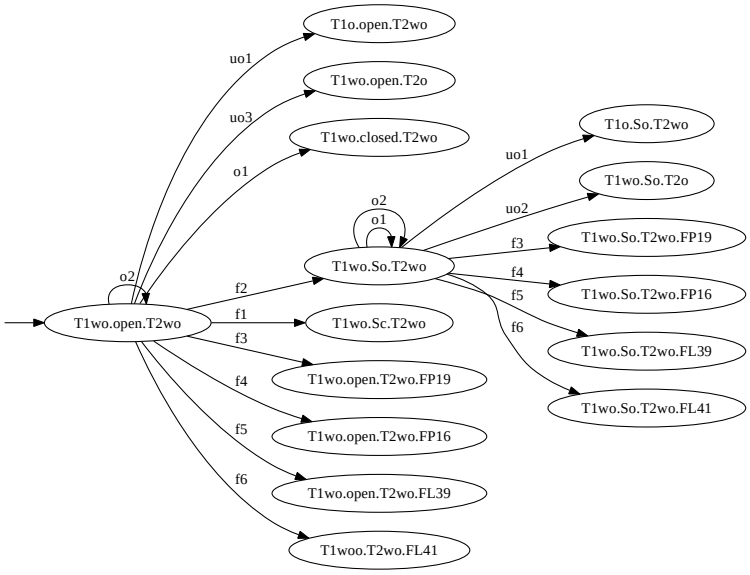


Figure 4.4: Initial incremental hybrid automaton $HA_{init}$

Next $B_{init}$, is incrementally built according to Algorithm 3.1 as shown in Fig. 4.5. Notice that $B_{init}$ includes feasible events that may occur. These events are $\delta_{13}$, $\delta_{14}$, $\delta_{12}$, $\delta_{\mathcal{F}^1_{\nu_1}}$, $\delta_{\mathcal{F}^2_{\nu_1}}$, $\delta_{\mathcal{F}^3_{\nu_1}}$, $\sigma_{o_1}$ and $\sigma_{o_2}$. Finally, the corresponding diagnoser $D_{init}$ is provided in Fig. 4.6.

### 4.4.2 Parametrized equations

The continuous dynamics corresponding to every tank represented by means of nonlinearities functions (4.3)-(4.4), are given by:

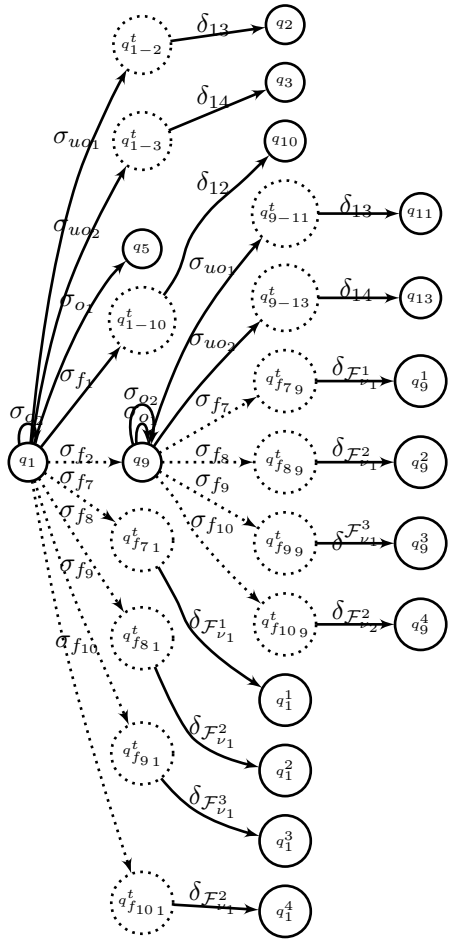


Figure 4.5: Initial incremental $B_{init}$

$$
\begin{aligned}
v_1(k+1) &= v_1(k) + \Delta t \left( \varrho_1^{in}(k) - \beta_1 S_1^1(v_1(k)) - \frac{D_1^1(v_1(k)) - \overline{v}_1}{\Delta t} \right) \\
v_2(k+1) &= v_2(k) + \Delta t \left( \varrho_2^{in}(k) - \beta_2 S_2^1(v_2(k)) - \frac{D_2^1(v_2(k)) - \overline{v}_2}{\Delta t} \right) \\
L_{39}(k) &= \frac{\beta_1}{M_{39}} S_1^1(v_1(k)) \\
L_{41}(k) &= \frac{\beta_2}{M_{47}} S_2^1(v_2(k))
\end{aligned}
$$

$$
\varrho_i^{in} = \varrho_i^{pluv_j}(k) + \beta_h \sum_{h=1}^{H} S_i^h(v_h(k)) + \sum_{l=1}^{L} \frac{D_i^l(v_l(k)) - \overline{v}_l}{\Delta t}
$$

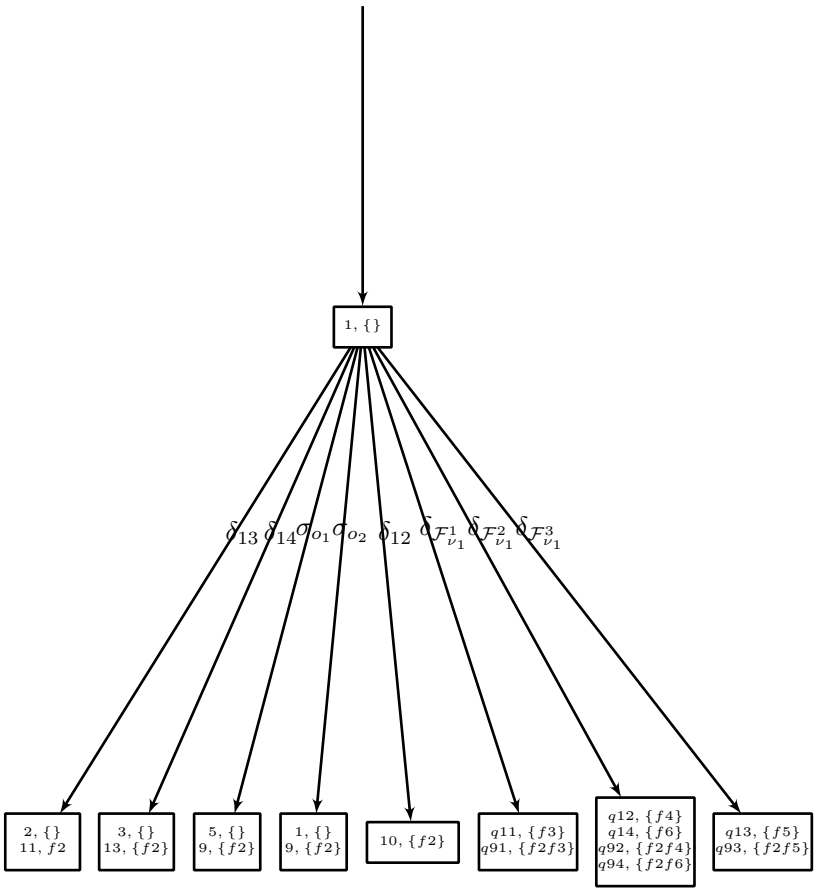


Figure 4.6: Initial incremental $D_{init}$

Notice that tank overflow nonlinearity can be represented by a dead zone function (4.4), whereas tank output flow equation can be described by a saturation function (4.3). In this case, the dead zone and saturation nonlinearities only depend on the tank volume. Given the system configuration, through this parametrization, a general model is obtained such that when a mode change is detected new modes are generated, and the model is properly instantiated. This is an offline procedure that is run once during the diagnosis process.

The input flows for every tank are as follows:

$$\begin{aligned} q_1^{in}(k) &= S_1\varphi_1 P_{19}(k) \\ q_2^{in}(k) &= S_2\varphi_2 P_{16}(k) + (1-\alpha_1)S_2^1(v_1(k)) \end{aligned}$$

The continuous state vector is given by:

$$\begin{bmatrix} v_1(k) \\ v_2(k) \end{bmatrix}$$

while the input vector is given by:

$$\begin{bmatrix} P_{19}(k) \\ P_{20}(k) \end{bmatrix}$$

### 4.4.3 Simulation scenario

Consider the same mode sequence $q_1 -> q_3 -> q_1 -> q_5$ described in the two-tank sewer network example in Chapter 3. The diagnoser tracks the mode sequence and detects and isolates faults. On the other hand, the traces of the incremental hybrid automaton, behavior automaton and diagnoser are built. The diagnoser report is provided in Table 4.1. In addition to the report described in Table 3.10, the number of diagnoser states and residuals generated at every iteration are also included to illustrate the benefit of the incremental hybrid diagnoser methodology.

The set of residuals are provided in Fig. 3.10. The remaining information, comprising continuous dynamics, set of non-discernible modes, and fault signature matrices are described in the two-tank sewer network example in Chapter 3.

| Mode change | Reported event | State diagnoser (total number of states ) | Generated residuals | Occurrence time (s) | Detection time (s) |
|---|---|---|---|---|---|
| $q_1 \rightarrow q_3$ | $\delta_{14}$ | $(q_3, \{\}), (q_{13}, \{f_2\})$ (7) | 8 | 3000 | 3300 |
| $q_3 \rightarrow q_1$ | $\delta_{41}$ | $(q_1, \{\}), (q_9, \{f_2\})$ (13) | 10 | 4200 | 4500 |
| $q_1 \rightarrow q_5$ | $\sigma_{o_1}$ | $(q_5, \{\}), (q_9, \{f_2\})$ (23) | 10 | 6900 | 6900 |
| $q_5 \rightarrow q_5^3$ fault $f_7 \in \mathcal{F}_{ns}$ in Mode $q_5$ | $\delta_{\mathcal{F}_{\nu_4}^1}$ | $(q_5^1, \{f_7\}), (q_5^2, \{f_9\})$ (23) | 10 | 9000 | 9300 |
| Total | | 23 | 10 | | |

Table 4.1: Hybrid diagnoser report

The resulting diagnoser corresponding to this simulation scenario is illustrated in Fig. 4.7. The figure shows several iteration steps of the hybrid diagnoser corresponding to its online incremental execution. The figure represents the incremental diagnoser built up to this time. The set of consistency indicators for the visited modes are generated and stored once in memory, thus avoiding duplication of procedures. Transitions and states in red describe the trajectory followed by the diagnoser. States and transitions in green are the generated states and events stored in memory representing the successors of the visited states (in red). States in blue correspond to the new feasible successor states generated at the current

moment. Transitions in blue correspond to the events that can occur at anytime before the automaton model have to be updated.

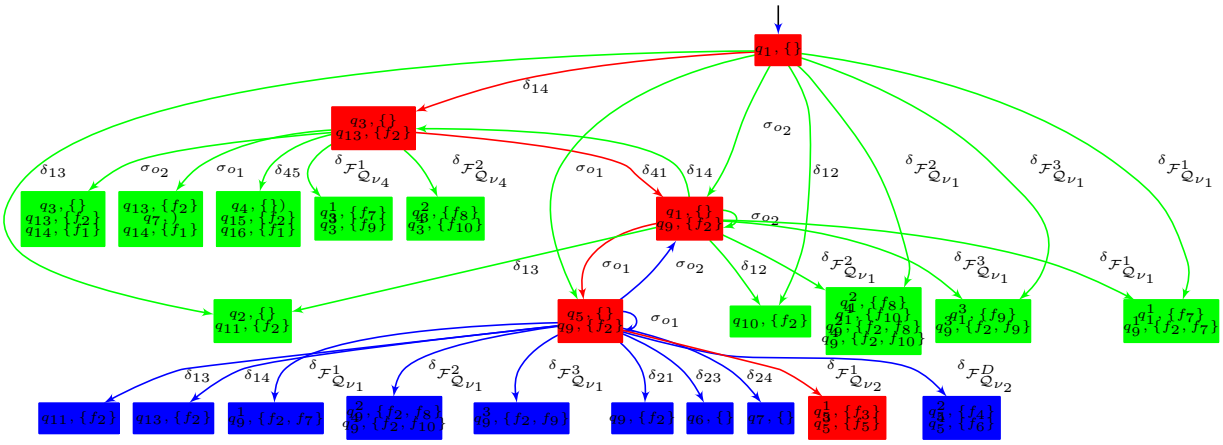

Figure 4.7: Hybrid diagnoser obtained following the incremental method

To illustrate the incremental building procedure, consider the case when event $\delta_{14}$ is identified, which corresponds to a transition from $q_1 \rightarrow q_3$ of $HA^k$. Fig. 4.8 shows a view of the full hybrid automaton with respect to the incremental part generated at $k = 3300s$. Successor modes of $q_{13}$ and $q_{14}$ are included in the incremental $HA^k$ at $k = 3300$ since $q_3$, $q_{13}$ and $q_{14}$ belong to $\mathcal{Q}_{\nu_4}$. Notice that some of the successors modes of $q_3$ have been previously generated represented in green color. However, their transitions are represented in blue color to indicate that their corresponding events must be taken into account for the diagnoser as feasible events at anytime.

The feasible events that may occur are $\delta_{45}$, $\delta_{41}$, $\delta_{12}$, $\delta_{\mathcal{F}^1_{\nu_4}}$, $\delta_{\mathcal{F}^2_{\nu_4}}$, $\sigma_{o_1}$ and $\sigma_{o_2}$. The incremental behavior automaton $B^k$ and diagnoser $D^k$ are built at $k = 3300$ and are shown in Figs. 4.9 and 4.10, respectively. They are built from the $HA^k$ shown in Fig. 4.8.

### 4.4.4 Discussion

In the online diagnosis process, all blocks shown in Fig. 4.1 cooperate. The diagnoser operates in some state $q_D$ and waits until the occurrence of an event. Two hypothesis may be derived when a mode change occurs: an observable discrete event of the $HA^k$ occurs or an unobservable event occurs. Once the event is identified, $HA^k$ and $B^k$ are updated following Algorithms 4.2 and 4.4.

The advantage of only generating an incremental part of the diagnoser is to reduce the number of modes to be handled in the online diagnosis procedure in comparison to the number of modes generated

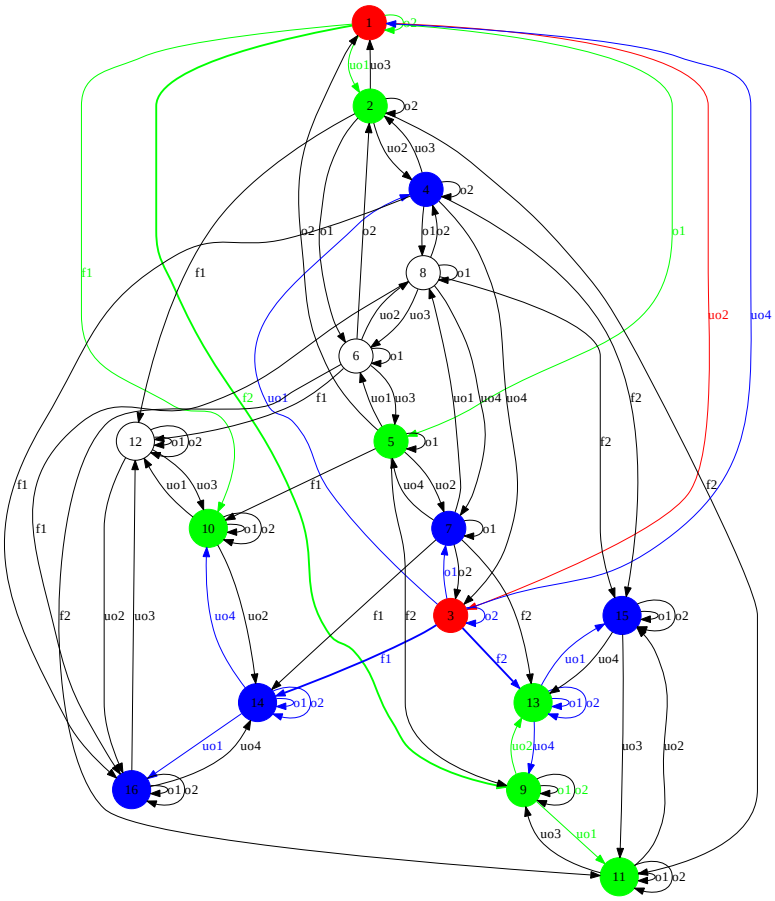


Figure 4.8: Incremental $HA(k)$ model at $k = 3300s$

in the global model. The incremental diagnoser building algorithm has a polynomial complexity ($|\mathcal{Q}_D^k| = n_q$). The depth of the exploration depends on the discernibility property between the current mode and their successors. Summarizing, the incremental method reduces the online memory usage but increments the online execution time, whereas the non-incremental method implementation might unaffordable due to online space requirements.

From a practical point of view, controlled systems are generally designed so that the control compensates for the faults that may occur, and reconfiguration policies are applied. This means that, although the number of possible modes may be theoretically high, the system really operates in a limited subset of these modes. Our incremental diagnoser method adapts to the actual operational life of the system and does not waste ressources in considering all the theoretical mode space.

Table 4.2 provides a comparison of the results obtained with the present method and the obtained in Chapter 3 with the complete offline diagnoser generation, stating out the benefits of the proposed method.

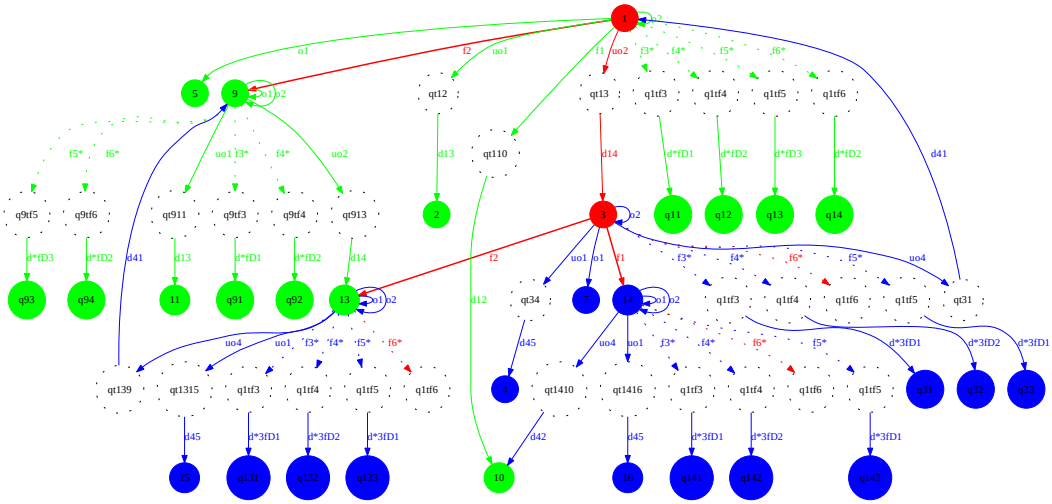

Figure 4.9: Incremental $BA(k)$ at $k = 3300s$

| | Non-incremental | Incremental |
|---|---|---|
| Number of diagnoser states | 75 | 23 per iteration |
| Number of residuals generated | 10 | 10 per iteration |
| Computational complexity | Exponential $(2^{N_{states_D}})$<br><br>$N_{states_D}$ Total number of diagnoser state | Polynomial $(N_{Succs_D(q_D)})$<br><br>$N_{Succs_D(q_D)}$ Total number of successors |

Table 4.2: Comparison between both methods according to the same simulation scenario

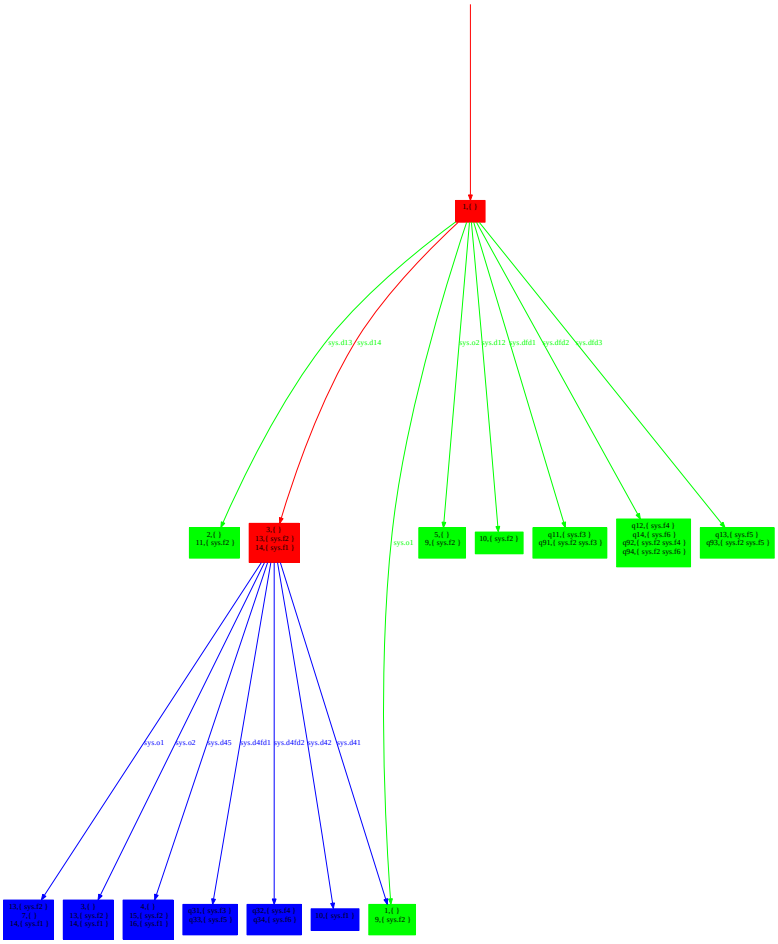


Figure 4.10: Incremental $D(k)$ at $k = 3300s$

# CHAPTER 5

# APPLICATION CASE STUDY

## 5.1 Study case description

To illustrate and validate the methodology proposed in this thesis, a representative part of the Barcelona sewer network presented in [Meseguer et al., 2010a] is used. In general, sewers are pipelines that collect and transport wastewater from city buildings and rain drains to treatment facilities before being released to the sea. Sewers are generally gravity operated, though pumps may be used if necessary [Ocampo, 2007, Ocampo and Puig, 2009].

The city of Barcelona has a combined sewer system (waste and rainwater go into the same sewer) of approximately 1500 Km. Additionally, the yearly rainfall is not very high (600 mm/year), but it includes storms typical of the Mediterranean climate that cause a lot of flooding problems and combined sewer overflows to the sea that cause pollution. Such a complex system is conducted through a control center in CLABSA (Barcelona Sewer Company) using a remote control system (in operation since 1994) that includes sensors, regulators, remote stations and communications. Nowadays, the urban drainage system contains 21 pumping stations, 36 gates, 10 valves and 10 retention tanks which are regulated in order to prevent flooding and combined sewer overflow to the environment. The remote control system is equipped with 56 remote stations including 22 rain-gauges and 136 water-level sensors which provide real-time information about rainfall and water level into the sewer system. All this information is centralized at the CLABSA Control Center through a supervisory control and data acquisition (SCADA) system.

Fig. 5.1 shows different regions of a representative part of the Barcelona sewer network that covers a surface of 22,6 $Km^2$ which will be used to illustrate the methodology. It comprises 3 redirection gates, 1 retention gate, 10 level sensors (limnimeters) and 4 rain-gauges (pluviometers) and 1 retention tank. There are two wastewater treatment plants (labeled with $WWTP1$ and $WWTP2$ in Fig. 5.2). A wastewater treatment plant consists in plants where, through physicochemical and biological processes, organic matter, bacteria, viruses and solids are removed from wastewaters before they are discharged in

rivers, lakes and seas. Nowadays the inclusion of such elements within the sewer networks is of great significance in order to preserve the ecosystem and maintain the environmental balance inside the water cycle.

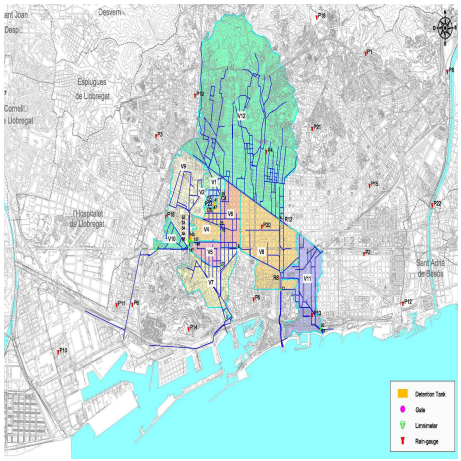


Figure 5.1: Different regions of the Barcelona city sewer network

The water flows in sewers by gravity since runoff is open-channel. Hence, it is accurately modeled by the Saint-Venant equations based on physical principles of mass conservation and energy. However, these equations are useful for offline operations (calibration and simulation ) of the sewer network not for online diagnosis. Alternatively, sewer networks may be modeled using the virtual tank modeling approach. Fig. 5.2 shows the model of the considered part of the Barcelona network using the virtual tank modeling approach [Cembrano et al., 2004]. Therefore, the decomposition of the sewer network into catchments is shown in Fig. 5.2. The elements that appear in the sewer are: nine virtual tanks, one retention tank, three redirection gates, one retention gate, four rain guages to measure the rain intensity and ten limnimeters to measure the sewer level. The control gates are opened or closed by a controller depending on the flow in the sewer.

Sewer networks present several elements exhibiting numerous operating modes depending on the sewer flows so they behave as a hybrid system. When the maximum level is reached an overflow situation occurs [Ocampo and Puig, 2009].

## 5.2 Hybrid modeling

A hybrid automaton model can be obtained to represent the hybrid phenomena concerning virtual tanks and control gates. The complete hybrid system model will be obtained by a composition procedure of

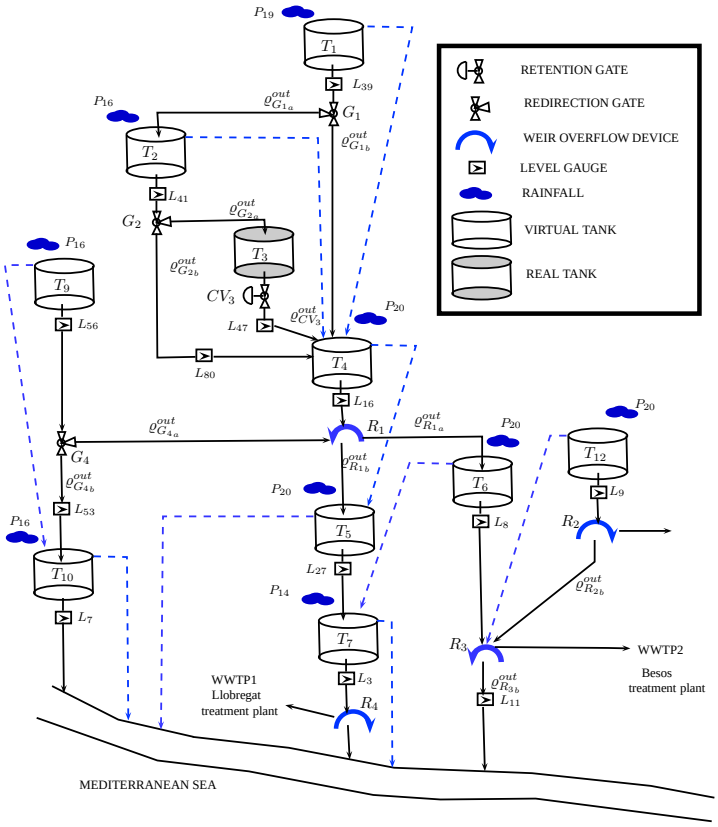


Figure 5.2: A representative part of the sewer network

component automata. Except for the wastewater treatment plants that have not taken into account.

Combining the virtual tanks approach and sewer network element automata, the following elementary models are introduced:

**Virtual tanks:** A virtual reservoir is a conceptual model of the sewer network catchment that approximates the hydraulics of the rain retention, runoff and sewage water.

The dynamic model of a virtual tank, assuming that behaves linearly, is given by the following discrete-time equation representing the water volume time evolution:

$$T_i : \qquad v_i(k+1) = v_i(k) + \Delta t(\varrho_i^{in}(k) - \varrho_i^{out}(k) - \varrho_i^{des}(k))$$

with $i \in \{0, 10\}$. The overflow is given by:

$$\varrho_i^{des}(k) = \begin{cases} \varrho_i^{in}(k) - \varrho_i^{out}(k) & \text{if } v_i(k) \geq \overline{v}_i \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

The input flow associated with a virtual tank is given by:

$$\varrho_i^{in} = \varrho_i^{pluv}(k) + \sum_{h=1}^{H} \varrho_i^{out_h}(k) + \sum_{l=1}^{L} \varrho_i^{des_l}(k) \tag{5.2}$$

where $\varrho_i^{pluv}(k) = S_i \gamma_i u_i(k)$ is associated with the rain intensity, $\varrho_i^{out_h}(k)$ corresponds to all output flows of other tanks coming in to tank $T_i$ and $\varrho_i^{des_l}(k)$ corresponds to all overflow coming in to the tank $T_i$ and $h, l \in \mathcal{Z}^+$.

The output flow for a given tank $i$ is given by:

$$\varrho_i^{out}(k) = \begin{cases} \beta_i v_i(k) & \text{if } \varrho_i^{in}(k) < \varrho_i^{out}(k) \\ \beta_i \overline{v}_i & \text{if } v_i(k) \geq \overline{v}_i \end{cases} \tag{5.3}$$

The relation between level and volume, derived from a relation between flow and volume, is given by:

$$L_i(k) = \frac{\beta_i}{M_i} v_i(k) \tag{5.4}$$

The general automaton for a virtual tank involves two discrete states: overflow ($o$) and non overflow ($wo$) situation as shown in Fig. 5.3.

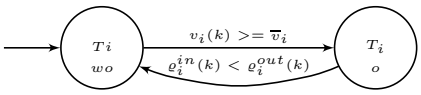


Figure 5.3: Automaton of a virtual tank $T_i$

The list of events for all virtual tank automata are described in Table 5.1. All these events are unobservable and spontaneous.

The virtual tank parameters are described in Table 5.2

**Gates**: In a real tank, a retention gate controls the outflow. Virtual tank outflows can not be closed but can be redirected using redirection gates. Redirection gates divert the flow from a nominal flow path it follows when the redirection gate is closed. The control gate input flow is divided in two output flows

| $T_i$ | Unobservable spontaneous events | label event | code event |
|---|---|---|---|
| $T_1$ | $v_1(k) \geq \overline{v}_1$ | $\sigma_{uo2}$ | 1 |
| | $q_1^{in}(k) < q_1^{out}(k)$ | $\sigma_{uo1}$ | 2 |
| $T_2$ | $v_2(k) \geq \overline{v}_2$ | $\sigma_{uo4}$ | 3 |
| | $q_2^{in}(k) < q_2^{out}(k)$ | $\sigma_{uo3}$ | 4 |
| $T_4$ | $v_4(k) \geq \overline{v}_4$ | $\sigma_{uo6}$ | 5 |
| | $q_4^{in}(k) < q_4^{out}(k)$ | $\sigma_{uo5}$ | 6 |
| $T_5$ | $v_5(k) \geq \overline{v}_5$ | $\sigma_{uo8}$ | 7 |
| | $q_5^{in}(k) < q_5^{out}(k)$ | $\sigma_{uo7}$ | 8 |
| $T_6$ | $v_6(k) \geq \overline{v}_6$ | $\sigma_{uo10}$ | 9 |
| | $q_6^{in}(k) < q_6^{out}(k)$ | $\sigma_{uo9}$ | 10 |
| $T_7$ | $v_7(k) \geq \overline{v}_7$ | $\sigma_{uo12}$ | 11 |
| | $q_7^{in}(k) < q_7^{out}(k)$ | $\sigma_{uo11}$ | 12 |
| $T_{12}$ | $v_{12}(k) \geq \overline{v}_{12}$ | $\sigma_{uo14}$ | 13 |
| | $q_{12}^{in}(k) < q_{12}^{out}(k)$ | $\sigma_{uo13}$ | 14 |
| $T_9$ | $v_9(k) \geq \overline{v}_9$ | $\sigma_{uo16}$ | 15 |
| | $q_9^{in}(k) < q_9^{out}(k)$ | $\sigma_{uo15}$ | 16 |
| $T_{10}$ | $v_{10}(k) \geq \overline{v}_{10}$ | $\sigma_{uo18}$ | 17 |
| | $q_{10}^{in}(k) < q_{10}^{out}(k)$ | $\sigma_{uo17}$ | 18 |

Table 5.1: List of events for virtual tank automata

| Parameter | description | units (MKS) |
|---|---|---|
| $\beta_i$ | Volume to flow conversion factor of external tank $T_i$ | $\frac{l}{s}$ |
| $M_i$ | Conversion factor in the output valve in $T_i$ | - |
| $S_i$ | Area of virtual tank $T_i$ | $m^2$ |
| $\gamma_i$ | Absorption factor of tank $T_i$ | - |
| $\overline{v_i}$ | Maximum volume in tank $T_i$ | $m^3$ |

Table 5.2: Virtual tank parameters

as can be seen in Fig. 5.4.

Output flows are defined according to the following equations:

$$\begin{cases} \varrho_{aG_j}(k) = (1-\alpha_j)\varrho_{G_j}^{in}(k) \\ \varrho_{bG_j}(k) = \alpha_j \varrho_{G_j}^{in}(k) \end{cases} \tag{5.5}$$

where $\alpha_j$ belongs to the interval $[0, 1]$. $0$ means that the retention gate is completely closed (i.e. all input flow is sent to $\varrho_{aG_j}(k)$), and $1$ means that the retention gate is completely open ( i.e. all input flow is sent to $\varrho_{bG_j}(k)$).

Control gates, are described by four discrete states: the nominal behaviors (open or closed) and the faulty behaviors (stuck open $(So)$ or stuck closed $(Sc)$) shown in Fig. 5.5.

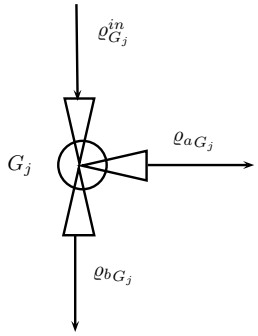


Figure 5.4: Control gate flows

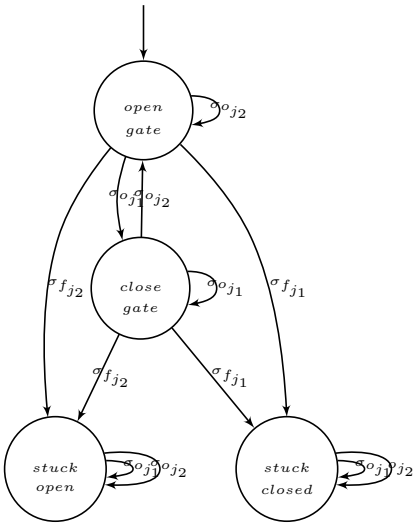


Figure 5.5: Automaton of a control gate $G_j$

The list of events for all control gate automata are described in Table 5.3. Some of them are observable (and controllable) and the others are unobservable faulty events. These faulty events are related to structural faults since they modify the system dynamics. In order to simplify, we assume that control gates can be either completely opened or closed. Thus, intermediate positions are not considered since it would imply an infinite number of modes.

Non-structural faults corresponds to faults in input and output sensors of the system. The list of non-structural faulty events are described in Table 5.4.

| Control gate | Event | Label events | Code events |
|---|---|---|---|
| $G_1$ | close | $\sigma_{o_1}$ | 19 |
| | open | $\sigma_{o_2}$ | 20 |
| | stuck closed | $\sigma_{f_{1a}}$ | 25 |
| | stuck open | $\sigma_{f_{1b}}$ | 26 |
| $G_2$ | close | $\sigma_{o_3}$ | 21 |
| | open | $\sigma_{o_4}$ | 22 |
| | stuck closed | $\sigma_{f_{2a}}$ | 27 |
| | stuck open | $\sigma_{f_{1b}}$ | 28 |
| $G_4$ | close | $\sigma_{o_5}$ | 23 |
| | open | $\sigma_{o_6}$ | 24 |
| | stuck closed | $\sigma_{f_{3a}}$ | 29 |
| | stuck open | $\sigma_{f_{3b}}$ | 30 |

Table 5.3: List of events for control gate automata

| Virtual tank | Liminimeter | fault label | Rain gauge | fault label |
|---|---|---|---|---|
| $T_1$ | $L_{39}$ | $f_7$ | $P_{19}$ | $f_{17}$ |
| $T_2$ | $L_{41}$ | $f_8$ | $P_{16}$ | $f_{18}$ |
| $T_3$ | $L_{47}$ | $f_9$ | - | - |
| $T_4$ | $L_{16}$ | $f_{10}$ | $P_{20}$ | $f_{19}$ |
| $T_5$ | $L_{27}$ | $f_{11}$ | $P_{20}$ | $f_{19}$ |
| $T_6$ | $L_8$ | $f_{12}$ | $P_{20}$ | $f_{19}$ |
| $T_7$ | $L_3$ | $f_{13}$ | $P_{14}$ | $f_{20}$ |
| $T_{12}$ | $L_9$ | $f_{14}$ | $P_{20}$ | $f_{19}$ |
| $T_9$ | $L_{56}$ | $f_{15}$ | $P_{16}$ | $f_{18}$ |
| $T_{10}$ | $L_7$ | $f_{16}$ | $P_{16}$ | $f_{18}$ |

Table 5.4: List of non-structural fault events

## 5.3 Simulations and results

As stated in Chapter 4, the incremental methodology is more appropriate than the non-incremental approach presented in Chapter 3 for the representative part of the sewer network due to its complexity. Before online diagnosis process starts, the initial incremental hybrid system model $HA_{init}$ is built considering the initial state of all components in the sewer network under no faulty situation. Next, the initial incremental behavior automaton $B_{init}$ and the initial incremental diagnoser $D_{init}$ are built from $HA_{init}$. $D_{init}$ provides information about the events to be monitored to detect some change in the system configuration or whether a fault occurs. During the simulation, $HA^k$, $B^k$ and $D^k$ are updated whenever some event is detected.

The value of the parameters used for the simulation are collected in Table 5.5.

| tank | $\beta_i(s^{-1})$ | $M_i$ | $S_i(m^2)$ | $\gamma_i$ | $\overline{v_i}(m^3)$ |
|---|---|---|---|---|---|
| $T_1$ | $7.1 \times 10^{-4}$ | 6.2871 | 323576 | 1.03 | 16901 |
| $T_2$ | $5.1 \times 10^{-4}$ | 6.6130 | 164869 | 10.4 | 43000 |
| $T_3$ | $2.1 \times 10^{-4}$ | 11.2620 | 5076 | - | 35000 |
| $T_4$ | $5.4 \times 10^{-3}$ | 3.0602 | 15707753 | 0.51 | 26659 |
| $T_5$ | $1.2 \times 10^{-4}$ | 5.3794 | 489892 | 1.93 | 27854 |
| $T_6$ | $5.6 \times 10^{-4}$ | 3.8700 | 925437 | 0.51 | 26659 |
| $T_7$ | $3.5 \times 10^{-4}$ | 14.5963 | 1570753 | 1.30 | 79229 |
| $T_{12}$ | $5.0 \times 10^{-4}$ | 26.0610 | 11345595 | 1.00 | 293248 |
| $T_9$ | $4.1 \times 10^{-4}$ | 5.5050 | 1823194 | 0.49 | 91988 |
| $T_{10}$ | $2.1 \times 10^{-4}$ | 34.6333 | 385274 | 5.40 | 175220 |

Table 5.5: Value of the sewer network parameters

### 5.3.1 Parametrized equations of the sewer network

The set of equations describing the virtual tanks are the following:

$$v_1(k+1) = v_1(k) + \Delta t(\varrho_1^{in} - \varrho_1^{out} - \varrho_1^{des}) \quad (5.6)$$
$$v_2(k+1) = v_2(k) + \Delta t(\varrho_2^{in} - \varrho_2^{out} - \varrho_2^{des}) \quad (5.7)$$
$$v_3(k+1) = v_3(k) + \Delta t(\varrho_3^{in} - \varrho_3^{out}) \quad (5.8)$$
$$v_4(k+1) = v_4(k) + \Delta t(\varrho_4^{in} - \varrho_4^{out} - \varrho_4^{des}) \quad (5.9)$$
$$v_5(k+1) = v_5(k) + \Delta t(\varrho_5^{in} - \varrho_5^{out} - \varrho_5^{des}) \quad (5.10)$$
$$v_6(k+1) = v_6(k) + \Delta t(\varrho_6^{in} - \varrho_6^{out} - \varrho_6^{des}) \quad (5.11)$$
$$v_7(k+1) = v_7(k) + \Delta t(\varrho_7^{in} - \varrho_7^{out} - \varrho_7^{des}) \quad (5.12)$$
$$v_{12}(k+1) = v_{12}(k) + \Delta t(\varrho_{12}^{in} - \varrho_{12}^{out} - \varrho_{12}^{des}) \quad (5.13)$$
$$v_9(k+1) = v_9(k) + \Delta t(\varrho_9^{in} - \varrho_9^{out} - \varrho_9^{des}) \quad (5.14)$$
$$v_{10}(k+1) = v_{10}(k) + \Delta t(\varrho_{10}^{in} - \varrho_{10}^{out} - \varrho_{10}^{des}) \quad (5.15)$$

Virtual tank input and output flows are detailed in Table 5.6. The interconnections between components are described in Table 5.6. The input flow of a virtual tank depends on the rain intensity, an output flow of other virtual tanks or a control gate, and the overflow coming of other virtual tanks.

| tank | output sensor | input sensor | output flows coming from other components | | Parameters |
|---|---|---|---|---|---|
| | | | output flow | overflow $\varrho^{des_l}$ | |
| $T_1$ | $L_{39}$ | $P_{19}$ | - | - | $S_1$, $\varphi_{19}$, $\beta_1$, $M_{39}$, $\overline{v_1}$ |
| $T_2$ | $L_{41}$ | $P_{16}$ | $\varrho_{aG_1}(k)$ | - | $S_2$, $\varphi_{16}$, $\beta_2$, $M_{41}$, $\overline{v_2}$ |
| $T_3$ | $L_{47}$ | - | $\varrho_{aG_2}(k)$ | - | $S_3$, $\beta_3$, $M_{47}$ |
| $T_4$ | $L_{16}$ | $P_{20}$ | $\varrho_{bG_2}(k) + \varrho_{bG_1}(k)$ | $\varrho_{T_1}^{des}(k) + \varrho_{T_2}^{des}(k)$ | $S_4$, $\beta_4$,$\varphi_{20}$, $M_{16}$,$\overline{v_4}$ |
| $T_5$ | $L_{27}$ | $P_{20}$ | $\varrho_{R_{1b}}^{out}(k)$ | $\varrho_{T_4}^{des}(k)$ | $S_5$, $\beta_5$,$\varphi_{20}$, $M_{27}$,$\overline{v_5}$ |
| $T_6$ | $L_8$ | $P_{20}$ | $\varrho_{R_{1a}}^{out}(k)$ | - | $S_6$, $\beta_6$, $M_8$,$\varphi_{20}$,$\overline{v_6}$ |
| $T_7$ | $L_3$ | $P_{14}$ | $\varrho_{T_5}^{out}(k)$ | $\varrho_{T_6}^{des}(k)$ | $S_7$, $\beta_7$, $M_3$,$\varphi_{20}$,$\overline{v_7}$ |
| $T_{12}$ | $L_9$ | $P_{20}$ | - | - | $S_{12}$, $\varphi_{20}$, $\beta_{12}$, $M_9$, $\overline{v_{12}}$ |
| $T_9$ | $L_{56}$ | $P_{16}$ | - | - | $S_9$, $\varphi_{16}$, $\beta_9$, $M_{56}$, $\overline{v_9}$ |
| $T_{10}$ | $L_7$ | $P_{16}$ | $\varrho_{bG_4}(k)$ | $\varrho_{T_9}^{des}(k)$ | $S_{10}$, $\beta_{10}$, $M_7$,$\varphi_{16}$,$\overline{v_{10}}$ |

Table 5.6: Virtual tank input and output flows

The flows through gates are detailed in Table 5.7. Therefore, the dependence of the control gate flows shown in Table 5.6 for virtual tanks are described in more detail in this table. The input flow in gate $G_1$ corresponds to the output flow of virtual tank $T_1$, the input flow in gate $G_1$ corresponds to the output flow of virtual tank $T_2$ and the input flow in gate $G_4$ corresponds to the output flow of virtual tank $T_9$.

Replacing the equations described in Tables 5.6 and 5.7 and substituting dead zone and saturation nonlinearities, the system mode equations can be described in a compact form such that they depend on the state components and system parameters. The advantage to represent the system model equations in

| Redirection gates | input flow $\varrho^{in}_{G_i}(k)$ | output flows | | parameters |
|---|---|---|---|---|
| | | $\varrho^{out}_{G_{i_a}}(k)$ | $\varrho^{out}_{G_{i_b}}(k)$ | |
| $G_1$ | $\varrho^{out}_{T_1}(k)$ | $(1-\alpha_1)\varrho^{out}_{T_1}(k)$ | $(\alpha_1)\varrho^{out}_{T_1}(k)$ | $\alpha_1$ |
| $G_2$ | $\varrho^{out}_{T_2}(k)$ | $(1-\alpha_2)\varrho^{out}_{T_2}(k)$ | $(\alpha_2)\varrho^{out}_{T_2}(k)$ | $\alpha_2$ |
| $G_4$ | $\varrho^{out}_{T_9}(k)$ | $(1-\alpha_4)\varrho^{out}_{T_9}(k)$ | $(\alpha_4)\varrho^{out}_{T_9}(k)$ | $\alpha_4$ |
| Retention gate | input flow | output flow | | parameter |
| $CV_3$ | $\varrho^{out}_{T_3}(k)$ | $\beta_3\varrho^{out}_{T_3}(k)$ | | |

Table 5.7: Redirection and retention gate parameters

this form is the easy way to obtain the different dynamics for the visited modes online whenever a new mode change is detected. Besides, the dynamics remains linear for all modes. The implemented Matlab function is described in Appendix A.

The simulator of the sewer network implemented by [Ocampo and Puig, 2009], allows us to validate the methodology. Fig. 5.6 shows the implemented scheme in Matlab. Data provided by rain gauges corresponds to real episodes of rain occurred in Barcelona registered by CLABSA. The data provided by limnimeters is generated by the simulator shown in Fig. 5.6 through the rain gauge data.

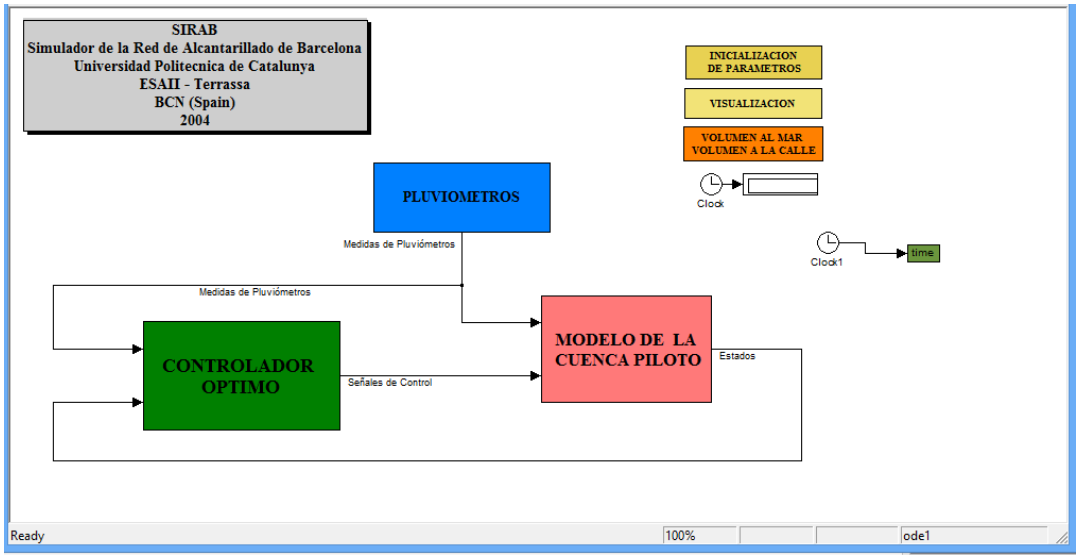


Figure 5.6: Simulator of the sewer network

## 5.4 Scenario I

System dynamics evolves following the mode sequence $q_1 - > q_3 - > q_{214} - > q_{140} - > q_{211} - > q_5 - > q_{885}$ for the rain episode occurred in Barcelona shown in Fig. 5.7 with a sampling time of $\Delta t = 300s$. Mode $q_1$ refers to the situation in which no tank is in overflow. Mode $q_3$ refers to $T_1$ being in overflow. $q_{214}$ refers to $T_2, T_4, T_5$ and $T_{12}$ being in overflow. $q_{140}$ refers to $T_2, T_4$ and $T_5$ being in overflow. Mode $q_{211}$ refers to $T_5$ and $T_4$ being in overflow and mode $q_5$ refers to $T_5$ being in overflow. Fig. 5.7 shows the rain gauge measurements for the considered rain episode.

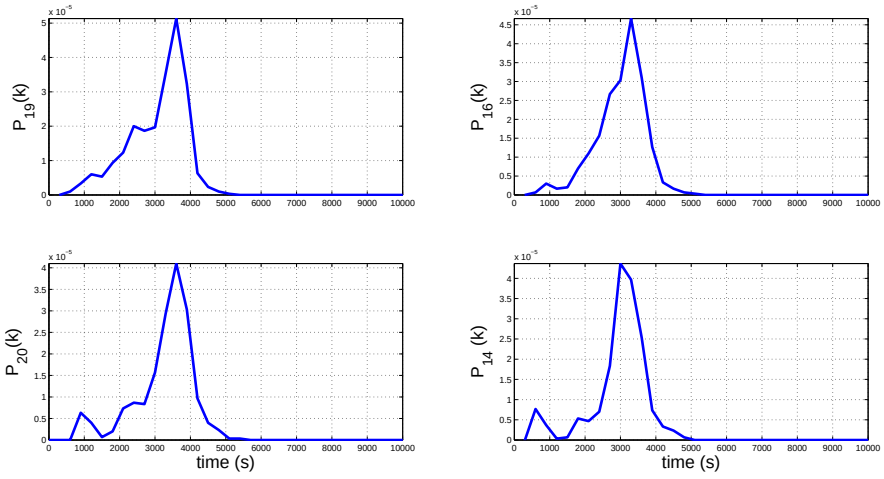


Figure 5.7: Example of a rain episode occurred in Barcelona

The measurements provided by the limnimeters are shown in Fig. 5.8. Notice that in the figure it is possible to see which tanks are in overflow. The green horizontal dashed line is the maximum level a virtual tank can reach. Therefore, the mode sequence can be deduced from system measurement.

Fig. 5.9 shows the set of residuals for the concerned modes. Remark that the residuals in all modes are consistent with measurements whenever system remains in them. The signature-events identified during the simulation are shown in vertical dashed lines in Fig. 5.9. These events are reported in Table 5.11.

Notice for instance that when the system is in mode $q_3$, $\Phi_{67}(k) = \mathbf{0}$ during the time interval $[3600s, 3900s]$ whereas the remaining consistency relations differ from zero.

Next, a non-structural fault occurs at $7800s$, that is detected by the diagnoser. The theoretical fault signature matrix to isolated this fault is provided by Table 5.8, which corresponds to mode $q_5 \in \mathcal{Q}_{\nu_{58}}$.

Fig. 5.10 plots the set of residuals for mode $q_5$. Remark that the observed signature is

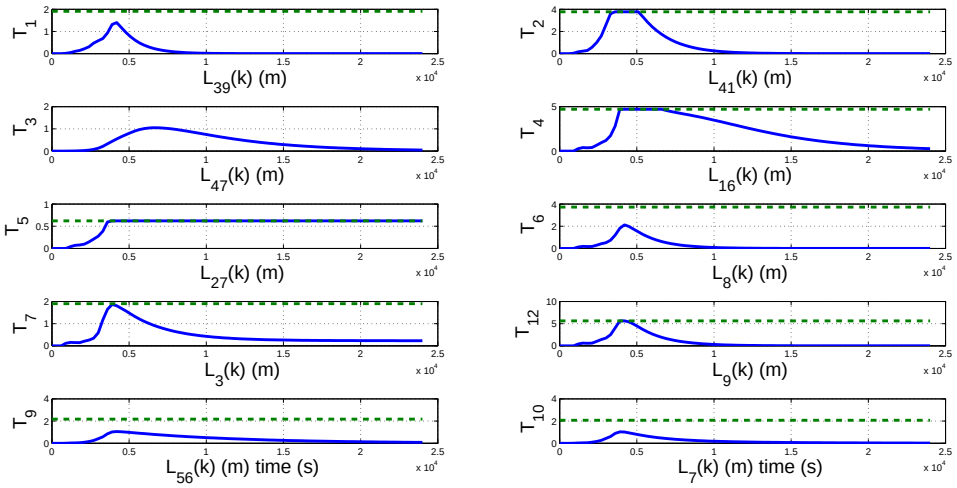


Figure 5.8: Levels provided by the limnimeters

| $f_{17}$ | $f_{18}$ | $f_{19}$ | $f_{20}$ | $f_7$ | $f_8$ | $f_9$ | $f_{10}$ | $f_{11}$ | $f_{12}$ | $f_{13}$ | $f_{14}$ | $f_{15}$ | $f_{16}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

Table 5.8: Fault signature matrix $\mathbf{FS}_{\nu_{58}}$

$[\ 0\quad 1\quad 1\quad 0\quad 0\quad 0\quad 0\quad 0\quad 0\ ]^t$ which, according to $\mathbf{FS}_{\nu_{58}}$, corresponds to a fault in sensor $L_{41}$ (i.e. the fault labeled with $f_8$ in Table 5.8).

The report given by the hybrid diagnoser is shown in Table 5.11. 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.

Figs. 5.11 and 5.12 show a part of $HA^k$ and $B^k$ in more detail. In red the visited modes and their

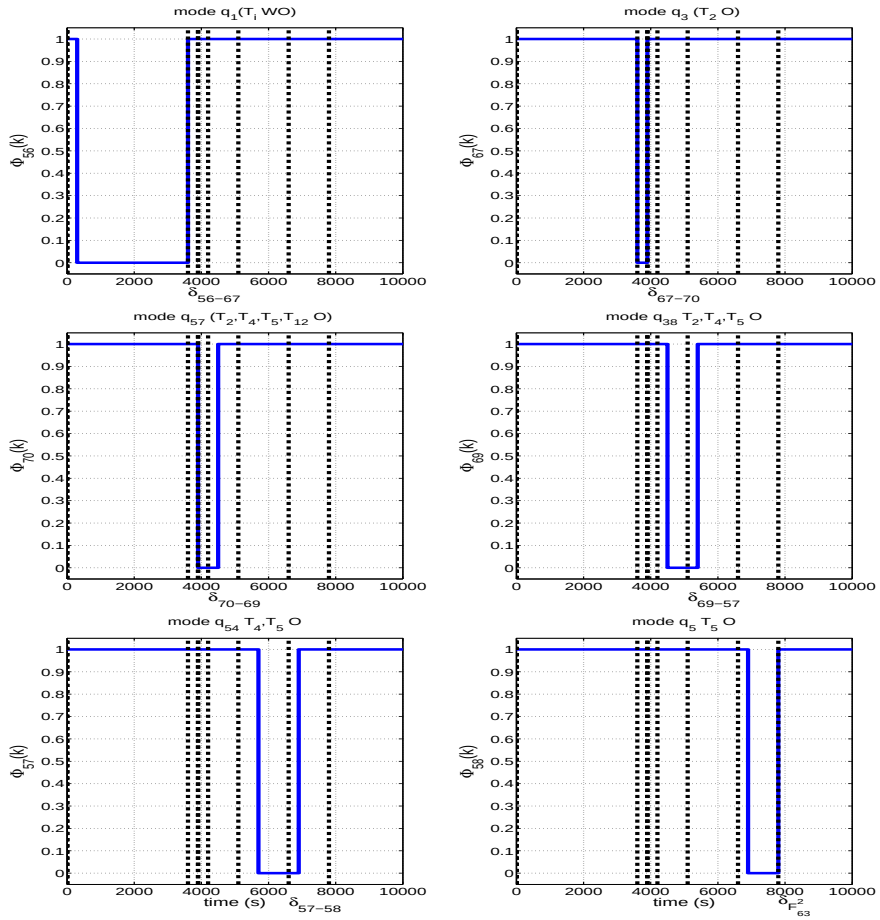


Figure 5.9: Binary residuals

| Mode change | Reported event | Current diagnoser state | Occurrence time (s) | Detection time (s) |
|---|---|---|---|---|
| Initial mode $q_1$ | - | $(q_1, \{\})$ | - | - |
| $q_1 \rightarrow q_3$ | $\delta_{56-67}$ | $q_3, \{\}q_{21}, \{f_{1b}\}q_{36}, \{f_{2b}\}q_{50}, \{f_{3b}\}$ | 3600 | 3600 |
| $q_3 \rightarrow q_{214}$ | $\delta_{67-70}$ | $q_{214}, \{\}q_{238}, \{f_{1a}\}q_{251}, \{f_{1b}\}q_{264}, \{f_{2b}\}q_{274}, \{f_{3a}\}$ | 3900 | 3900 |
| $q_{214} \rightarrow q_{140}$ | $\sigma_{70-69}$ | $q_{140}, \{\}q_{166}, \{f_{1a}\}q_{179}, \{f_{1b}\}$ | 4200 | 4500 |
| $q_{140} \rightarrow q_{211}$ | $\sigma_{69-57}$ | $q_{211}, \{\}q_{248}, \{f_{1b}\}q_{261}, \{f_{2b}\}q_{271}, \{f_{3a}\}$ | 5100 | 5400 |
| $q_{211} \rightarrow q_5$ | $\sigma_{57-58}$ | $q_5, \{\}q_{23}, \{f_{1b}\}q_{38}, \{f_{2b}\}q_{52}, \{f_{3a}\}$ | 6600 | 6900 |
| $q_5 \rightarrow q_{885}$ fault in $L_{41} \in \mathcal{F}_{ns}$ | $\delta_{\mathcal{F}_{58}^2}$ | $q_{885}, \{f_8\}$ $q_{899}, \{f_{1b}f_8\}q_{913}, \{f_{2b}f_8\}q_{927}, \{f_{3a}f_8\}$ | 7800 | 7800 |

Table 5.9: Hybrid diagnoser report for Scenario I

non-discernible successor modes generated in the parallel composition are represented. The total number of modes in $HA^k$ is 697, while the number of modes in $B^k$ is 1506 and the total number of diagnoser

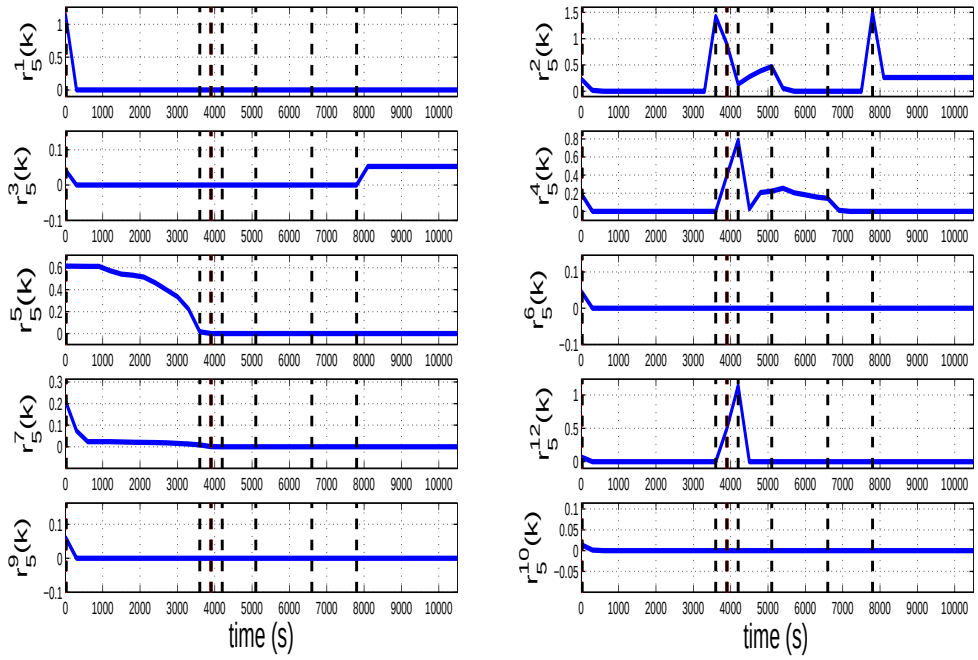


Figure 5.10: Residuals of mode $q_5$ belonging to $\mathcal{Q}_{\nu_{58}}$

states is 156. $HA^k$, $B^k$ and $D^k$ during the simulation scenario are shown in Figs. 5.13, 5.14 and 5.15 respectively [1].

[1] These figures are presented just to visualize the complexity because of the size of the problem.

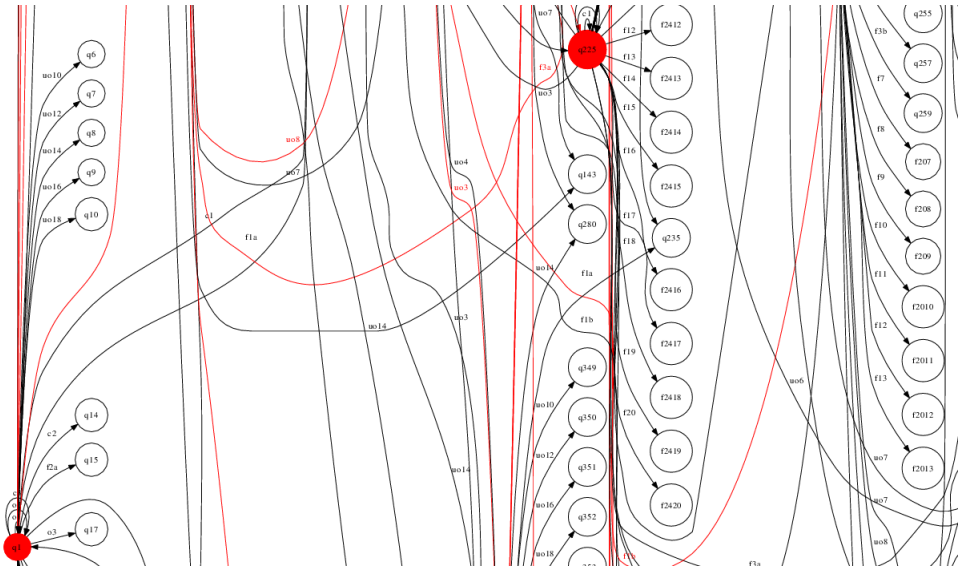

Figure 5.11: A part of $HA^k$ for the simulation Scenario I

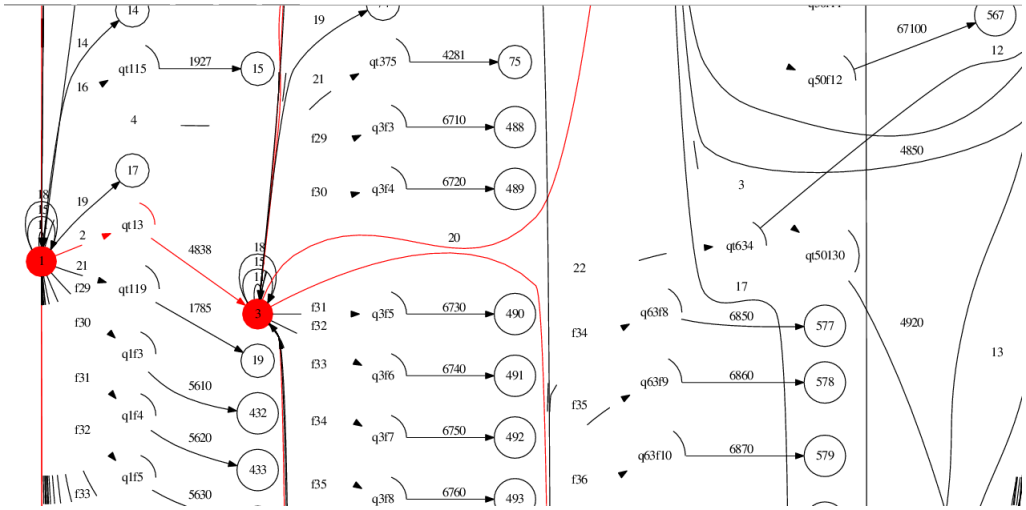

Figure 5.12: A part of $B^k$ for simulated Scenario I

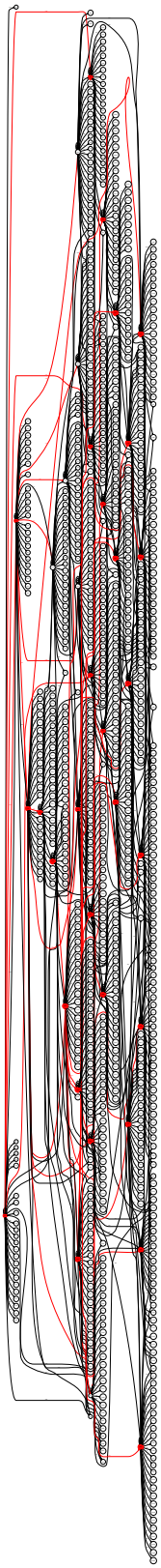

Figure 5.13: $HA^k$ for simulated Scenario I

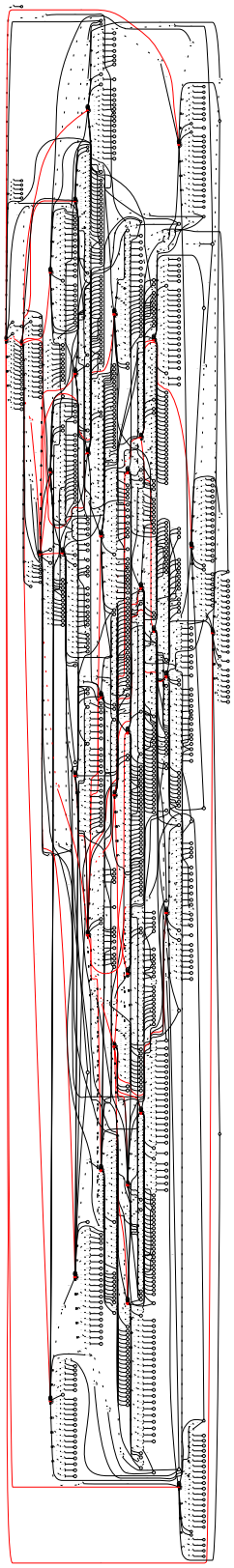

Figure 5.14: $B^k$ for simulated Scenario I

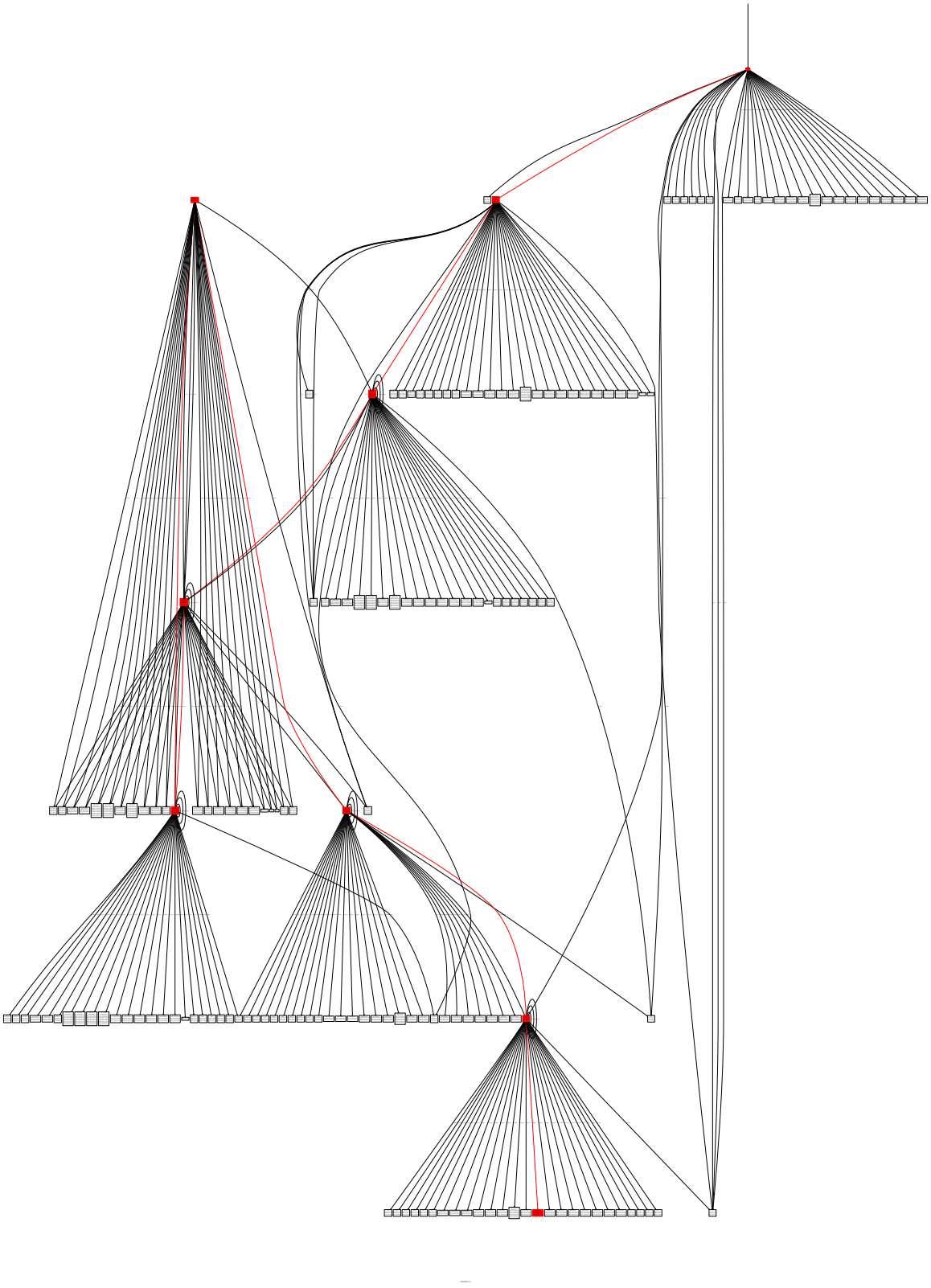

Figure 5.15: $D^k$ for the simulation Scenario I

## 5.5 Scenario II

For this scenario, the system dynamics evolves following the mode sequence $q_1 \rightarrow q_{14} \rightarrow q_{29} \rightarrow q_{47} \rightarrow q_{355}$ for the rain episode occurred in Barcelona shown in Fig. 5.16. The initial mode is $q_1$ as in the previous scenario, mode $q_{14}$ corresponds to closing gate $G_2$, mode $q_{29}$ refers to closing gate $G_1$, mode $q_{47}$ corresponds to a structural faulty mode (valve $G_1$ in stuck closed state) and $q_{355}$ is a non-structural fault in sensor $L_{39}$.

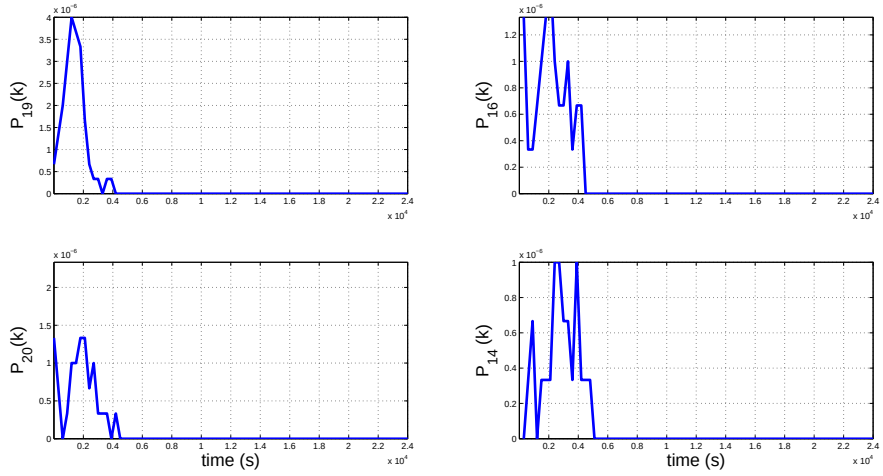


Figure 5.16: Example of a rain episode occurred in Barcelona

In Scenario I, the diagnosis was carried out when the rain caused overflow in virtual tanks. In this scenario, rain intensity does not cause overflow in virtual tanks as it can be seen in Fig. 5.17. Moreover, gate switching is simulated to test how the diagnoser behaves under observable events and structural and non-structural faults in components.

The set of consistency indicators are generated for the concerned modes (see Fig. 5.18). In this case, when an observable event occurs, the set of residuals are not used to detect the transitions between modes, but in any case they must be generated. Notice that the set of residuals are consistent with measurements when observable events occur. Gate $G_2$ stays in stuck closed position when open command is triggered. Therefore, the set of residuals for mode $q_{47}$ is equivalent to mode $q_{29}$. In this case, the set of residuals can be used to confirm a transition given by an observable event. It is possible to know that the system is in stuck closed state because the set of residuals does not change when the command is executed.

Fig. 5.19 plots the set of residuals for mode $q_{47}$. It is possible to detect and isolate the non-structural fault in liminimeter $L_{39}$ since the corresponding theoretical signature is $[\ 1\quad 0\quad 0\quad 1\quad 0\quad 0\quad 0\quad 0\quad 0\ ]^t$ (see Table 5.10) which coincides with the observed fault signature

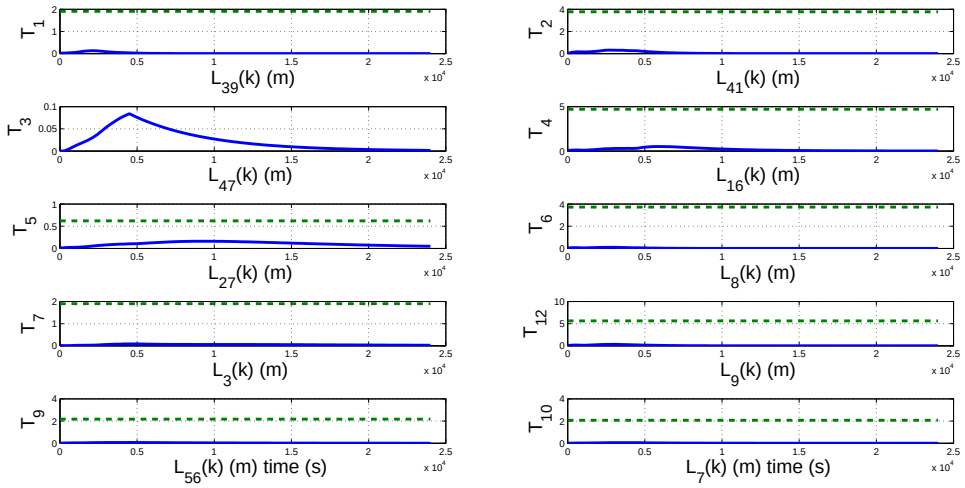


Figure 5.17: Levels provided by the limnimeters

in sensor $L_{39}$ shown in Fig. 5.19.

| $f_{17}$ | $f_{18}$ | $f_{19}$ | $f_{20}$ | $f_7$ | $f_8$ | $f_9$ | $f_{10}$ | $f_{11}$ | $f_{12}$ | $f_{13}$ | $f_{14}$ | $f_{15}$ | $f_{16}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

Table 5.10: Fault signature matrix $\mathbf{FS}_{v_{58}}$

The report given by the diagnoser for Scenario II is provided in Table 5.11. The transitions are detected instantaneously because they correspond to observable events. In the case of a non-structural fault, residuals are activated immediately after they appear. The structural fault is detected because closing valve $G_2$ does not produce any change in the set of residuals.

The total number of modes in $HA^k$ is 260, the number of modes in $B^k$ is 637 and the number of diagnoser states is 130. $HA^k$, $B^k$ and $D^k$ generated during the simulation scenario are shown in Figs.

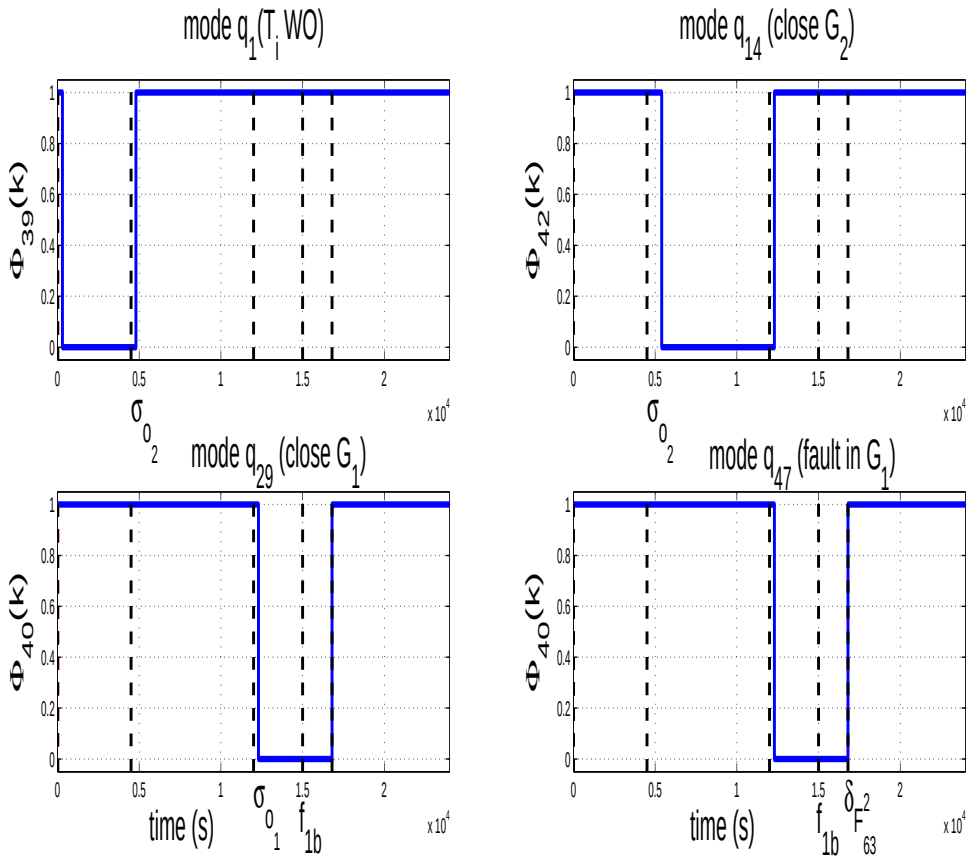


Figure 5.18: Binary residuals for the concerned modes for Scenario II

| Mode change | Reported event | Current diagnoser state | Occurrence time (s) | Detection time (s) |
|---|---|---|---|---|
| Initial mode $q_1$ | - | $(q_1, \{\})$ | - | - |
| $q_1 \rightarrow q_{14}$ | $\sigma_{o_2}$ | $q_{14}, \{\} q_{16}, \{f_{2b}\} q_{31}, \{f_{1b}\} q_{33}, \{f_{3a}\}$ | 3600 | 3600 |
| $q_{14} \rightarrow q_{29}$ | $\sigma_{o_1}$ | $q_{29}, \{\} q_{31}, \{f_{1b}\} q_{44}, \{f_{2a}\} q_{45}, \{f_{2b}\}$ $q_{47}, \{f_{3b}\}$ | 3900 | 3900 |
| $q_{29} \rightarrow q_{47}$ | $\sigma_{f_{1b}}$ | $q_{29}, \{\} q_{31}, \{f_{1b}\} q_{44}, \{f_{2a}\}$ $q_{45}, \{f_{2b}\} q_{47}, \{f_{3b}\}$ | 4200 | 4500 |
| $q_{47} \rightarrow q_{356}$ fault in $L_{39} \in \mathcal{F}_{ns}$ | $\delta_{\mathcal{F}_{40}^2}$ | $q_{219}, \{f_7\} q_{331}, \{f_{1a} f_7\} q_{345}, \{f_{2b} f_7\}$ $q_{359}, \{f_{3a} f_7\} q_{373}, \{f_{1a} f_{3a} f_7\}$ $q_{387}, \{f_{2a} f_{3a} f_7\}$ | 7800 | 7800 |

Table 5.11: Hybrid diagnoser report for Scenario II

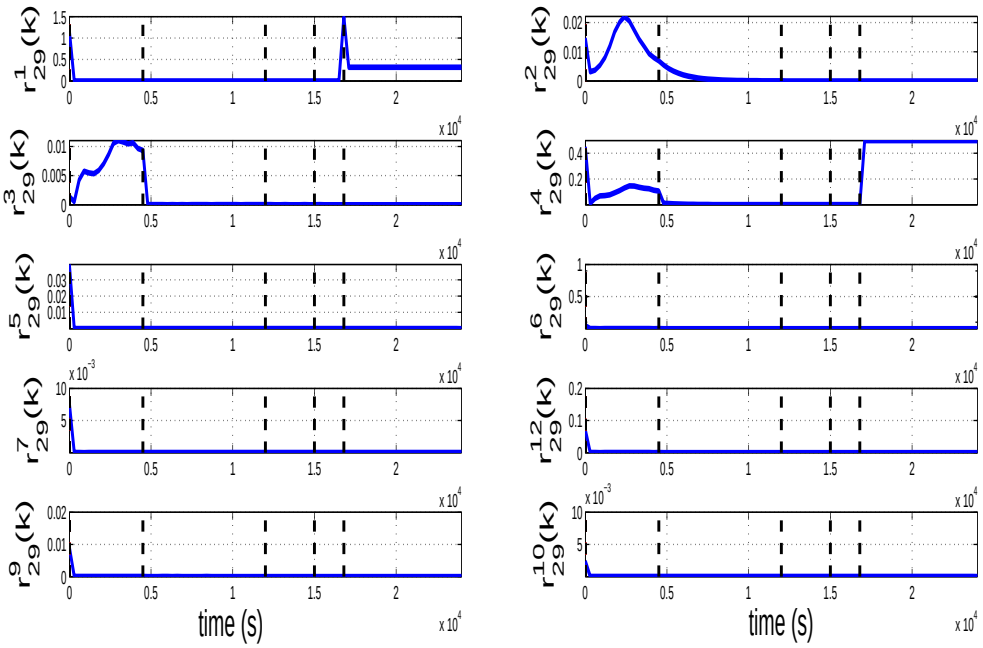


Figure 5.19: Residuals of mode $q_{47}$ belonging to $\mathcal{Q}_{\nu_1}$

5.20, 5.21 and 5.22 respectively.

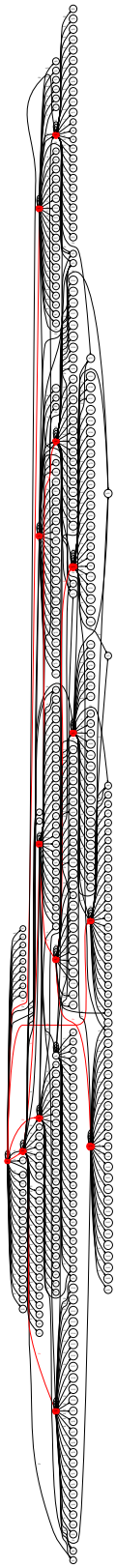

Figure 5.20: $HA^k$ for simulated Scenario II

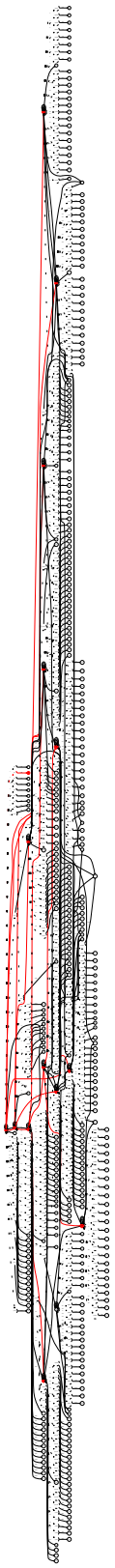

Figure 5.21: $B^k$ for simulated Scenario II

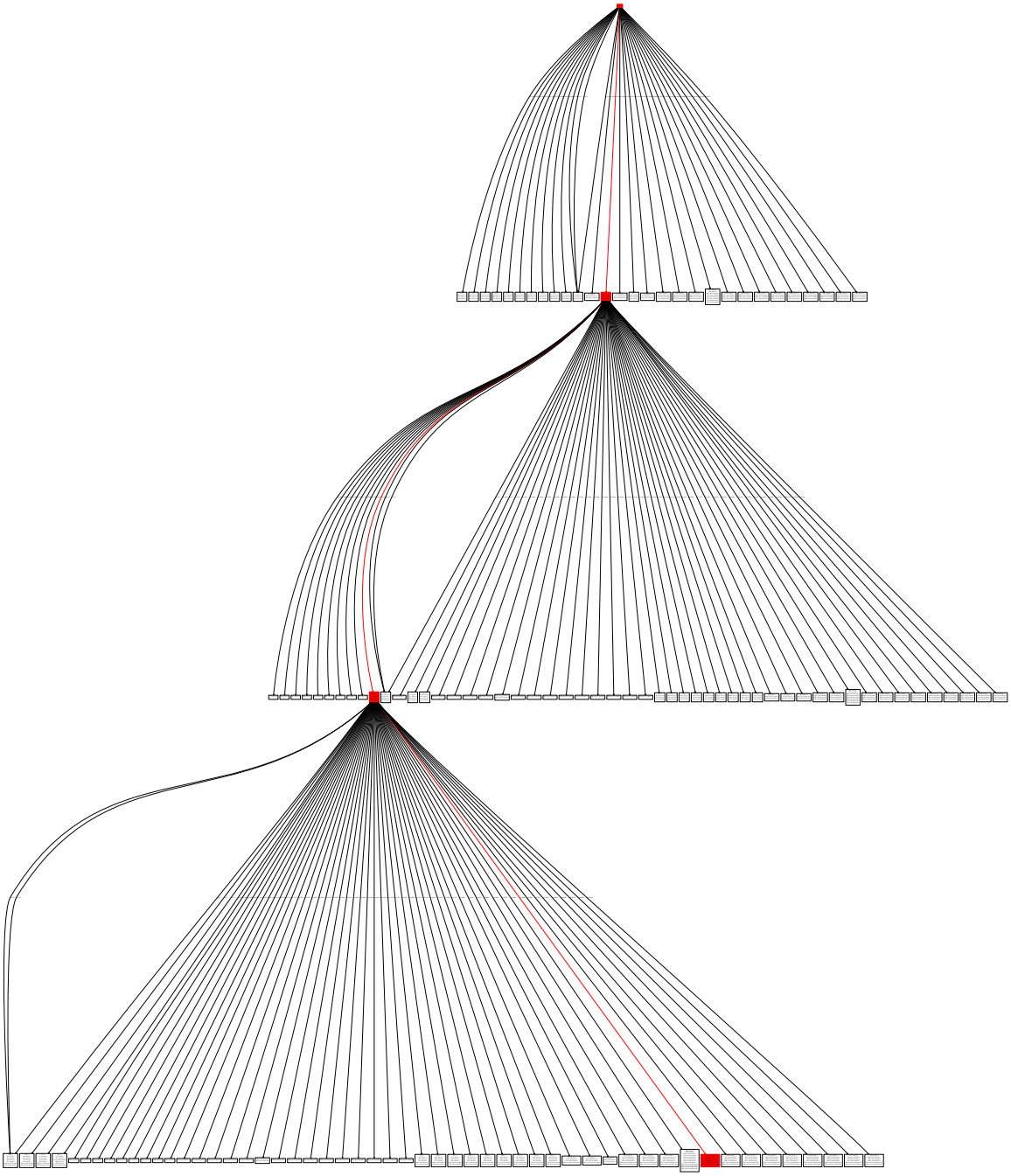

Figure 5.22: $D^k$ for simulated Scenario II

## 5.6 Results analysis

### 5.6.1 Automata composition analysis

Components automata are shown in Table 5.12, describing the number of components, the number of component discrete-states and the total number of modes if composition was carried out for the whole sewer network (i.e. considering all possible combinations among the discrete-state labels).

| Component | Num. components | Num. discrete states | Num. transitions | Num. non-struct. fault states |
|---|---|---|---|---|
| Virtual tanks | 9 | $2^{11} = 512$ | 4608 | - |
| Real tank | 1 | - | - | - |
| Output sensors | 10 | - | - | 10 |
| Input sensors | 4 | - | - | 4 |
| Redirection gates | 3 | $4^3 = 64$ | | - |
| Retention gates | 1 | - | - | - |
| Weir overflow devices | 4 | - | - | - |
| Total of discrete states | | 32768 | 589824 | 14*32768=458752 |

Table 5.12: Automata composition analysis for the sewer network

Applying the methodology presented in Chapter 3, the number of modes to be generated is 32768 modes, the number of residuals to be computed is in the worst case 327680 residuals. Algorithm 3.1 would require an exponential time to build $B$ and $D$. Hence, it is not possible to easily obtain an implementable hybrid automaton model. Even if binary residuals of the active groups (the current mode and their successors) were only computed in the online diagnosis process, the computational cost would be high since the algorithm to update active groups requires an exploration of the full diagnoser model. Alternatively, incremental diagnosis deals with this problem, avoiding to build the entire hybrid model and automaton.

### 5.6.2 Complexity analysis and benefits of the methodology

To study the complexity of the proposed methodology, results from Scenario I will be used. Table 5.13 reports the space and time complexity for this scenario. Unlike the non-incremental method, the number of visited modes increases if a mode change in the system is detected. Notice that one of the main advantages is that the number of modes increases slowly from one iteration step to another. In case of transitions to already visited modes in the system, the knowledge-base does not need to be updated. Hence , update is not executed in the algorithms that generate $HA^k$, $B^k$ and $D^k$. Similarly occurs, in the case of non-structural faulty mode transitions.

If full mode discernibility in $HA$ was guaranteed, the number of diagnoser states would tend to the number of modes $|\mathcal{Q}_D| \approx |\mathcal{Q}|$, depending on the depth of the $HA^k$ exploration. Time complexity

| Mode change | $\Phi^k$ ,$\mathcal{Q}_\nu^k$ | $HA^k$ $\lvert\mathcal{Q}_\mathcal{N} \cup \mathcal{Q}_{\mathcal{F}_s}\rvert + \lvert\mathcal{Q}_{\mathcal{F}_s}\rvert$ | $B^k$ $\lvert\overline{\mathcal{Q}}\rvert((\lvert\overline{T}\rvert))$ | $D^k$ $\lvert\mathcal{Q}_D\rvert$ |
|---|---|---|---|---|
| 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 $f_7 \in \mathcal{F}_{ns}$ in Mode $q_5$ | 690 , 69 | 0 | 0 | 156 |
| Total | 690 ,69 | $431 + 266 = 697$ | 1506 (1781) | 156 |

Table 5.13: Sewer network complexity results for Scenario I

increases since $HA^k$, $B^k$ and $D^k$ are computed at the same time. To guarantee that a transition can be detected, residual dynamics should be faster than the system (300s in the sewer network application). The algorithms are executed in an acceptable time. Time complexity increases but is much less than the sampling time. The set of residuals to be computed depend on the active set of non-discernible modes, which varies between 130 and 200.

Optimal diagnosis performance depends on weather conditions. Under no rain condition, water level remains stable and transitions between modes do not vary faster than in a rain episode. Besides, rain intensity does not imply an overflow situation as can be seen in the Scenario II. After a non-structural fault is detected, the diagnosis stops and the number of modes and states remains in a constant value.

### 5.6.3 Limitations in hybrid diagnosis

After detecting a non-structural fault, continuous dynamics must be recomputed to take into account the fault effect. Non-structural faults affect the continuous model used to generated the set of residuals. The loss of information should be compensated otherwise diagnosis would be erroneous. This is not a trivial task. It could be considered whenever a new system model have a reasonable online execution time to update it. In the methodology, there are no mechanisms to do that. Hence, until a detected fault is not repaired, the diagnoser cannot proceed.

The occurrence time between two transitions in $HA^k$ is an important aspect to be considered. The sampling time, the residuals dynamics and the observable events occurrence play an important role in hybrid diagnosis. For this reason, the methodology assumes that events can sequentially occur during the system evolution in a minimal time between them (see Assumption 3.2). This time is associated with the dwell time and the sampling time. If multiple transitions took place at the same time, some of them could not be detected (see Assumption 3.1).

The use of a binary codification implies a loss of information since the residual activation may exhibit different dynamics (slow or fast). Moreover, after a fault or mode change, the persistency of the binary residual indicator activation $(\phi_i^l(k))$ of each set of residuals is independent of the others (see Assumption 3.3).

In some cases, consistency indicators are active at different instant times. The methodology does not include a logic based on the transient response but in the permanent response. A delay is present in the diagnosis to consider settling time of dynamical residuals and is being included in the event generation block.

Another common limitation is the instability of the mode detection test indicator (chattering) since the presence of noise and the binary test used. A threshold for every residual has been experimentally chosen for the sewer network.

# CHAPTER 6

# EXTENSIONS TO THE FAULT DIAGNOSIS METHODOLOGIES FOR HYBRID SYSTEMS

The methodology to detect and isolate faults developed in previous chapters can be improved considering some aspects neglected so far as robustness and nonlinearities always present in systems and assuming fault models are known. In the case of considering uncertainty, binary residual computation is improved generating an adaptive threshold that considers model uncertainty. Regarding the issue of unknown fault models and nonlinearities, the use of structural models allows to generate residuals even if some equations are nonlinear. The logic applied to detect and isolate faults allows to make hypothesis regarding multiple fault occurrence and detect non-modeled faults using a component oriented fault diagnosis approach.

## 6.1 Hybrid system diagnosis under model uncertainty

The diagnosis method for hybrid systems, presented in this thesis, is based on generating the set of residuals by means of the parity space approach. The robustness is enhanced by using a passive strategy based on generating an adaptive threshold that considers model uncertainty in the residual evaluation extending the results for the LTI case presented in [Vento et al., 2012].

In the hybrid automaton model only the continuous dynamics are assumed to be affected by modeling uncertainty. The algorithms to build the behavior automaton $B$ and the diagnoser $D$ do not change since they do not depend on parameter uncertainty.

### 6.1.1 Continuous dynamics with model uncertainty

The continuous dynamics with parametric uncertainty concerning mode $i \in \mathcal{Q}_{\mathcal{N}} \cup \mathcal{Q}_{\mathcal{F}_s}$ are defined as follows:

$$\mathbf{x}(k+1) = \mathbf{A}_i(\widetilde{\boldsymbol{\theta}})\mathbf{x}(k) + \mathbf{B}_i(\widetilde{\boldsymbol{\theta}})\mathbf{u}(k) + \mathbf{F}_{xi}(\widetilde{\boldsymbol{\theta}})\mathbf{f}(k) + \mathbf{E}_{xi}(\widetilde{\boldsymbol{\theta}}) \tag{6.1}$$

where $\mathbf{A}_i(\widetilde{\boldsymbol{\theta}}) \in \mathcal{R}^{nx \times nx}$, $\mathbf{B}_i(\widetilde{\boldsymbol{\theta}}) \in \mathcal{R}^{nx \times nu}$ and $\mathbf{E}_{xi}(\widetilde{\boldsymbol{\theta}}) \in \mathcal{R}^{nx \times 1}$ are the state matrices in mode $i$, and $\mathbf{f}(k) \in \mathcal{R}^{nf}$ represents the system faults, with $\mathbf{F}_{xi}(\widetilde{\boldsymbol{\theta}}) \in \mathcal{R}^{nx \times nf}$ being the fault distribution matrix in mode $i$. Some of the model parameters $(\widetilde{\boldsymbol{\theta}})$ are assumed to be time-invariant and unknown but bounded by an interval set, i.e., they belong to the set $\boldsymbol{\Theta} = \{\boldsymbol{\theta} \in \mathcal{R}^{n\boldsymbol{\theta}} | \underline{\boldsymbol{\theta}} \leq \boldsymbol{\theta} \leq \overline{\boldsymbol{\theta}}\}$. This set represents the uncertainty on the exact knowledge of the real system parameters $(\widetilde{\boldsymbol{\theta}})$. Analogously, the measurement system equation can be defined as follows:

$$\mathbf{y}(k) = \mathbf{C}_i(\widetilde{\boldsymbol{\theta}})\mathbf{x}(k) + \mathbf{D}_i(\widetilde{\boldsymbol{\theta}})\mathbf{u}(k) + \mathbf{F}_{y_i}(\widetilde{\boldsymbol{\theta}})\mathbf{f}(k) + \mathbf{E}_{y_i}(\widetilde{\boldsymbol{\theta}}) \tag{6.2}$$

where $\mathbf{C}_i(\widetilde{\boldsymbol{\theta}}) \in \mathcal{R}^{ny \times nx}$, $\mathbf{D}_i(\widetilde{\boldsymbol{\theta}}) \in \mathcal{R}^{ny \times nu}$ and $\mathbf{E}_{y_i}(\widetilde{\boldsymbol{\theta}}) \in \mathcal{R}^{ny \times 1}$ are the output matrices in mode $i$, and $\mathbf{F}_{y_i}(\widetilde{\boldsymbol{\theta}}) \in \mathcal{R}^{ny \times nf}$ is the fault distribution matrix in mode $i$.

Alternatively, the model given by (6.1)- (6.2) can be expressed in input-output form using the shift $p$-operator (or delay operator) assuming zero initial conditions as follows

$$\mathbf{y}(k) = \mathbf{M}_i(p^{-1}, \widetilde{\boldsymbol{\theta}})\mathbf{u}(k) + \boldsymbol{\Upsilon}_i(p^{-1}, \widetilde{\boldsymbol{\theta}})\mathbf{f}_{ns}(k) + \mathbf{E}_{mi}(p^{-1}, \widetilde{\boldsymbol{\theta}}) \tag{6.3}$$

where:

$$\begin{aligned}
\mathbf{M}_i(p^{-1}, \widetilde{\boldsymbol{\theta}}) &= \mathbf{C}_i(\widetilde{\boldsymbol{\theta}})(p\mathbf{I} - \mathbf{A}_i(\widetilde{\boldsymbol{\theta}}))^{-1}\mathbf{B}_i(\widetilde{\boldsymbol{\theta}}) + \mathbf{D}_i(\widetilde{\boldsymbol{\theta}}) \\
\boldsymbol{\Upsilon}_i(p^{-1}, \widetilde{\boldsymbol{\theta}}) &= \mathbf{C}_i(\widetilde{\boldsymbol{\theta}})(p\mathbf{I} - \mathbf{A}_i(\widetilde{\boldsymbol{\theta}}))^{-1}\mathbf{F}_{xi}(\widetilde{\boldsymbol{\theta}}) + \mathbf{F}_{y_i}(\widetilde{\boldsymbol{\theta}}) \\
\mathbf{E}_{my_i}(p^{-1}, \widetilde{\boldsymbol{\theta}}) &= \mathbf{E}_{y_i}(\widetilde{\boldsymbol{\theta}})\frac{p}{p-1} \\
\mathbf{E}_{mxi}(p^{-1}, \widetilde{\boldsymbol{\theta}}) &= \mathbf{C}_i(\widetilde{\boldsymbol{\theta}})(p\mathbf{I} - \mathbf{A}_i(\widetilde{\boldsymbol{\theta}}))^{-1}\mathbf{E}_{xi}(\widetilde{\boldsymbol{\theta}})\frac{p}{p-1} \\
\mathbf{E}_{mi}(p^{-1}, \widetilde{\boldsymbol{\theta}}) &= \mathbf{E}_{my_i}(p^{-1}, (\widetilde{\boldsymbol{\theta}})) + \mathbf{E}_{mxi}(p^{-1}, \widetilde{\boldsymbol{\theta}})
\end{aligned}$$

### 6.1.2 Residual generation

Considering that the residuals are generated using the parity space approach, the residual expression generated for each mode is given by:

$$\mathbf{r}_i(k,\boldsymbol{\theta}) = \mathbf{W}_i(\boldsymbol{\theta})\bar{\mathbf{Y}}(k) - \mathbf{W}_i(\boldsymbol{\theta})\mathbf{T}_{iu,\rho}(\boldsymbol{\theta})\bar{\mathbf{U}}(k) - \mathbf{W}_i(\boldsymbol{\theta})\mathbf{T}_{iG,\rho}(\boldsymbol{\theta}) \tag{6.4}$$

where $\rho$ is the residual order, $\mathbf{W}_i(\boldsymbol{\theta})$ is a matrix such that $\mathbf{W}_i(\boldsymbol{\theta})\mathbf{O}_i(\boldsymbol{\theta}) = 0$, and $\mathbf{T}_{iu,\rho}(\boldsymbol{\theta})$, $\mathbf{O}_i(\boldsymbol{\theta})$ and $\mathbf{T}_{iE,\rho}(\boldsymbol{\theta})$ matrices are given by:

$$\mathbf{T}_{iu,\rho}(\boldsymbol{\theta}) = \begin{pmatrix} \mathbf{D}_i(\boldsymbol{\theta}) & \cdots & 0 & 0 \\ \mathbf{C}_i(\boldsymbol{\theta})\mathbf{B}_i(\boldsymbol{\theta}) & \cdots & 0 & 0 \\ \vdots & \ddots & & \vdots \\ \mathbf{C}_i(\boldsymbol{\theta})(\mathbf{A}_i(\boldsymbol{\theta}))^{\rho-1}\mathbf{B}_i(\boldsymbol{\theta}) & \cdots & & \mathbf{D}_i(\boldsymbol{\theta}) \end{pmatrix}$$

$$\mathbf{O}_i(\boldsymbol{\theta}) = \begin{pmatrix} \mathbf{C}_i(\boldsymbol{\theta}) \\ \mathbf{C}_i(\boldsymbol{\theta})\mathbf{A}_i(\boldsymbol{\theta}) \\ \vdots \\ \mathbf{C}_i(\boldsymbol{\theta})(\mathbf{A}_i(\boldsymbol{\theta}))^{\rho} \end{pmatrix}$$

and $\bar{\mathbf{Y}}(k) = \begin{bmatrix} \mathbf{y}(k-\rho) & \mathbf{y}(k-\rho+1) & \cdots & \mathbf{y}(k) \end{bmatrix}^T$, $\bar{\mathbf{U}}(k)$ and $\bar{\mathbf{F}}(k)$ are similar vectors.

$$\mathbf{T}_{iE,q}(\boldsymbol{\theta}) = \begin{pmatrix} \mathbf{E}_{y_i}(\boldsymbol{\theta}) \\ \mathbf{C}_i(\boldsymbol{\theta})\mathbf{E}_{xi}(\boldsymbol{\theta}) + \mathbf{E}_{y_i}(\boldsymbol{\theta}) \\ \vdots \\ \mathbf{C}_i(\boldsymbol{\theta})(\mathbf{A}_i(\boldsymbol{\theta}))^{\rho-1}\mathbf{E}_{xi}(\boldsymbol{\theta}) + \cdots + \mathbf{E}_{y_i}(\boldsymbol{\theta}) \end{pmatrix}$$

Because of the inclusion of uncertain parameters in the continuous dynamics of the hybrid system model, the determination of $\mathbf{W}_i(\boldsymbol{\theta})$ is not a trivial task. One possible approach is proposed in [S. and Adrot, 2006]. Here, a different approach, based on the equivalence that there exists between the parity space approach and input-output models [Ding et al., 2008], is used. Assume that the system model input-output form at a given operating point where the $i^{th}$ output is described the following transfer function:

$$\mathbf{y}_i(q,\boldsymbol{\theta}) = \frac{b_{\rho_i}(\boldsymbol{\theta})p^{\rho} + b_{p-1_i}(\boldsymbol{\theta})p^{\rho-1} + \cdots + b_{0i}(\boldsymbol{\theta})}{p^{\rho} + a_{p-1_i}(\boldsymbol{\theta})p^{\rho-1} + \cdots + a_0(\boldsymbol{\theta})}\mathbf{u}(q) \tag{6.5}$$

A way to construct the parity space residuals is based on defining the transformation vector as follows

$$\mathbf{W}_i(\boldsymbol{\theta}) = \left[\begin{array}{cccc} a_{0i}(\boldsymbol{\theta}) & \cdots & a_{p-1_i}(\boldsymbol{\theta}) & 1 \end{array}\right] \tag{6.6}$$

This definition can be justified according to the Cayley-Hamilton theorem. It can be proved that[1] $\mathbf{W}_i(\boldsymbol{\theta})obsv(\mathbf{A}_i(\boldsymbol{\theta}), \mathbf{C}_i(\boldsymbol{\theta})) = 0$ is satisfied by considering each output of Equation (6.5) independently:

$$\begin{gathered} A_i(\boldsymbol{\theta})^p + a_{\rho-1_i}(\boldsymbol{\theta})A_i(\boldsymbol{\theta})^{\rho-1} + \cdots + a_{0i}(\boldsymbol{\theta})A_i(\boldsymbol{\theta}) = 0 \\ \Rightarrow \left[\begin{array}{cccc} a_{0i}(\boldsymbol{\theta}) & \cdots & a_{p-1_i}(\boldsymbol{\theta}) & 1 \end{array}\right] \left[\begin{array}{c} \mathbf{c}_i(\boldsymbol{\theta}) \\ \mathbf{c}_i(\boldsymbol{\theta})\mathbf{A}_i(\boldsymbol{\theta}) \\ \vdots \\ \mathbf{c}_i(\boldsymbol{\theta})\mathbf{A}_i(\boldsymbol{\theta})^{\rho} \end{array}\right] = 0 \end{gathered}$$

where $\mathbf{A}_i(\boldsymbol{\theta}), \mathbf{c}_i(\boldsymbol{\theta})$ denotes the state space matrices of the transfer function given by Equation (6.5). Moreover,

$$\mathbf{W}_i(\boldsymbol{\theta})\mathbf{T}_{iu,\rho}(\boldsymbol{\theta}) = \left[\begin{array}{cccc} b_{0i}(\boldsymbol{\theta}) & \cdots & b\rho - 1_i(\boldsymbol{\theta}) & b_{\rho_i}(\boldsymbol{\theta}) \end{array}\right]$$

and

$$\mathbf{W}_i(\boldsymbol{\theta})\mathbf{T}_{iE,p}(\boldsymbol{\theta}) = \left[\begin{array}{cccc} e_{0i}(\boldsymbol{\theta}) & \cdots & e_{\rho-1_i}(\boldsymbol{\theta}) & e_{\rho_i}(\boldsymbol{\theta}) \end{array}\right]$$

Under this approach, the number of residuals is equal to the number of system outputs for a given mode.

Alternatively, the residuals can be expressed using the input-output form [Meseguer et al., 2010a] as follows:

$$\mathbf{r}_i^{\circ}(k,\boldsymbol{\theta}) = \mathbf{y}(k) - \mathbf{G}_i(p^{-1},\boldsymbol{\theta})\mathbf{u}(k) - \mathbf{H}_i(p^{-1},\boldsymbol{\theta})\mathbf{y}(k) - \mathbf{E}_{mi}(p^{-1},\boldsymbol{\theta}) \tag{6.7}$$

where $\mathbf{G}_i(p^{-1},\boldsymbol{\theta})$, $\mathbf{H}_i(p^{-1},\boldsymbol{\theta})$ and $\mathbf{E}_{mi}(p^{-1},\boldsymbol{\theta})$ can be obtained from of the input-output model in predictor form. Moreover, with the previous selection of $\mathbf{W}_i(\boldsymbol{\theta})$, an equivalence between input/ouput and parity space predictors can be established through the following relations:

[1] $obsv$ denotes the observability matrix

$$\begin{aligned}
\mathbf{H}_i(p^{-1},\boldsymbol{\theta}) &= \mathbf{I} - \mathbf{W}_i(\boldsymbol{\theta}) \begin{bmatrix} \mathbf{I}p^{-\rho} \\ \vdots \\ \mathbf{I} \end{bmatrix} \\
\mathbf{G}_i(p^{-1},\boldsymbol{\theta}) &= \mathbf{W}_i(\boldsymbol{\theta})\mathbf{T}_{i_{\mathbf{u}},\rho}(\boldsymbol{\theta}) \\
\mathbf{E}_{mi}(p^{-1},\boldsymbol{\theta}) &= \mathbf{W}_i(\boldsymbol{\theta})\mathbf{T}_{i_{\mathbf{G}},\rho}(\boldsymbol{\theta})
\end{aligned}$$

### 6.1.3 Residual evaluation

The set of residuals generated for each mode are compared with a threshold value (zero in the ideal case). When the residual is larger than the threshold, it is concluded that the system is faulty or a mode change has occurred. However, by considering the effect of the uncertain parameters $\boldsymbol{\theta}$ on the estimated output model response, an interval for $\widehat{y}_i(k)$ should be determined at every time instant instead of a single value. Thereby, $\widehat{y}_i(k,\boldsymbol{\theta})$ is bounded by the interval $\left[\underline{\widehat{y}}_i(k,\boldsymbol{\theta}), \overline{\widehat{y}}_i(k,\boldsymbol{\theta})\right]$, where for each output:

$$\underline{\widehat{y_i^j}}(k,\boldsymbol{\theta}) = \min_{\theta\in\Theta}(\widehat{y}_i^j(k,\boldsymbol{\theta})) \quad \text{and} \quad \overline{\widehat{y_i^j}}(k,\boldsymbol{\theta}) = \max_{\theta\in\Theta}(\widehat{y}_i^j(k,\boldsymbol{\theta})) \tag{6.8}$$

with $j \in \{1,\cdots n_y\}$. In the free fault case, each system output fulfils:

$$y_i^j(k,\boldsymbol{\theta}) \in \left[\underline{\widehat{y_i^j}}(k,\boldsymbol{\theta}), \overline{\widehat{y_i^j}}(k,\boldsymbol{\theta})\right] \tag{6.9}$$

Alternatively, the previous fault detection test can be formulated using the residuals given by Eq. (2.1). A convenient way of considering the effect of parameter uncertainty in the residual evaluation consists in using the nominal model $\widehat{y}_i^\circ(k,\boldsymbol{\theta}^\circ)$ obtained under assumption $\boldsymbol{\theta} = \boldsymbol{\theta}^\circ \in \boldsymbol{\Theta}$. In the following, the notation $\widehat{y}_i^\circ(k) \triangleq \widehat{y}_i^\circ(k,\boldsymbol{\theta}^\circ)$ will be assumed. Thus, the nominal residual can be evaluated as follows:

$$\mathbf{r}_i^\circ(k) = \mathbf{y}(k) - \widehat{\mathbf{y}}_i^\circ(k) \tag{6.10}$$

and the effect of parameter uncertainty will be bounded component wise by the following interval

$$\left[\underline{r}_i^{\circ j}(k), \overline{r}_i^{\circ j}(k)\right] \tag{6.11}$$

where:

$$\underline{r}_i^{\circ j}(k) = \underline{\widehat{y}}_i^j(k) - \widehat{y}_i^{\circ j}(k) \quad \text{and} \quad \overline{r}_i^{\circ j}(k) = \overline{\widehat{y}}_i^j(k) - \widehat{y}_i^{\circ j}(k) \tag{6.12}$$

being $\underline{\widehat{y}}_i^j(k)$ and $\overline{\widehat{y}}_i^j(k)$ the bounds of the $j^{th}$ system output estimation computed obtained according to Eq. (6.8).

Once generated, residual (6.10) is evaluated component wise against interval (6.11) to detect a fault:

$$\phi_i^j(k) = \begin{cases} 1 & \text{if } r_i^{\circ j}(k) \notin \left[\underline{r}_i^{\circ j}(k), \overline{r}_i^{\circ j}(k)\right] \quad \text{(fault)} \\ 0 & \text{otherwise} \quad \text{(no fault)} \end{cases} \tag{6.13}$$

where interval $\left[\underline{r}_i^{\circ j}(k), \overline{r}_i^{\circ j}(k)\right]$ plays the role of an adaptive threshold. Thus, the observed fault signature $\mathbf{\Phi}_i(k) = [\phi_i^1(k), \cdots, \phi_i^{ny}(k)]$ is generated in a robust way.

### 6.1.4 Mode discernibility under parametric uncertainty

Theoretically the discernibility between two modes is deduced using the mathematical properties developed in Chapter 3. In practice, the concept of discernibility depends on the residuals belonging to an interval. Hence, non-discernibility with parametric uncertainty must be defined.

**Definition 6.1.** Two modes $q^i$ and $q^j$ are said to be weakly non-discernible if and only if residuals $\mathbf{r}_i^\circ(k)$ (generated considering the mode $i$ model) and $\mathbf{r}_j^\circ(k)$ (generated considering the mode $j$ model) both belong to their intervals (i.e., $\mathbf{r}_i^\circ(k) \in [\underline{\mathbf{r}}_i^\circ(k), \overline{\mathbf{r}}_i^\circ(k)]$, $\mathbf{r}_j^\circ(k) \in \left[\underline{\mathbf{r}}_j^\circ(k), \overline{\mathbf{r}}_i^\circ(k)\right]$ holds) when they are computed using signals $(\mathbf{y}(k), \mathbf{u}(k))$ corresponding to mode $q_i$ or mode $q_j$.

In the case that residuals are generated using the parity space approach, the discernibility function is equivalent to evaluate the following condition (deduced in [Cocquempot et al., 2004]) without parametric uncertainty:

$$rank[\mathbf{O}_i] \neq \quad rank[\mathbf{O}_j] \neq \quad rank\begin{bmatrix} \mathbf{O}_i & \mathbf{O}_j & \Delta_{ij} \end{bmatrix} \tag{6.14}$$

where $\Delta_{ij} = \mathbf{T}_{i\,u,q} - \mathbf{T}_{j\,u,q}$.

When $\mathbf{E}_{x\,i}$ and $\mathbf{E}_{y\,i}$ appear in the continuous dynamics of the hybrid model, a similar analysis can be done to obtain the condition of non discernibility as follows:

$$rank[\mathbf{O}_i(\boldsymbol{\theta})] = rank[\mathbf{O}_j(\boldsymbol{\theta})] = rank\begin{bmatrix} \mathbf{O}_i(\boldsymbol{\theta}) & \mathbf{O}_j(\boldsymbol{\theta}) & \Delta_{ij}(\boldsymbol{\theta}) & \Delta_{E_{ij}}(\boldsymbol{\theta}) \end{bmatrix} \tag{6.15}$$

where $\Delta_{ij}(\boldsymbol{\theta}) = \mathbf{T}_{i\,u,q}(\boldsymbol{\theta}) - \mathbf{T}_{j\,u,q}(\boldsymbol{\theta})$ and $\Delta_{\mathbf{E}_{x\,ij}}(\boldsymbol{\theta}) = \mathbf{T}_{i\,E,q}(\boldsymbol{\theta}) - \mathbf{T}_{j\,E,q}(\boldsymbol{\theta})$.

For non-structural faults, the residual fault sensitivity can be determined using its internal form. In the case of the parity space approach, this form is given by [Blanke et al., 2006] as follows:

$$\mathbf{r}_i(k) = \mathbf{W}_i(\boldsymbol{\theta})\mathbf{T}_{if,p}(\boldsymbol{\theta})\bar{\mathbf{F}}(k) \tag{6.16}$$

where $\mathbf{T}_{if,p}(\boldsymbol{\theta})$ is a matrix similar to $\mathbf{T}_{iu,p}(\boldsymbol{\theta})$ replacing $\mathbf{D}_i(\theta)$ with $\mathbf{F}_{y_i}(\boldsymbol{\theta})$, $\mathbf{B}_i(\boldsymbol{\theta})$ with $\mathbf{F}_{xi}(\boldsymbol{\theta})$ and $\bar{\mathbf{F}}(k)$ is a vector similar to $\bar{\mathbf{Y}}(k)$. According to [Meseguer et al., 2010a], the residual fault sensitivity is given by

$$\boldsymbol{\Lambda}_i(p^{-1}) = \frac{\partial \mathbf{r}_i(k)}{\partial \mathbf{f}_{ns}} \tag{6.17}$$

Thus, the residual fault sensitivity under the parity space approach is given by:

$$\boldsymbol{\Lambda}_i(p^{-1}, \boldsymbol{\theta}) = \mathbf{W}_i(\boldsymbol{\theta})\mathbf{T}_{if,\rho}(\boldsymbol{\theta}) \begin{bmatrix} \mathbf{I}_{nf}p^{-\rho} \\ \vdots \\ \mathbf{I}_{nf} \end{bmatrix} \tag{6.18}$$

A non-structural fault affecting the system can be detected if the active residuals in the theoretical fault signature satisfy that $r_i^{\circ j}(k) \notin \left[\underline{r}_i^{\circ j}(k), \overline{r}_i^{\circ j}(k)\right]$.

### 6.1.5 Mode tracking logic

Algorithm 6.1 briefly describes the residual-based reasoning carried out by the diagnoser to identify an event occurrence. The algorithm checks for the current diagnoser state whether $\mathbf{r}_i^{\circ}(k) \in [\underline{\mathbf{r}}_i^{\circ}(k), \overline{\mathbf{r}}_i^{\circ}(k)]$ holds or not. In case of a mode change, the set of residuals of some successor mode will fulfill $\mathbf{r}_j^{\circ}(k) \in \left[\underline{\mathbf{r}}_j^{\circ}(k), \overline{\mathbf{r}}_j^{\circ}(k)\right]$. In the case of a fault, the set of binary residuals in the current mode are compared with the theoretical fault signature to isolate the fault.

### 6.1.6 Illustrative example

Let us consider only tanks $T_1, T_2, T_3$ in the sewer network example (see Section 5.1) and assume the following mode sequence $q_1 \rightarrow q_3 \rightarrow q_1$ has occurred. The full hybrid automaton model is only composed of four nominal modes (neglecting structural faulty modes to simplify the problem) and 16 non-structural faulty modes. The measurements provide by rain gauges and limnimeters are plot in Figs. 6.1 and 6.2, with a sampling time of $\Delta t = 300s$.

Through measurements provided by limnimeters, it is observed that $T_2$ is in overflow ($L_{41}(k)$). For instance, the predictor used for residual generation corresponding to mode 3 is

**Algorithm 6.1** Event_Processing_Uncertainty($q_D$)

**loop**
  **wait until** $\mathbf{r}_i^\circ(k) \notin [\underline{\mathbf{r}}_i^\circ(k), \overline{\mathbf{r}}_i^\circ(k)]$ **or** $\sigma_o \in \Sigma_o$ occurs
  **if** $\sigma_o$ occurs **then**
    $\sigma_D := \sigma_o$
  **else**
    **for all** $q_{Dj} \in Succs(q_{Di})$ **do**
      **if** $\mathbf{r}_j^\circ(k) \in \left[\underline{\mathbf{r}}_j^\circ(k), \overline{\mathbf{r}}_j^\circ(k)\right]$ **then**
        $COND1 := true$
        `break`
      **end if**
    **end for**
    **for all** $q_{Dj} \in Succs(q_{Di})$ **do**
      **if** $\Phi_{q_{D_i}}(k) = \mathbf{FS}_{\nu_i}(\bullet, \mathcal{F}^\iota_{\nu_i})$ **then**
        $COND2 := true$
        `break`
      **end if**
    **end for**
    **if** $COND1 = false$ **and** $COND2 = false$ **then**
      **print** `Unknown event`
    **else**
      **if** $COND1$ and $COND2$ **then**
        $\sigma_D := \delta$
      **else**
        **if** $COND1$ **then**
          $\sigma_D := \delta_{\nu_i - \nu_j}$
        **else**
          $\sigma_D := \delta_{\mathcal{F}^\iota_{\nu_i}}$
        **end if**
      **end if**
    **end if**
    **return**
  **end if**
**end loop**

$$\widehat{\mathbf{y}}^{\circ 3}(k) = \begin{bmatrix} \theta_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \theta_2 \end{bmatrix} \mathbf{y}(k) + \begin{bmatrix} \theta_3 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{u}(k) + \begin{bmatrix} 0 \\ \theta_4 \\ \theta_5 \end{bmatrix} \tag{6.19}$$

that has been obtained using the state space model of the sewer network using matrix $\mathbf{W}(\boldsymbol{\theta})$. The uncertain parameters have been estimated using the algorithm proposed by [S. and Adrot, 2006] leading to the following intervals: $\theta_1 \in [0.7083, 0.8657]$, $\theta_2 \in [0.8460, 1.0340]$, $\theta_3 \in [1.0162, 1.2420] \cdot 10^4$, $\theta_4 \in [3.3942, 4.1485]$ and $\theta_5 \in [0.1196, 0.1462]$. The value of $\mathbf{W}(\boldsymbol{\theta})$ for this mode is:

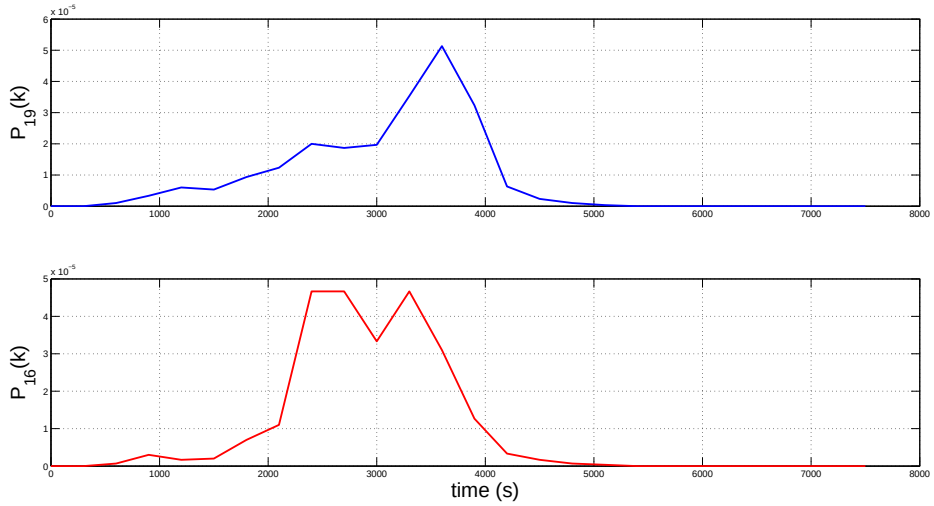


Figure 6.1: Measurements provided by rain gauges

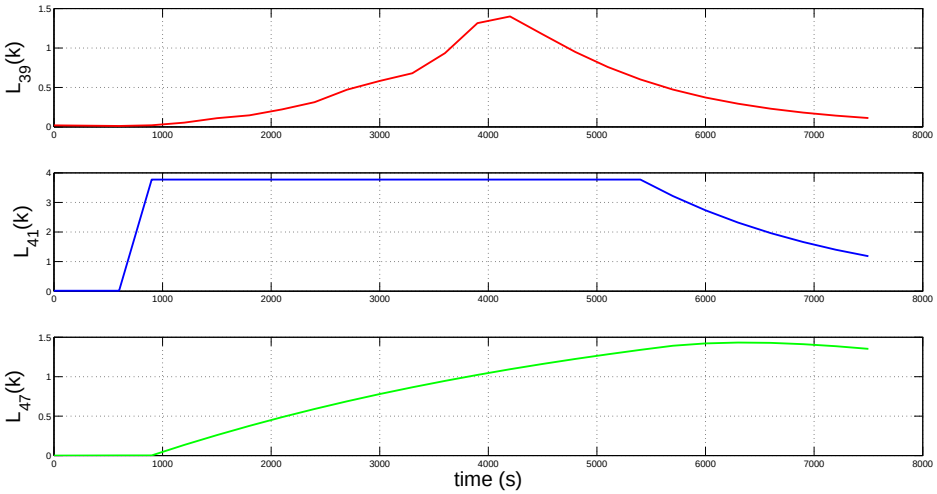


Figure 6.2: Measurements provided by limnimeters

$$\mathbf{W}(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{\theta}_1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \boldsymbol{\theta}_2 & 0 & 0 & 1 \end{bmatrix}$$

Fig. 6.3 illustrates the residual evolution for the considered scenario. Notice that, for instance, when a transition from mode $q_1 \rightarrow q_3$ occurs then $\mathbf{r}_1^\circ(k) \notin [\underline{\mathbf{r}}_1^\circ(k), \overline{\mathbf{r}}_1^\circ(k)]$ and $\mathbf{r}_3^\circ(k) \in [\underline{\mathbf{r}}_3^\circ(k), \overline{\mathbf{r}}_3^\circ(k)]$ holds. Remark that all modes are discernible according to the criterion explained in Section 6.1.4.

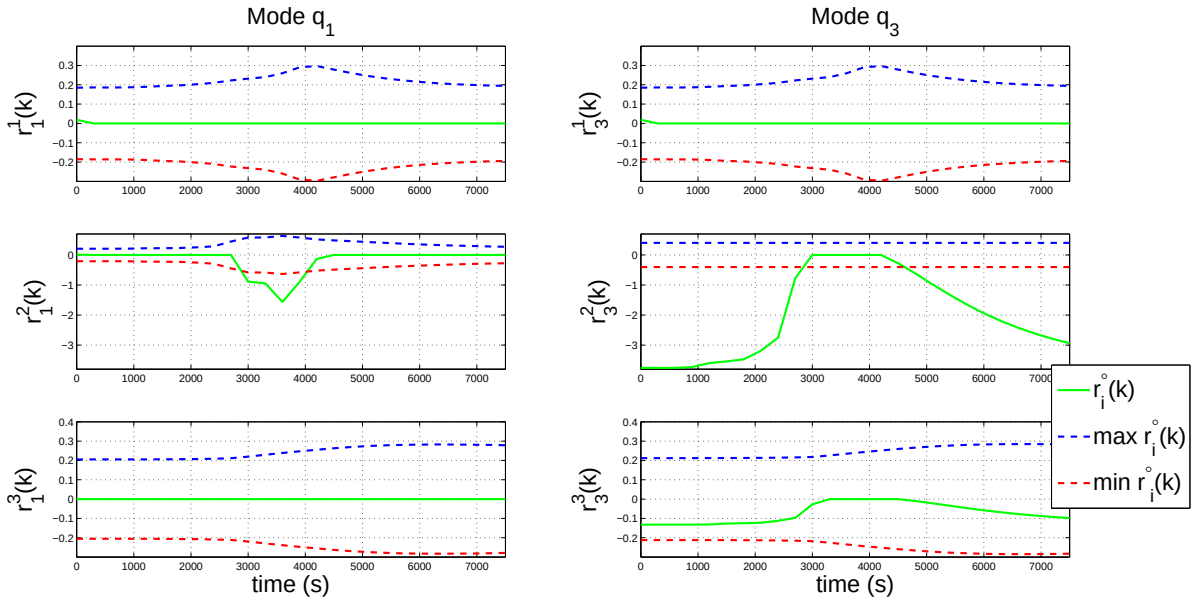


Figure 6.3: Mode change detection using interval models

Finally, an additive fault in sensor $L_{39}$ occurs at time $3600s$. Consequently the residuals of mode $q_3$ are triggered and the diagnoser stops. Notice that in Fig. 6.4, when the fault occurs $\mathbf{r}_3^\circ(k) \notin [\underline{\mathbf{r}}_3^\circ(k), \overline{\mathbf{r}}_3^\circ(k)]$ holds. In fact, the observed fault signature is $\Phi_i(k) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^t$, which corresponds to the theoretical fault signature obtained applying Eq. (6.18).

Fig. 6.5 shows in solid line the simulated system state evolution, whereas the dashed line is the state sequence estimated by the diagnoser.

From this results, is observed the robustness of the proposed methodology under parametric uncertainty since modeling errors are considered in the detection process. Parity space equations are used to evaluate the residuals online removing the dependence on state variables. Uncertainty is determined based on the equivalence that there exists between input-output models and parity equations.

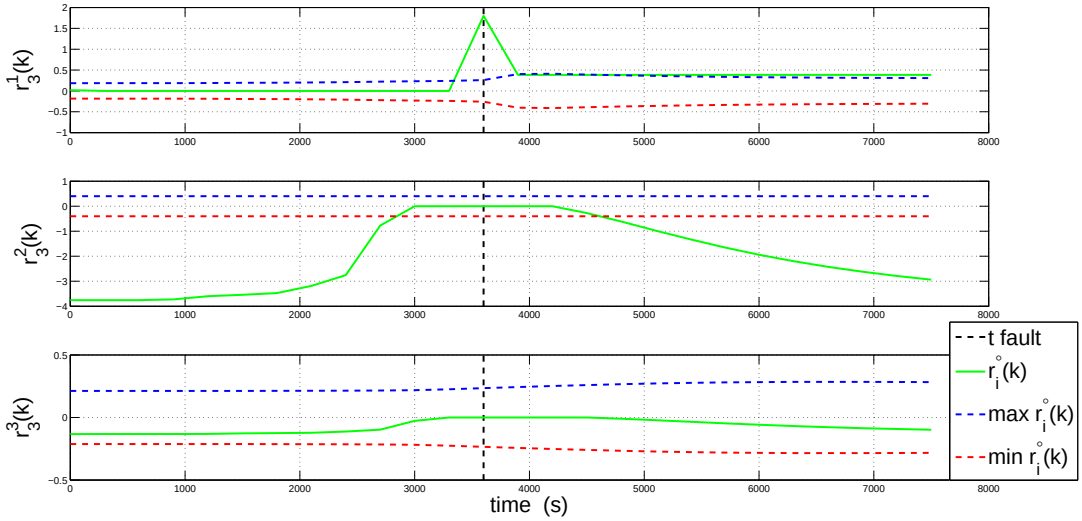


Figure 6.4: Fault detection using interval models

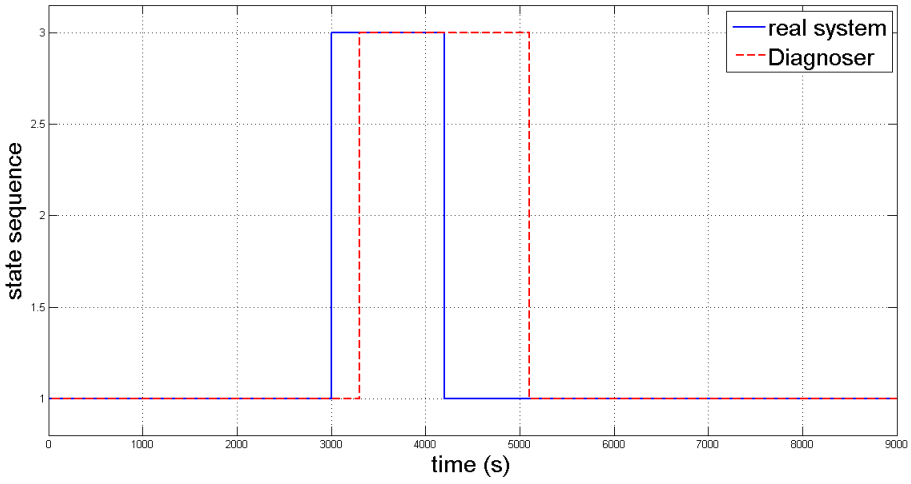


Figure 6.5: Diagnoser vs. system using interval models

## 6.2 Hybrid diagnosis based on components, extending the DX approach

Applying the methodologies developed, diagnosis involves detecting and isolating a fault using the DX approach. In previous chapters, the diagnosis task assumes that the faults to be diagnosed are known and can be modeled, e.g. additive faults in sensors or other anticipated fault models. The contribution of the section is to adapt the fault diagnosis methodology for hybrid systems such that fault models should not be anticipated nor modeled.

The proposed enhancement of the methodology relies on the use of consistency-based reasoning through a set of ARRs along the FDI approach and proposes to localize the faulty components using the component support of the ARRs following the DX component oriented approach.

A hybrid automaton is used to represent the discrete-event system behavior and the continuous dynamics associated to each system component. Given the component oriented model, a set of component-supported ARRs is generated for each mode using structural analysis [Blanke et al., 2006]. The advantages of applying this approach for hybrid systems are that no fault models are needed, multiple faults are naturally in the scope of the method, and the equations that describe the component continuous dynamics can be nonlinear. All these issues had not been previously considered.

### 6.2.1 Hybrid model adaptation

The behavior of a component $M_j \in \mathcal{M}$ in a mode $q^i \in \mathcal{Q}$ is governed by an equation denoted by $C_j^i$. Model equations depend on a set of physical variables, denoted by $Z$, which is divided in two subsets $Z = X \cup K$, unknown and known variables.

A component $M_j \in \mathcal{M}$ is represented by a set of equations as follows:

$$M_j = \{C_j^i : \forall q^i \in \mathcal{Q}\} \tag{6.20}$$

On the other hand, the system model or system description *SD* is given by the whole set of equations. For the sake of simplicity, it is assumed that the number of components is the same in all modes of the hybrid automaton, and that each component behavior is described by a unique equation.

Summarizing, the hybrid system model is described by the following hybrid automaton

$$HA =< \mathcal{Q}, X, U, Y, \mathcal{F}, \mathcal{M}, \Sigma, \mathcal{T} >$$

where:

- $\mathcal{Q} = \{q^i : i \in \{1, 2, \cdots, w\}\}$ is a set of modes (discrete states in the automaton) and $W$ indexes the modes. $q^0$ is the initial mode.
- $X$ is the subset of non-measured or unknown variables.
- $U \subseteq K$ is the subset of control variables.
- $Y \subseteq K$ is the subset of measured variables.
- $\mathcal{M}$ is the set of components, the same for each mode $i \in W$, which are modeled as in Eq. (6.20).
- $f_{M_j} \in \mathcal{F}$ is a set of faults. Each fault concerns one single component.
- $\Sigma = \Sigma_s \cup \Sigma_c$ is a set of events. Spontaneous mode switching events ($\Sigma_s$) and input events ($\Sigma_c$) are considered. A spontaneous event $\sigma_s \subseteq \Sigma_s$ is issued when the state vector intersects a jump surface $S_{\sigma_s} = \{X : s_{\sigma_s} X = \mathbf{0}\}$, with $s_{\sigma_s}$ being a linear switching condition.
- $\mathcal{T} : \mathcal{Q} \times \Sigma \rightarrow \mathcal{Q}$ defines a discrete state transition function.

### 6.2.2 Residuals generation using structural models

The structural analysis proposed by [Staroswiecki and Declerck, 1989] is a way to connect both approaches using *component-supported ARRs*. ARR support (also known as *possible conflicts*) was introduced in [Cordier et al., 1995] as the components whose models are involved in the redundancy expressions. The structural analysis approach allows to derive automatically analytical redundancy relations from elementary component models [Blanke et al., 2006], keeping track of the components used in this process. The problem of generating ARRs from model equations involves finding just overdeterminated subsystems of equations (constraints) [Krysander et al., 2008].

Combining the measurement models with the process model, the analytical expressions of ARRs [Blanke et al., 2006], which are defined as relations between known variables, can then be derived. These relations are used in the fault diagnosis procedure to check consistency between the observed and the predicted system behavior. As soon as an inconsistency is detected, fault isolation is triggered to provide an explanation in terms of feasible faults.

According to [Cordier et al., 1995], a method to compute diagnosis using the DX approach is based upon the concept of *R-conflict* (or for short *conflict* in the following). A conflict points out correctness assumptions for the components which underly a symptom (i.e. a discrepancy between the system description $SD$, the set of components $COMPS$, and the observations $OBS$ generated using a consistency checking procedure). At least, one of the components in a conflict is faulty in order to account for the observations (or equivalently it cannot be the case that all the components of the conflict behave normally). A minimal conflict is a conflict, which does not strictly include (set inclusion) any conflict. Using minimal conflicts, it is possible to give a characterization of minimal diagnosis, which provides a basis for computing them. This characterization is based on the *minimal hitting set* concept [Cordier et al., 1995].

A hitting set for a collection $\mathcal{C}$ of sets is a set $H \subseteq \cup\{S|S \in \mathcal{C}\}$ such that $H \cap S \neq \{\}$ for each $S \in \mathcal{C}$. A hitting set is minimal if and only if no proper subset of it is a hitting set of $\mathcal{C}$. Thus, a (minimal) diagnosis $\Delta$ given a system description $SD$, a set of components $COMPS$ and observations $OBS$ is obtained as a (minimal) hitting set for the collection of (minimal) conflicts for $(SD, COMPS, OBS)$.

The set of residuals for each mode is given by the following equation:

$$\mathbf{r}^i(k) \quad = \quad \mathbf{g}^i(\mathbf{u}(k), \mathbf{y}(k)) \tag{6.21}$$

The set of residuals are generated using structural analysis theory [Blanke et al., 2006]. A structural model is an abstraction of the analytical model taking into account which variables are involved in which equations, neglecting the mathematical expression of the equations.

The structural model is defined as follows [Blanke et al., 2006], adapted for hybrid systems:

**Definition 6.2.** Given a set of model equations, $\mathcal{C}^j$, that depend on variables $\mathcal{Z} = \{\mathcal{X} \cup \mathcal{K}\}$, with $\mathcal{K} = \{\mathcal{Y} \cup \mathcal{U}\}$, the structural model is defined as a bipartite graph $BG^j = \{\mathcal{C}^j, \mathcal{Z}, \xi\}$ where $\mathcal{C}^j$ and $\mathcal{Z}$ are the vertices and $\xi$ is the set of edges defined by:

$(c_l^j, z_\varepsilon) \in \xi$ if the variable $z_\varepsilon$ appears in constraint $c_l^j$

### 6.2.3 Fault detection and isolation based on components

Fault diagnosis consists in determining a set of hypotheses about component states (through conflicts) that are consistent with observations. As already mentioned, the relation between conflicts and ARRs is provided by the concept of component-support [Cordier et al., 1995, Travé-Massuyès et al., 2006].

**Definition 6.3.** The *support* of an AAR, is the set of underlying components whose primary relations (constraints) are involved in the ARR.

The fault signature matrix or minimal conflict matrix (**MCM**$_i$) in mode $q_i$ crosses ARRs in rows against faults in columns. Every row provides the support of an ARR and every column the fault signature of a fault. Concerning component faults the interpretation of some entry of $r_i^j$ being zero is that the $\mathcal{M}_j$ component does not belong to the support of the $j^{th}$ ARR. Otherwise, in case of being one, the $j^{th}$ component belongs to the support of the $i^{th}$ ARR.

Then, if an ARR in mode $q_i$ is not satisfied by the set of observations, the support of this ARR is a conflict.

**Definition 6.4.** A fault $f_{\mathcal{M}_j}$ is detectable in a mode $q_i$, if $\mathcal{M}_j$ belongs to the *support* of some ARR in this mode.

The fault diagnosis is obtained from the minimal hitting sets of conflicts, and these conflicts can be seen as the set of ARR supports not satisfied by the set of observations. These conflicts can be derived by looking at the rows of the minimal conflict matrix corresponding to violated ARRs.

**Definition 6.5.** Two faults $f_{\mathcal{M}_i}$ and $f_{\mathcal{M}_j}$, are said to be isolable if and only if for any observation, when $f_{\mathcal{M}_i}$ is among the diagnosis candidates, $f_{\mathcal{M}_j}$ never is, and conversely.

For example, for a given mode $q_i$, consider the theoretical $\mathbf{MCM}_i$ in Table 6.1. Assume that the actual signature derived from observations is $r_i^1 = 1$, $r_i^2 = 1$ and $r_i^3 = 0$. Thus, the $\mathbf{MCM}_i$ involves the rows in Table 6.1 corresponding to ARRs $r_i^1$ and $r_i^2$. Next, using the $\mathbf{MCM}_i$ and the actual signature, the minimal conflicts on the hypothesis about the behavior on the components can be derived. Therefore, the minimal diagnosis that can be derived, assuming that only one component can fail and based only in the set of activated residuals, is: there is a fault in $f_{\mathcal{M}_1}$ or a fault in $f_{\mathcal{M}_2}$.

| | $f_{\mathcal{M}_1}$ | $f_{\mathcal{M}_2}$ | $f_{\mathcal{M}_3}$ | $f_{\mathcal{M}_4}$ | $f_{\mathcal{M}_5}$ |
|---|---|---|---|---|---|
| $r_i^1$ | 1 | 1 | 0 | 1 | 0 |
| $r_i^2$ | 1 | 1 | 0 | 0 | 1 |
| $r_i^3$ | 0 | 1 | 1 | 0 | 0 |

Table 6.1: $\mathbf{MCM}_i$ for mode $q_i$

### 6.2.4 Illustrative example

Let us consider the same scenario in Section 6.1.6, to illustrate the method enhancement based on components. The set of components $\mathcal{M}$, which in this case are the same for all modes, consist of: 3 tanks ($T_1$, $T_2$ and $T_3$), 3 output pipes (one for each tank) and 5 sensors ($L_{39}, L_{41}, L_{47}, P_{19}, P_{16}$). Each component $M_j$ involves a single equation $c_j$. The set of elementary models $c_j$ (e.g., for mode $q_3$) are given by the following relations:

$$
\begin{aligned}
c_1 &: \quad v_0(k+1) = v_0(k) + \Delta t(S_{19}\varphi_{19}P_{19} - \beta_0 v_0(k)) \\
c_2 &: \quad v_1(k+1) = \overline{v_1} \\
c_3 &: \quad v_2(k+1) = (1 - \Delta t\beta_2)v_2(k) + \Delta t\beta_1\overline{v_1}
\end{aligned}
$$

$$
\begin{array}{llll}
c_4: & L_{39}(k) = \frac{\beta_0}{M_{39}} v_0(k) & c_8: & L_{41}(k) = L_{41m}(k) \\
c_5: & L_{41}(k) = \frac{\beta_1}{M_{41}} \overline{v_1} & c_9: & L_{47}(k) = L_{47m}(k) \\
c_6: & L_{47}(k) = \frac{\beta_2}{M_{47}} v_2(k) & c_{10}: & P_{19}(k) = P_{19m}(k) \\
c_7: & L_{39}(k) = L_{39m}(k) & c_{11}: & P_{16}(k) = P_{16m}(k)
\end{array}
$$

In each mode the system is modeled using structural analysis where $Z = \{v_0, v_1, v_2, P_{19}, P_{16}, L_{39}, L_{41}, L_{47}, P_{19m}, P_{16m}, L_{39m}, L_{41m}, L_{47m}\}$, $K = \{P_{19m}, P_{16m}, L_{39m}, L_{41m}, L_{47m}\}$ and $X = \{v_0, v_1, v_2\ P_{19}, P_{16}, L_{39}, L_{41}, L_{47}\}$.

In order to simplify the residual generation, assume that only faults in the following components $\{M_7, M_8, M_9, M_{10}, M_{11}\}$ are considered in $\mathcal{F}$. These faults correspond to output sensor faults $\{f_{L39}, f_{L41}, f_{L47}\}$ and input sensor faults $\{f_{P19}, f_{P16}\}$, respectively.

Residuals are generated using structural analysis [Blanke et al., 2006], by eliminating the unknown variables between elementary relations. This process is carried out in each mode obtaining a set of residuals that depend on the mode. Next, a minimal conflict matrix is generated. For example for mode $q_3$, the minimal conflict matrix is given by Table 6.2.

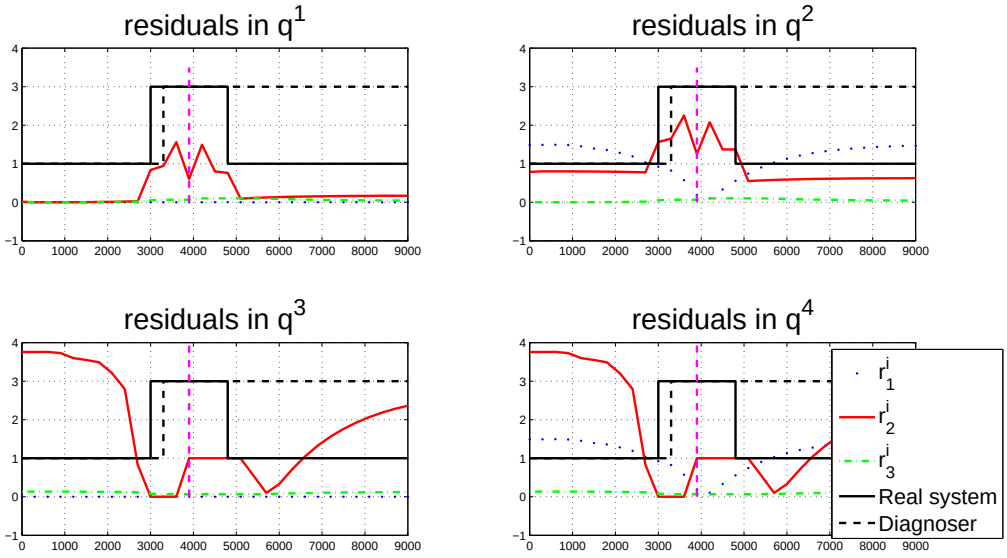


Figure 6.6: Mode tracking based on consistency indicators

| | $f_{M_7}$ | $f_{M_8}$ | $f_{M_9}$ | $f_{M10}$ | $f_{M11}$ |
|---|---|---|---|---|---|
| $r_1^3$ | 1 | 0 | 0 | 1 | 0 |
| $r_2^3$ | 0 | 1 | 0 | 0 | 0 |
| $r_3^3$ | 0 | 0 | 1 | 0 | 0 |

Table 6.2: Minimal conflict matrix for mode $q^3$

Fig. 6.6 shows in solid line the simulated system state evolution, while the dashed line is the state sequence estimated by the diagnoser. The figure also plots the set of residuals in each mode.

According to Fig. 6.6, the estimated mode coincides with the real mode of the system as long as the corresponding set of residuals are equal to zero (e.g, residuals $\mathbf{r}_1(k) = \mathbf{0}$ when in mode 1, $\mathbf{r}_3(k) = \mathbf{0}$ when in mode 3, and so on).

The diagnoser report is provided in Table 6.3. Transition $q_1 \rightarrow q_3$ occurs at $3000s$ and it is reported at $3300s$. Then, an additive fault in sensor $L_{41}$ appears at time $3900s$ and it is detected at $4200s$ when the system is in mode $q_3$.

| Mode change | Reported event | State diagnoser | Occurrence time (s) | Detection time (s) |
|---|---|---|---|---|
| $q_1 \rightarrow q_3$ | $\delta_{14}$ | $(q_3, \{\})$ | 3000 | 3300 |
| $q_3 \rightarrow q_3^2$ fault $f_2 \in \mathcal{F}_{ns}$ in Mode $q_3$ | $\delta_{\mathcal{F}_{\nu_4}^2}$ | $(q_3^2, \{f_2\})$ | 3900 | 4200 |

Table 6.3: Hybrid diagnoser report

# CHAPTER 7

# CONCLUSIONS

## 7.1 Contributions

A methodology and architecture to design and implement a diagnoser in the framework of hybrid systems has been proposed. The diagnoser design is based on the model of the system using an hybrid automata that is used to calculate the set of residuals for each mode. The implementation and logic to operate the diagnoser is based on these residuals to detect transitions between states in the hybrid automaton.

A method to incrementally build a hybrid diagnoser has been presented. The diagnoser is built whenever the system requires it after an event occurs (signature-event or input event). The method comprises the detection and isolation of structural and non-structural faults which are included in the system model. The diagnoser executes the tasks of mode recognition and identification using consistency indicators generated from a set of residuals for every mode, and then builds the part of the diagnoser required according to the system operation. Thus, the obtained diagnoser requires less memory space and can be efficiently generated online.

A tool has been developed which allow to build the diagnoser in an automatic way. The performance of the proposed approach has been successfully tested in a representative part of the Barcelona sewer network.

The methodology has been extended to build a diagnoser based on reasoning about components. Consequently detection and isolation of multiple faults is possible and nonlinear models can be included for hybrid systems diagnsis. Besides, parametric uncertainty has been considered in the methodology. Parity space equations are used to evaluate the residuals online, eliminating the dependence on state variables. Uncertainty is determined based on the equivalence that there exists between input-output models and parity equations.

## 7.2 Directions for future research

To continue the research proposed in this thesis, some ideas are outlined below:

- The algorithms of the diagnoser building could be improved from the implementation point of view. They would be adapted to incrementally compute the necessary part of the diagnoser, since currently the DIADES tool computes all the feasible traces based on the observable event occurrence.
- The inclusion of a logic based on the delays of the residuals, which takes into account the activation order of the residuals, would be an interesting issue to deal with. The discernibility property could be extended taking into account to the residual activation order. Thus, the mode identification methodology would be improved.
- Uncertainty in the parameters for hybrid systems can be extended, applying other techniques as the set membership approach. These techniques will improve the consistency tests in the methodology (to detect and isolate a mode change). It is also possible to study the noise effect in the measurements.
- The implementation software tool that allows the automatic the generation of the set of residuals using structural models. The process of the residuals generation is still carried out manually. The possibility to generate in an automatic way would allow to take into account nonlinearities in online diagnosis.
- Discernibility properties could be studied in the case of residuals being generated taking into account non-linear behaviors. From this point of view, the residual design have to be adapted to non-linear functions. Moreover, mathematical properties to evaluate discernibility should be studied using non-linear models. From the practical point of view, the consistency tests to detect and isolate mode changes in the system operate as in the linear case.
- To propose a solution based on distributed diagnosers built incrementally for extremely complex systems such as the case of the Barcelona sewer network, which is controlled locally. The methodology would include how local diagnosers interact each other to give a correct diagnostic.

# BIBLIOGRAPHY

[Bayoudh and Travé-Massuyès, 2012] Bayoudh, M. and Travé-Massuyès, L. (2012). Diagnosability analysis of hybrid systems cast in a discrete-event framework. *International Journal of Discrete Events Dynamic Systems (JDEDS)*.

[Bayoudh et al., 2008] Bayoudh, M., Travé-Massuyès, L., and Olive, X. (2008). Hybrid systems diagnosis by coupling continuous and discrete event techniques. In *Proceedings of the 17th International Federation of Automatic Control, World Congress, IFAC-WC*, pages 7265–7270, Seoul (Korea).

[Bayoudh et al., 2009] Bayoudh, M., Travé-Massuyès, L., and 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.*, pages 6970 – 6975, Shanghai (China).

[Benazera and Travé-Massuyès, 2009] Benazera, E. and 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.

[Biswas et al., 2004] Biswas, G., Cordier, M., Lunze, J., L. Travé-Massuyès, L., and Staroswiecki, M. (2004). Diagnosis of complex systems: bridging the methodologies of the fdi and dx communities. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 34(5):57–95.

[Blanke et al., 2006] Blanke, M., Kinnaert, M., Lunze, J., and Staroswiecki, M. (2006). *Diagnosis and Fault Tolerant Control*. Springer, 2st edition.

[Blom and Bar-Shalom, 1988] Blom, H. and Bar-Shalom, Y. (1988). The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE Transactions on Automatic Control*, 33:780–783.

[Bregon et al., 2012] Bregon, A., Alonso, C., Biswas, G., Pulido, B., and Moya, N. (2012). Fault diagnosis in hybrid systems using possible conflicts. In *Fault Detection, Supervision and Safety of Technical Processes*, volume 8, pages 132–137.

[Cassandras and Lafortune, 2008] Cassandras, C. and Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Springer.

[Cembrano et al., 2004] Cembrano, G., Quevedo, J., Salamero, M., Puig, V., and Figueras, J. (2004). Optimal control of urban drainage systems: a case study. *Control Engineering Practice*, 12(1):1–9.

[Chow and Willsky, 1984] Chow, E. and Willsky, A. (1984). Analytical redundancy and the design of robust failure detection systems. *IEEE Transactions on Automatic Control*, 29(7):11–13.

[Cocquempot et al., 2004] Cocquempot, V., Mezyani, T., and Staroswiecki, M. (2004). Fault detection and isolation for hybrid systems using structured parity residuals. In *5th Asian Control Conference*.

[Cordier et al., 1995] Cordier, M., Dague, P., Lévy, F., Montmain, J., Staroswiecki, M., and Travé-Massuyès, L. (1995). Conflicts versus analytical redundancey relations: a comparative analysis of the model-based diagnosis approach from the artificial intelillence and automatic control perspectives. *IEEE Trans. on Systems, Man, and Cybernetics. Part B:Cybernetics*, 34(5):2163–2177.

[Daigle, 2008] 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, Tennessee.

[de Freitas, 2002] de Freitas, N. (2002). Rao-blackwellised particle filtering for fault diagnosis. In *Proceedings of the IEEE Aerospace Conference 2002*, volume 4, pages 1767–1772.

[Ding et al., 2008] Ding, X., Kinnaert, M., Lunze, J., and Staroswiecki, M. (2008). *Model Based Fault Diagnosis Techniques*. Springer.

[Dong and He, 2006] Dong, M. and He, D. (2006). Hidden semi-markov model-based methodology for multi-sensor equipment health diagnosis and prognosis. *European Journal of Operational Research*, 178:858–878.

[Georges et al., 2011] Georges, J.-P., Theilliol, D., Cocquempot, V., Ponsart, J.-C., and Aubrun, C. (2011). Fault tolerance in networked control systems under intermittent observations. *Int. J. Appl. Math. Comput. Sci*, 21(4):639–648.

[Gertler, 1997] Gertler, J. (1997). Fault detection and isolation using parity relations. *Control Engineering Practice*, 5(5):653–661.

[Heemels et al., 2001] Heemels, W., Schutter, B. D., and Bemporad, A. (2001). Equivalence of hybrid dynamical models. *Automatica*, 37:1085–1091.

[Hofbaur and Williams, 2004] Hofbaur, M. W. and Williams, B. C. (2004). Hybrid estimation of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 34(5):2178–2191.

[Isserman, 1997] Isserman, R. (1997). Supervision, fault-detection, and fault-diagnosis methods -an introduction. *Control Engineering Practice*, 5(5):639–652.

[Krysander et al., 2008] Krysander, M., Aslund, J., and Nyberg, M. (2008). An efficient algorithm for finding minimal overconstrained subsystems for model-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics Part A*, 38(1):197–206.

[Lygeros et al., 2003] Lygeros, J., Henrik, K., and Zhang, J. (2003). Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48.

[Meseguer et al., 2010a] Meseguer, J., Puig, V., and Escobet, T. (2010a). Fault diagnosis using a timed discrete event approach based on interval observers. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 40(5):900–916.

[Meseguer et al., 2010b] Meseguer, J., Puig, V., and Escobet, T. (2010b). Observer gain effect in linear interval observer-based fault detection. *Journal of process control*, 20(8):944–956.

[Mezyani, 2007] Mezyani, T. (2007). *Diagnostic des Systèmes Dynamiques Hybrides*. PhD thesis, Université Lille1, France.

[Mignone, 2002] Mignone, D. (2002). *Control and Estimation of Hybrid Systems with Mathematical Optimizations*. PhD thesis, Swiss Federal Institute of Technology (ETH) Zurich, Italy.

[Narasimhan and Biswas, 2007] Narasimhan, S. and Biswas, G. (2007). Model-based diagnosis of hybrid systems. *IEEE Transactions on Systems, Man and Cybernetics*, 37(3).

[Ocampo, 2007] Ocampo, C. (2007). *Model Predictive Control of Complex Systems including Fault Tolerance Capabilities: Application to Sewer Networks*. PhD thesis, University Polytechnic of Catalonia, Barcelona, Spain.

[Ocampo and Puig, 2009] Ocampo, C. and Puig, V. (2009). Fault-tolerant model predictive control within the hybrid systems framework: Application to sewer networks. *International Journal of Adaptive Control and Signal Processing*, 23(8):757–787.

[Patton and Chen, 1997] Patton, R. J. and Chen, J. (1997). Observer-based fault detection and isolation: robustness and applications. *Control Engineering Practice*, 5(5):671–682.

[Pencolé, 2012] Pencolé, Y. (2012). How to use dd-diagnoser. `http://homepages.laas.fr/~ypencole/diades/html/tutorials/diagnoser/tutorial_diagnoser.html#Sampath-1995-ID1`.

[Puig et al., 2004] Puig, V., Quevedo, J., Escobet, T., and Pulido, B. (2004). On the integration of fault detection and isolation in model based fault diagnosis. In *5th International Workshop on Principles of Diagnosis*, Carcassonne,France.

[Ramadge and Wonham, 1989] Ramadge, P. and Wonham, W. (1989). The control of discrete-event systems. *Proc IEEE*, 77(1):81–98.

[Reither, 1987] Reither, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95.

[S. and Adrot, 2006] S., S. P. and Adrot, O. (2006). Parity relations for linear uncertain dynamic systems. *Automatica*, 42:1553–1562.

[Sampath et al., 1995] Sampath, M., Sengupta, R., and Lafortune, S. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575.

[Staroswiecki and Comtet-Varga, 2001] Staroswiecki, M. and Comtet-Varga, G. (2001). Analytical redundace relations for fault detection and isolation in algebraic dynamic systems. *Automatica*, 37(5):687–699.

[Staroswiecki and Declerck, 1989] Staroswiecki, M. and Declerck, P. (1989). Analytical redundancy in non-linear interconnected systems by means of structural analysis. In *IFAC Advanced Information Processing in Automatic Control (AIPAC'89)*, pages 51–55, Nancy (France).

[Travé-Massuyès et al., 2009] Travé-Massuyès, L., Bayoudh, M., and 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*, pages 6970–6975, Shanghai, P.R. China.

[Travé-Massuyès et al., 2006] Travé-Massuyès, L., Escobet, T., and Olive, X. (2006). Diagnosability analysis based on component-supported analytical redundancy relations. *IEEE Transactions on Systems, Man, and Cybernetics Part A*, 34(6):1146–1160.

[Vento et al., 2010] Vento, J., Puig, V., and Sarrate, R. (2010). Fault detection and isolation of hybrid system using diagnosers that combine discrete and continuous dynamics. In *Conference on Control and Fault Tolerant System*, Nice, French.

[Vento et al., 2011] Vento, J., Puig, V., and Sarrate, R. (2011). A methodology for building a fault diagnoser for hybrid systems. In *9th European Workshop on Advance Control and Diagnosis*, Budapest, Hungry.

[Vento et al., 2012] Vento, J., Puig, V., and Sarrate, R. (2012). Parity space hybrid system diagnosis under model uncertainty. In *20th Mediterranean Conference on Control and Automation (MED)*, Barcelona, Spain.

[Zhao et al., 2005] Zhao, F., Koustsoukos, X., Haussecker, H., Reich, J., and Cheung, P. (2005). Monitoring and fault diagnosis of hybrid systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 35(6):1225–1240.

# Part I

# Appendices

# APPENDIX A

# MATLAB CODE OF THE IMPLEMENTED PROGRAMS

## A.1 Simulink scheme for online diagnosis execution

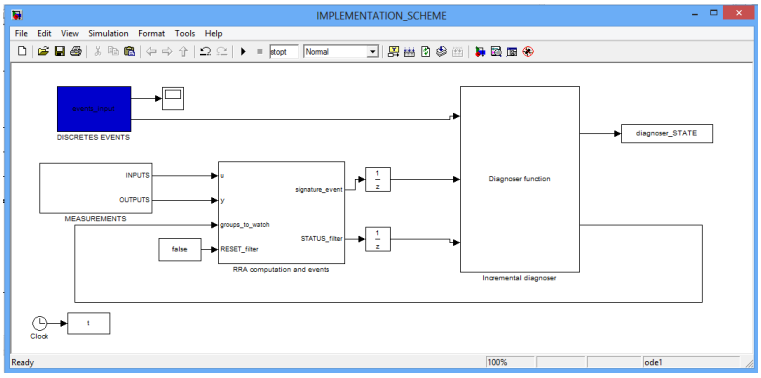


Figure A.1: Implementation scheme

## A.2 Incremental composition function

```
case 'HA_k'

    q=param1;
```

```
q_p=param2;

adj_mat=IncComposition('get_adj_mat');
inc_mat=IncComposition('get_inc_mat');
modes=IncComposition('get_HA_modes')+1;
events=IncComposition('get_HA_events');

adj_mat_aux={};
inc_mat_aux={};
[o,p]=size(inc_mat);

for hh=1:o
    for h=1:IncComposition('get_HA_modes')
        adj_mat_aux{hh,h}=adj_mat{hh,h};
    end
end

for hh=1:o
    for h=1:IncComposition('get_HA_events')
        inc_mat_aux{hh,h}=inc_mat{hh,h};
    end
end

modes_visited=IncComposition('get_Qv_HA');
if find(modes_visited==q)
    new_mode=find(modes_visited==q);
else
    modes_visited=[modes_visited q];
    IncComposition('set_Qv_HA',modes_visited);
    new_mode=size(adj_mat,1)+1;

    adj_mat_aux{new_mode,q_p}=1;
end

label_str=IncComposition('get_label_mode_HAinc',q);
label_num=IncComposition('get_label_num_HAinc',q);
label_modes_inc=label_num;
label_str_inc=label_str;
```

```
for z=1:IncComposition('get_Ncomponents')
    incidence_matrix=IncComposition('get_incidence_matrix',z);
    Gamma=incidence_matrix(label_num(z),:);

    for x=1:IncComposition('get_events_size',z)

        if Gamma(1,x)~=0
            label_modes_inc(z)=Gamma(1,x);
            ev_name=IncComposition('get_event_name',z,x);
            [visited,mode_v]=find_mode(label_modes_inc,adj_mat);
            [stored,event_v]=find_event(ev_name,inc_mat);
            if stored==0
                events=events+1;
                IncComposition('set_label_event_HAinc',events,ev_name);
            end

            if label_modes_inc==label_num

                disp(['Succs modes of q' num2str(q) ' to q' num2str(q) ' label
                    ...' mat2str(label_num)  '  event:...'  ev_name])

                if stored==1
                    inc_mat_aux{new_mode,event_v}=mode_v;

                else
                    inc_mat_aux{new_mode,events}=mode_v;

                end
                    adj_mat_aux{new_mode,q}=1;
            else
                if visited==1
                    adj_mat_aux{new_mode,mode_v}=1;
                    if stored==0
                        inc_mat_aux{new_mode,events}=mode_v;
                    else
                        inc_mat_aux{new_mode,event_v}=mode_v;
                    end
```

```
                    disp(['Succs modes of q' num2str(q) ' to q' num2str(mode_v
                        ) ' label ...' mat2str(label_modes_inc)  ' event:...'
                         ev_name])
                else
                    LC=IncComposition('get_label_id',z,Gamma(1,x));
                    label_str_inc{z}=LC;
                    disp(['Succs modes of q' num2str(q) ' to q' num2str(modes)
                         ' label ...' mat2str(label_modes_inc)  ' event:...'
                        ev_name])
                    adj_mat_aux{new_mode,modes}=1;
                    if stored==0
                        inc_mat_aux{new_mode,events}=modes;
                    else
                        inc_mat_aux{new_mode,event_v}=modes;
                    end

                    IncComposition('set_label_mode_HAinc',modes, label_str_inc
                        );
                    IncComposition('set_label_num_HAinc',modes,
                        label_modes_inc);
                    modes=modes+1;
                end
            end
        end
        label_str_inc=label_str;
        label_modes_inc=label_num;
    end
end
IncComposition('set_HA_events',events);
IncComposition('set_HA_modes',size(adj_mat_aux,2));
IncComposition('set_adj_mat',adj_mat_aux);
IncComposition('set_inc_mat',inc_mat_aux);
```

## A.3 Parametrized equations of the sewer network

### A.3.1 State space matrices

```
function [Ai,Bi,Gi,Ci,Di,X0i]=parametrized_mode(LM,VOLMAX_,E_,S_,CVC_,M_,DELTAT,alfa_,
    w1,nx,nu)

virtual_tanks=9;
control_gates=3;

pos_init=virtual_tanks+1;

for k=1:control_gates

    OpenGate=strmatch('op',LM(pos_init));

    if ~isempty(OpenGate)
        if OpenGate==1;
            alfa_(k)=0;
        else
            alfa_(k)=1;
        end
    else
        StuckOpen=strmatch('sc',LM(pos_init));

        if StuckOpen==1;
            alfa_(k)=0;
        else
            alfa_(k)=1;
        end
    end

    pos_init=pos_init+1;
end

Ai=zeros(nx,nx);
Ai(1,1)=((1-DELTAT*CVC_(1))*sat(LM,1,VOLMAX_))*(1-dzn(LM,1));
Ai(2,1)=(1-alfa_(1))*DELTAT*CVC_(1)*sat(LM,1,VOLMAX_)*(1-dzn(LM,1))*(1-dzn(LM,2));
Ai(3,1)=0;
Ai(4,1)=alfa_(1)*CVC_(1)*DELTAT*(-1+dzn(LM,1))*(1-dzn(LM,3));
Ai(2,2)=((1-DELTAT*CVC_(2))*sat(LM,2,VOLMAX_))*(1-dzn(LM,2));
Ai(4,2)=alfa_(2)*CVC_(2)*DELTAT*(1-dzn(LM,2))*(1-dzn(LM,3));
Ai(3,2)=(1-alfa_(2))*CVC_(2)*DELTAT*(1-dzn(LM,2));
```

```
Ai(3,3)=1-CVC_(3)*DELTAT;
Ai(4,3)=DELTAT*CVC_(3);
Ai(4,4)=(1-DELTAT*CVC_(4))*sat(LM,3,VOLMAX_)*(1-dzn(LM,3));
Ai(5,4)=DELTAT*w1*CVC_(4)*sat(LM,3,VOLMAX_)*(1-dzn(LM,3))*(1-dzn(LM,4));
Ai(6,4)=DELTAT*(1-w1)*CVC_(4)*sat(LM,3,VOLMAX_)*(1-dzn(LM,3))*(1-dzn(LM,5));
Ai(5,5)=(1-DELTAT*CVC_(5))*sat(LM,4,VOLMAX_)*(1-dzn(LM,4));
Ai(5,9)=DELTAT*w1*(1-alfa_(3))*CVC_(9)*(1-dzn(LM,8))*sat(LM,8,VOLMAX_)*(1-dzn(LM,4));
Ai(6,6)=(1-DELTAT*CVC_(6))*sat(LM,5,VOLMAX_)*(1-dzn(LM,5));
Ai(6,9)=DELTAT*(1-w1)*(1-alfa_(3))*CVC_(9)*(1-dzn(LM,8))*sat(LM,8,VOLMAX_)*(1-dzn(LM
    ,5));
Ai(7,7)=(1-DELTAT*CVC_(7))*sat(LM,6,VOLMAX_)*(1-dzn(LM,6));
Ai(7,5)=(DELTAT*CVC_(5))*sat(LM,4,VOLMAX_)*(1-dzn(LM,4))*(1-dzn(LM,6));
Ai(8,8)=(1-DELTAT*CVC_(8))*sat(LM,7,VOLMAX_)*(1-dzn(LM,7));
Ai(9,9)=(1-DELTAT*CVC_(9))*sat(LM,8,VOLMAX_)*(1-dzn(LM,8));
Ai(10,10)=(1-DELTAT*CVC_(10))*sat(LM,9,VOLMAX_)*(1-dzn(LM,9));
Ai(10,9)=(DELTAT*CVC_(9)*alfa_(3))*sat(LM,9,VOLMAX_)*(1-dzn(LM,8))*(1-dzn(LM,9)); %
% MATRIX Bi
%nu=4;
Bi=zeros(nx,nu);
Bi(1,1)=DELTAT*S_(1)*E_(1)*(1-dzn(LM,1));
Bi(2,2)=DELTAT*S_(2)*E_(2)*(1-dzn(LM,2));
Bi(4,3)=DELTAT*S_(3)*E_(3)*(1-dzn(LM,3));
Bi(5,3)=DELTAT*S_(4)*E_(4)*(1-dzn(LM,4));
Bi(6,3)=DELTAT*S_(5)*E_(5)*(1-dzn(LM,5));
Bi(7,4)=DELTAT*S_(6)*E_(6)*(1-dzn(LM,6));
Bi(8,3)=DELTAT*S_(7)*E_(7)*(1-dzn(LM,7));
Bi(9,2)=DELTAT*S_(8)*E_(8)*(1-dzn(LM,8));
Bi(10,2)=DELTAT*S_(9)*E_(9)*(1-dzn(LM,9));
% MATRIX Gi
Gi=zeros(nx,1);
X0i=zeros(nx,1);

Gi(1)=VOLMAX_(1)*dzn(LM,1);
Gi(2)=VOLMAX_(2)*dzn(LM,2)+(1-alfa_(1))*DELTAT*CVC_(1)*sat(LM,1,VOLMAX_)*(dzn(LM,1))
    *(1-dzn(LM,2));
Gi(3)=(1-alfa_(2))*CVC_(2)*DELTAT*(dzn(LM,2));
Gi(4)=VOLMAX_(3)*dzn(LM,3)+alfa_(2)*CVC_(2)*DELTAT*(dzn(LM,2))*(1-dzn(LM,3))+alfa_(1)*
    CVC_(1)*DELTAT*(dzn(LM,1))*(1-dzn(LM,3));
```

```
74 Gi(5)=VOLMAX_(4)*dzn(LM,4)+DELTAT*w1*CVC_(4)*sat(LM,3,VOLMAX_)*(dzn(LM,3))*(1-dzn(LM
       ,4))+DELTAT*w1*(1-alfa_(3))*CVC_(9)*(dzn(LM,8))*sat(LM,8,VOLMAX_)*(1-dzn(LM,4));
   Gi(6)=VOLMAX_(5)*dzn(LM,5);
76 Gi(7)=VOLMAX_(6)*dzn(LM,6)+(DELTAT*CVC_(5))*sat(LM,4,VOLMAX_)*(dzn(LM,4))*(1-dzn(LM,6)
       );
   Gi(8)=VOLMAX_(7)*dzn(LM,7);
78 Gi(9)=VOLMAX_(8)*dzn(LM,8);
   Gi(10)=VOLMAX_(9)*dzn(LM,9)+(DELTAT*CVC_(9)*alfa_(3))*sat(LM,8,VOLMAX_)*(dzn(LM,8))
       *(1-dzn(LM,9));
80
   Ci= eye(nx,nx);
82
   for j=1:nx
84     for cc=1:nx
           if Ci(j,cc)==1
86             Ci(j,cc)=CVC_(j)/M_(j);
           end
88     end
   end
90
   Di=zeros(nx,nu);
92
   end
```

## A.4 Residual generator

```
    y=[];
    y_max=[];
    y_min=[];
    measurements=[];

    for i=1:n_groups    %SysHybride('get_automate_size')
        y=[y;SysHybride('get_matGn',i)*input(:,2)+SysHybride('get_matHn',i)*output
            (:,2)+SysHybride('get_matQn',i)];
        y_max=[y_max;SysHybride('get_maxGn',i)*input(:,2)+SysHybride('get_maxHn',i)*
            output(:,2)+SysHybride('get_maxQn',i)];
        y_min=[y_min;SysHybride('get_minGn',i)*input(:,2)+SysHybride('get_minHn',i)*
            output(:,2)+SysHybride('get_minQn',i)];
```

```
        measurements=[measurements;output(:,1)];
    end
    %————— RESIDUALS GENERATION —————%
    r=measurements-y;
    r_max=y_max-y;
    r_min=y_min-y;

%    if current_time==600
%        keyboard
%    end
    umbs=Diagnoser1('get_threhold_total');
   r_bin2=abs(r)>umbs;

    for i=1:length(r)
        if r(i)>r_max(i) || r(i)<r_min(i)
            r_bin(i)=true;
        else
            r_bin(i)=false;
         end
    end
```

### A.4.1 Matrices of the residual equations

```
%————————— var_predictor ——————————————————————————————————%
%   Parameters of the predcitor model.
%
%————————————————————————————————————————————————————————————%
                var_predictor= struct(...
                    'matHn',{},...
                    'matGn',{},...
                    'matQn',{},...
                    'matHnq',{},...
                    'matGnq',{},...
                    'matQnq',{}...
                    );
```

```
%————————————————————————————————————————————————————————————%
%       Predictor Model
```

```
%-------------------------------------------------------------------------%
    case 'get_matHn'
        var_predictor(param1).matHn = SysHybride('get_matC',param1)*...
            SysHybride('get_matA',param1)*...
            (SysHybride('get_matC',param1))^(-1);

        return_value=var_predictor(param1).matHn;

    case 'get_matGn'
        var_predictor(param1).matGn = SysHybride('get_matC',param1)*...
            SysHybride('get_matB',param1)+SysHybride('get_matD',param1);
        return_value=var_predictor(param1).matGn;

    case 'get_matQn'
        var_predictor(param1).matQn = SysHybride('get_matC',param1)*...
            SysHybride('get_matGx',param1);
        return_value=var_predictor(param1).matQn;
%-------------------------------------------------------------------------%
```

### A.4.2 Fault signature matrix

```
function FS_mode_parametrized(i)

PARAM=IncComposition('get_Parameters_Network');

VOLMAX_=PARAM.VOL;
E_=PARAM.E;
S_=PARAM.S;
CVC_=PARAM.CVC;
M_=PARAM.M;
DELTAT=PARAM.Ts;
alfa_=PARAM.alfa;
w1=PARAM.w1;
nx=PARAM.nx;
nu=PARAM.nu;
ny=PARAM.ny;

LMi=IncComposition('get_label_mode_HAinc',i);
```

```
[Ai,Bi,Gi,Ci,Di]=parametrized_mode(LMi,VOLMAX_,E_,S_,CVC_,M_,DELTAT,alfa_,w1,nx,nu);

Hni=Ci*Ai*(Ci)^(-1);
%Gni=Ci*Bi+Di;

syms q
Hnqi=Ci*((q*eye(nx))^(-1))*Ai*(Ci)^(-1);
Gnqi=Ci*((q*eye(nx))^(-1))*Bi+Di;

Fy_=[zeros(ny,nu) eye(ny)];
Fy=eye(ny);
Fx=eye(nu);

FSM_num=[];

 Fx_=[-Bi zeros(ny)];
          Gf=(Ci*((q*eye(nx)-Ai)^(-1)))*Fx_+Fy_;
         FSM=(eye(ny)-Hni)*Gf;

          Sfy=(eye(ny)-Hnqi)*Fy;
          Sfu=-Gnqi*Fx;
          nfu=size(Sfu,2);
          nfy=size(Sfy,2);
          FSM_aux=[];
          for j=1:nfu
              FSM_aux=[FSM_aux Sfu(:,j)];
          end
          for j=1:nfy
              FSM_aux=[FSM_aux Sfy(:,j)];
          end
          FSM_=FSM_aux;
          disp(['mode ',num2str(i)]);


          XX=subs(FSM_,q,1);
          FSM_num_= double(XX);
          FSM_num=double(FSM_num_~=0);    % Faults inputs / Faults outputs
          Diagnoser1('set_FSM',i,FSM_num);
           [FSM_aux,comp,conj]=isolable_set_faults1(FSM_num);
```

```
            Diagnoser1('set_FSM_isolable_faults',i,FSM_aux);
            Diagnoser1('set_events_faults',i,conj);
            FSM_isolable_set{i}=FSM_aux;
            Diagnoser1('set_components_by_sets',i,comp);

end
```

## A.5 $B$ builder code implemented in Matlab

```
function [N_states,N_trans,events_obs,events_faults,all_trans_DIADES,label_modes,
    N_upd_]=...
    recalculate_B_inc1(mode_init,N_events,N_groups,N_ant_inc) %,groups)
%disp('Incremental B building')
PARAM=IncComposition('get_Parameters_Network');

VOLMAX_=PARAM.VOL;
E_=PARAM.E;
S_=PARAM.S;
CVC_=PARAM.CVC;
M_=PARAM.M;
DELTAT=PARAM.Ts;
alfa_=PARAM.alfa;
w1=PARAM.w1;
nx=PARAM.nx;
nu=PARAM.nu;
% mode: current mode
% N_ant_inc: firts numeric label mode that will be used by non-structural
% faulty modes
% N_upd_: the last numeric label mode used.
% N: automate size
% N_groups: number of diagnosable groups
% N_events: number of events
% groups: sets of discernable modes in HA

% initial variables
% Q_SF \cup Q_N
mode=mode_init;
label_modes=[];
```

```
all_trans_DIADES=[];
events_obs=[];
events_faults=[];
% Q_NSF
label_modes_NF=[];
all_trans_DIADES_NF=[];
events_obs_NF=[];
events_faults_NF=[];

N_trans=0;
N_states=1;
label_modes=[label_modes mode];

N_modes_NF=0;
N_trans_NF=0;

N_ant=N_ant_inc;

visited_m=IncComposition('get_Qv_HA');
for k=1:size(visited_m,2)
    disp(['k  ' num2str(k)])

    mode=visited_m(k);
    [n, succs_modes, events_number] = automaton_next_states_inc(mode);

    for i=1:n
        disp(['i' num2str(i)])
        is_obs=IncComposition('get_HAevent_obs',events_number(i));
        are_discernible=discernible_function(mode,succs_modes(i),VOLMAX_,E_,S_,CVC_,M_
            ,DELTAT,alfa_,w1,nx,nu);
        is_fault_ev=IncComposition('get_HAevent_fault',events_number(i));

        if sum(label_modes==succs_modes(i))==0
            % Do not add repeated elemensts.
            label_modes=[label_modes succs_modes(i)];
            N_states=N_states+1;
        end

        if is_obs
```

```
            label_event=events_number(i);
68          trans_DIADES=[num2str(mode) ' -> ' num2str(succs_modes(i)) ' ' num2str(
                label_event) '\n'];
            if sum(events_obs==events_number(i))==0
70              events_obs=[events_obs events_number(i)];
            end
72          all_trans_DIADES=[all_trans_DIADES trans_DIADES];
            N_trans=N_trans+1;
74      else


76

        if are_discernible
78          eid = event_id_from_residuals(mode, succs_modes(i)); % eid source-
                destination
            buff = eid - N_events;
80          source_gid = mod(buff, (N_groups+1)); % source mode calcualated from eid
            dest_gid = (buff - source_gid)/(N_groups+1); % destination mode
                calcualated from eid
82

            label_event=eid;
84

            if sum(events_obs==eid)==0
86              events_obs=[events_obs eid];
            end
88

            if is_fault_ev
90

                label_mode_inter=['20' num2str(mode) num2str(i)];
92              label_mode_num=str2num(label_mode_inter);
                label_modes=[label_modes label_mode_num];
94              N_states=N_states+1;

96              if sum(events_faults==events_number(i))==0
                    events_faults=[events_faults events_number(i)];
98              end

100             trans_DIADES_inter=[num2str(mode) ' -> ' num2str(label_mode_num) ' '
                    num2str(events_number(i)) '\n'];
                all_trans_DIADES=[all_trans_DIADES trans_DIADES_inter];
```

```
102                 N_trans=N_trans+1;
                    trans_DIADES=[num2str(label_mode_num) ' -> ' num2str(succs_modes(i)) '
                        ' num2str(label_event) '\n'];
104                 all_trans_DIADES=[all_trans_DIADES trans_DIADES];
                    N_trans=N_trans+1;
106             end

108         else

110             trans_DIADES=[num2str(mode) ' -> ' num2str(succs_modes(i)) ' ' num2str(
                    events_number(i)) '\n'];
                if is_fault_ev
112                 if sum(events_faults==events_number(i))==0
                        events_faults=[events_faults events_number(i)];
114                 end
                end
116             all_trans_DIADES=[all_trans_DIADES trans_DIADES];
                N_trans=N_trans+1;
118
            end % are_disc
120
            end % is_obs
122
    end
124
  % disp('FS')
126   [N_modes_NF,N_trans_NF,events_obs_NF,events_faults_NF,all_trans_DIADES_NF,
          label_modes_NF,N_upd]=...
          recalculate_B_FNS_inc1(mode,N_ant);
128
    N_ant=N_upd;
130 N_upd_=N_ant;
    N_trans=N_trans+N_trans_NF;
132
    for w=1:length(label_modes_NF)
134     if sum(label_modes==label_modes_NF(w))==0
            label_modes=[label_modes label_modes_NF(w)];
136         N_states=N_states+1; % update counter
        end
```

```
138     end

140     for w=1:length(events_obs_NF)
            if sum(events_obs==events_obs_NF(w))==0
142             events_obs=[events_obs events_obs_NF(w)];
            end
144     end
        for w=1:length(events_faults_NF)
146         if sum(events_faults==events_faults_NF(w))==0
                events_faults=[events_faults events_faults_NF(w)];
148         end
        end
150
        all_trans_DIADES=[all_trans_DIADES all_trans_DIADES_NF];
152

    end
```

## A.6 Mode tracking logic code implemented in Matlab

```
function Outputs(block)

current_time = get_param('HYBRID_SCHEME','SimulationTime');

persistent n_groups
persistent first_pass
persistent residuals r rp rb %FSM
persistent FSM_is faults_ev
persistent mode_change_ev next_states
persistent start
persistent fault_detected

current_mode=block.InputPort(2).Data; % 1

if current_time==SysHybride('get_time_step')
    start=true;
end

if isempty(first_pass)
```

```
    current_mode=1;
    residuals=zeros(SysHybride('get_output_size')*SysHybride('get_automate_size'),2);
    n_groups = diagnosables_groups();
    r=cell(1,n_groups);
    rp=cell(1,n_groups);
    rb=cell(1,n_groups);

    FSM_is=cell(1,n_groups);
    start=false;
    for i=1:n_groups

        FSM_is{i}=Diagnoser('get_FSM_isolable_faults',i);

    end
    total_faults_ev= Diagnoser('get_total_events_faults'); %sum(Isolable_set);
    faults_ev=false(1,total_faults_ev)';
    fault_detected=false;
    mode_change_ev=false(1,size(Diagnoser('get_events_id'),2));
    next_states=build_next_statesII(current_mode);
    first_pass = 1;

end

residuals = circshift(residuals, [0,1]);
residuals(:,1)=block.InputPort(1).Data;
j=1;
aux=[];
for i=1:SysHybride('get_automate_size')

    r{i}=residuals(j:SysHybride('get_output_size')*i,1);
    rp{i}=residuals(j:SysHybride('get_output_size')*i,2);
    j=j+SysHybride('get_output_size');
    rb{i}=sum(r{i})~=0;
    aux= [aux;rb{i}];

end

next_states=build_next_statesII(current_mode);
```

```
if current_time~=0
    if find(aux==0) % there is no fault in the system
        if ~fault_detected
            ev_aux=[];
            succs=length(next_states);
            for j=1:succs

                if sum(r{next_states(j)})==0 && sum(rp{current_mode})~=0
                    disp(['Mode change detected from mode ',num2str(current_mode),' to
                        mode ',num2str(next_states(j)),' at time ', num2str(
                        current_time)])
                    ev_aux=[ev_aux;1];
                else
                    ev_aux=[ev_aux;0];
                end

            end
            mode_change_ev((current_mode-1)*succs+1:current_mode*succs)=ev_aux;
            if sum(ev_aux>1)
                fprintf('   [t][Diagnoser automaton] WARNING : multiples events have
                    been dropped!\n');

            end
        end
    end
end
%%-------------------EVENTS ASSOCIATED TO FSM ONLY WITH ISOLABLE FAULT SETS
if current_time~=0 % Assuming that at instant k=0 there is no fault
    if aux % all residuals are different from zero in all modes.
        if sum(rp{current_mode})~=0
            [init,final]=calcul_index(current_mode); %to isolate residuals from mode
                of interest to compare with FS
            init_1=init;
            for i=1:size(FSM_is{current_mode},2)
                comps=Diagnoser('get_components_by_sets',current_mode);
                if r{current_mode}==FSM_is{current_mode}(:,i)
                    y=comps(i);
                    disp(['Fault detected in ',y,' in mode ',num2str(current_mode),'
                        at time: ',num2str(current_time)])
```

```
                    disp(['Signature ' mat2str(FSM_is{current_mode}(:,i))])
                    faults_ev(init)=1;
                    fault_detected=true;
                    disp('STOP DIAGNOSIS')
                     set_param('HYBRID_SCHEME', 'SimulationCommand', 'stop')
                    current_time = get_param('HYBRID_SCHEME','SimulationTime');
                end
                init=init+1;
            end
            if sum(faults_ev(init_1:final)>1)
                fprintf('   [t][Diagnoser automaton] WARNING : multiples events have
                    been dropped !\n');
            end
        end
    end
end
block.OutputPort(1).Data=start;
block.OutputPort(2).Data=mode_change_ev;
block.OutputPort(3).Data=faults_ev;
```