UNIVERSITAT POLITÈCNICA DE CATALUNYA

Programa de Doctorat:

AUTOMÀTICA, ROBÒTICA I VISIÓ

Tesi Doctoral

Sensor Placement for Fault Diagnosis based on Structural Models: Application to a Fuel Cell Stack System

Albert Rosich Oliva

Directors: Ramon Sarrate
Fatiha Nejjari

Abril de 2011

To Marta

Abstract

Diagnosing faults correctly can prevent serious damage on systems or a loss of process performance. For this reason, the study of fault diagnosis system design is an important research area. However, the application of novel techniques in industrial processes are rare. This is mainly due to the fact that the achieved diagnosis performance is not the expected one, or the design techniques are cumbersome and difficult to implement. The present work tries to mitigate these problems by choosing the location of sensors in the process. It is known that diagnosis systems strongly depend on the on-line information acquired from the process. Therefore, appropriate sensor location will lead to better diagnosis performance and implementation facilities.

In order to properly deal with the sensor placement problem for complex systems, a suitable framework based on structural models is developed beforehand. Some simplifications (e.g. single faults, no faulty models and a one-to-one correspondence between a model equation and a fault) are considered in order to only focus on the sensor placement analysis. This framework also allows to easily handle installed sensors in the system and to derive some properties that will be used later to develop the proposed sensor placement approaches. Faults affecting the placed sensors are also considered in this framework, as well as redundant sensors (i.e. sensors that measure an already measured variable).

In this work, the combinatorial nature of the sensor placement problem is shown. This means that representing the complete set of possible solutions may be unfeasible when the number of candidate sensors grows. Therefore, only optimal solutions will be sought. The present work proposes three new approaches to solve the optimal sensor placement problem for fault diagnosis based on structural models. The first approach focuses on efficiently generating the minimal redundant sub-models when different candidate sensor configurations are tested. The second approach is based on binary integer programming. And the third one is specifically devoted

to perform the sensor placement when the unknown variable computation in the residual generators is taken into account.

Deriving residuals from the minimal redundant sub-models requires computing the unknown variables of such sub-models. This is not a trivial issue when complex systems with non-linear equations (e.g. look-up tables, piecewise functions, maps, etc.) are considered. For this reason, a method that ensures the unknown variable computation by assigning causality between equations and variables is developed. This method is combined with the sensors placement problem. Sensors are placed such that the unknown variable computation in the residual generators is guaranteed.

Finally, the sensor placement techniques are applied to a fuel cell stack system. The model used to describe the behaviour of this system consists of 96 equations, most of them non-linear. Furthermore, there are 30 candidate sensors to improve the diagnosis specifications. The results obtained from this case study are used to strength the applicability of the proposed approaches.

Keywords: optimal sensor placement, fault diagnosis, fault detection and isolation, structural analysis, minimal redundant sub-model, binary integer programming, causal computability, fuel cell stack system.

Resum

Diagnosticar fallades correctament pot evitar importants d'anys en els sistemes o la pèrdua de prestacions del procés. Per aquest motiu, l'estudi del disseny de diagnosticadors de fallades és una important àrea de recerca. Malgrat tot, la implementació de tècniques innovadores en els processos industrial és poc freqüent. Això és degut, principalment, a que les prestacions assolides pel diagnosticador dissenyat estan per sota de les esperades o bé a que les tècniques de disseny són massa complexes i difícilment implementables. El present treball té per objectiu intentar resoldre aquests problemes mitjançant la localització de sensors en el procés. D'aquesta manera, instal·lant els sensors apropiats s'obtenen millores en les prestacions del diagnosticador i facilitats d'implementació.

Per tal de poder treballar adequadament amb el problema de localització de sensors en sistemes complexos, prèviament es desenvolupa un marc de treball basat en models estructurals. En aquest marc de treball es suposen una sèrie de simplificacions (fallades simples, models de fallada inexistents i una correspondència d'un a un entre equacions del model i fallades) amb la finalitat de centrar-se només en la problemàtica de la localització de sensors. Aquest marc de treball també permet considerar de forma eficient els sensors instal·lats en el sistema, així com la possibilitat d'extreure algunes propietats de diagnosis que seran posteriorment considerades a l'hora de desenvolupar els diferents mètodes proposats per a la localització de sensors. A més, es tenen també en compte les fallades que afecten als sensors instal·lats, així com sensors redundants (sensors que mesuren una variable que ja està mesurada).

El problema de la localització de sensors és de naturalesa combinatòria, tal i com es mostra en el present treball. Això significa que la representació de totes les possibles solucions pot esdevenir impossible quan el nombre de sensors a considerar és gran. Per tant, tan sols es buscarà la solució òptima. Es proposen tres enfocaments diferents per tal de resoldre la localització òptima de sensors per diagnosi de fallades basada en models estruc-

turals. El primer mètode es centra en la generació eficient de sub-models de redundància mínima per verificar les diferents configuracions de sensors candidates. El segon mètode utilitza la programació binària entera per resoldre el problema. Finalment, el tercer mètode té en compte el càlcul de les variables desconegudes en els generadors de residus a l'hora de resoldre el problema de la localització de sensors.

La extracció de residus a partir dels sub-models de redundància mínima requereix la determinació del valor de les variables desconegudes d'aquests sub-models. Aquesta tasca no és gens fàcil al tractar-se de sistemes complexes on apareixen equacions no lineals (per exemple, taules, funcions a trossos, corbes característiques, etc.). Per aquest motiu, s'ha desenvolupat un mètode que permet assegurar la calculabilitat de les variables desconegudes mitjançant l'assignació de causalitats entre equacions i variables. Aquest mètode es combina amb la localització de sensors. És a dir, es cerquen els sensors òptims que cal instal·lar en el sistema, per tal de garantir la calculabilitat de les variables desconegudes en els generadors de residus.

Les tècniques de localització de sensors són aplicades a un sistema basat en una pila de combustible. El model emprat per descriure el comportament d'aquest sistema consta de 96 equacions, la majoria de les quals són no lineals. A més, hi ha la possibilitat d'instal·lar fins a 30 sensors per tal de millorar la diagnosis del sistema. Degut a aquestes característiques del sistema i del model, els resultats obtinguts mitjançant aquest cas d'estudi reafirmen l'aplicabilitat dels mètodes proposats.

Paraules clau: localització de sensors òptima, diagnosis de fallades, detecció i aïllament de fallades, anàlisis de models estructurals, sub-model de redundància mínima, programació binària i entera, calculabilitat causal, sistema de pila de combustible.

Acknowledgement

This dissertation was carried out at the Advanced Control Systems (SAC) with the financial support of the Spanish Ministerio de Educación y Ciencia within the FPI program (CICYT project DPI2005-05415). I would like to thank my supervisor, professor Joseba Quevedo, for letting me join the research group. I greatly appreciate all the work done by my advisor, Dr. Ramon Sarrate, and my co-advisor, Dr. Fatiha Nejjari, for proofreading the manuscript and their guidance on my research.

I am indebted to Dr. Fernando Bianchi for patiently answering my questions and shedding some light on my worries. I also thank Dr. Ari Ingimundarson for helping me during the first days, his advice was always valuable. And, I would also like to thank all the people that is part or have been part of the Aula Boeing office, especially Rosa, Miquel, Fernando, Juli and Clàudia, for creating a really friendly atmosphere.

During my PhD, I had the opportunity to enjoy two stays abroad. I thank Stéphane Ploix for welcoming me into his group in Grenoble (France) and Erik Frisk for receive me in Linköping (Sweden) and helping me out with my research.

Lastly, I also want to thank Marta and my family for their support, patience and encouragement during these years. Without them, this thesis would not have been a real fulfilment.

Albert Rosich Oliva

Terrassa, December 2010

Notation

Consistency Diagnosis Notation

$\mathbf{C}$	The set of all process components.
$\mathbf{B}$	The set of all behavioral modes in a process.
$\mathbf{M}$	The set of all process model equations.
$\mathbf{X}$	The set of all unknown variables in a model.
$\mathbf{Y}$	The set of all known variables in a model.
$\mathbb{X}$	The domain of unknown variables $\mathbf{X}$.
$\mathbb{Y}$	The domain of known variables $\mathbf{Y}$.
$\mathcal{O}(M)$	The consistent observations of a model M.
$\mathsf{assump}(M)$	The set of behavioral modes supported by M.
$\mathcal{D}(\mathbf{y})$	The diagnosis statement inferred from the known variables $\mathbf{y}$.

Bipartite Graph Theory Notation

G	A bipartite graph.
$\|U\|$	The cardinality of a set U.
$\mathcal{M}_G$	A matching in the graph G.
$\mathcal{M}_G^{max}$	A maximum matching in the graph G.
$\nu(G)$	The size (cardinality) of a maximum matching in G.
$\mathcal{M}_G^V$	A complete matching of the vertex set V.
$\mathcal{M}_G^p$	A perfect matching in G.
$\partial^V \mathcal{M}$	The set of vertices in V incidents to the edges in $\mathcal{M}$.
$p_0(U)$	The surplus function.

Structural Model Notation

G^+	The over-determined part of the model represented by G.

G^0	The just-determined part of the model represented by G.
G^-	The under-determined part of the model represented by G.
$\varphi_s(M)$	The structural redundancy of a model M.
$\text{var}_Z(M)$	The variables in Z contained in some equation in M.
$\text{var}_M(Z)$	The equations in M containing some variable in Z.
M^+	The over-determined set of equations in M.
M^0	The just-determined set of equations in M.
M^-	The under-determined set of equations in M.

Sensor Placement Notation

$\mathbf{\Omega}$	The set of all possible MSO sets.
$\mathbf{F}$	The set of faults.
F_D	The set of detectable faults.
$\mathbb{F}_I^1$	The non-symmetric fault isolability characterisation.
$\mathbb{F}_I^2$	The symmetric fault isolability characterisation.
$F_{D_{max}}$	The set of maximum detectable faults.
$\mathbb{F}_{I_{max}}^1$	The set of maximum isolable faults using non-symmetric characterisation.
$\mathbb{F}_{I_{max}}^2$	The set of maximum isolable faults using symmetric characterisation.
$\mathbf{S}$	The set of all candidate sensors.
$\mathbf{S}'$	The set of all redundant sensors.
S_0	The set of useful sensors.
$C(S)$	The cost of a sensor configuration S.
$[\omega]_S$	An equivalent class of MSO sets containing all the sensor equations of S.
Ω_S	The set of MSO sets generated with S sensors in the model.
$\mathbf{q}$	The binary vector indicating the sensors chosen for installation.
$\mathbf{W}$	The binary matrix relating MSO sets and sensors.
$\mathbf{V}$	The binary matrix relating MSO sets and process faults.
$\boldsymbol{\rho}$	The non-linear MSO set selector vector.
$\boldsymbol{\lambda}$	The linear MSO set selector vector.
$\mathcal{C}_k^n$	The k-combination number of a set with n elements.
$\mathcal{P}_k^n$	The k-permutation number of a set with n elements.

Causal Computation Notation

A_L	The set of edges indicating a linear relation.
$A_\times$	The set of edges indicating a causal (but non-linear) relation.
A_Δ	The set of edges indicating a linear relation.
$\mathcal{X}_L$	A set of linear computable variables.
$\mathcal{E}_L$	A set of equations to compute linear variables.
$\mathcal{X}_\times$	A set of causally (but non-linear) computable variables.
$\mathcal{E}_\times$	A set of equations to compute causal variables.
$\mathcal{X}$	The set of all causally computable variables.
$\mathcal{E}$	The causally computable part of a model.
$\mathbb{F}_\mathcal{D}$	The set of causally detectable faults.
$\mathbb{F}^1_\mathcal{I}$	The set of causally isolable faults.
$\mathbf{\Omega}_c$	The set of all causally computable MSO sets.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Nowadays, there is no doubt that diagnosis systems are of great importance for any complex process. Neglecting them or designing bad diagnosis systems can have a serious impact on process economy, product quality, safety, productivity and pollution level. Automatic fault diagnosis arises in the beginning of the 70's with the advent of the on-line computer control of processes. Up to now, there have been enormous advances in this field and many works have been and are still being published. However, in many industrial processes the diagnosis tasks are currently performed by hand or using naive techniques.

The field of fault diagnosis covers a wide variety of disciplines and use a large number of techniques. A first classification can be done by considering the *a priori* knowledge of the system to diagnose (Vekatasubramanian et al., 2003). When knowledge concerns past experiences in the process, it is called process history based diagnosis. On the other hand, when this knowledge develops from an explicit formal model, it is known as model based diagnosis.

The model based diagnosis task can be roughly summarised as diagnosing faults in the system by comparing the set of system observations (mainly controlled and measured signals) with a model. However, there is no general framework for such diagnosis systems, and a suitable approach basically depends on the system model and the faults to diagnose.

Processes can suffer from faults or malfunctions in any time during their working life. A general diagnosis system works in parallel with the process. The diagnosis system inputs are the observations provided by the process measurements and the control or operating signals, see Figure 1.1. From these observations, the diagnosis system tries to infer which behaviour described by the model is actual in the process. This usually implies the explicit or implicit computation of the values of the unknown model vari-

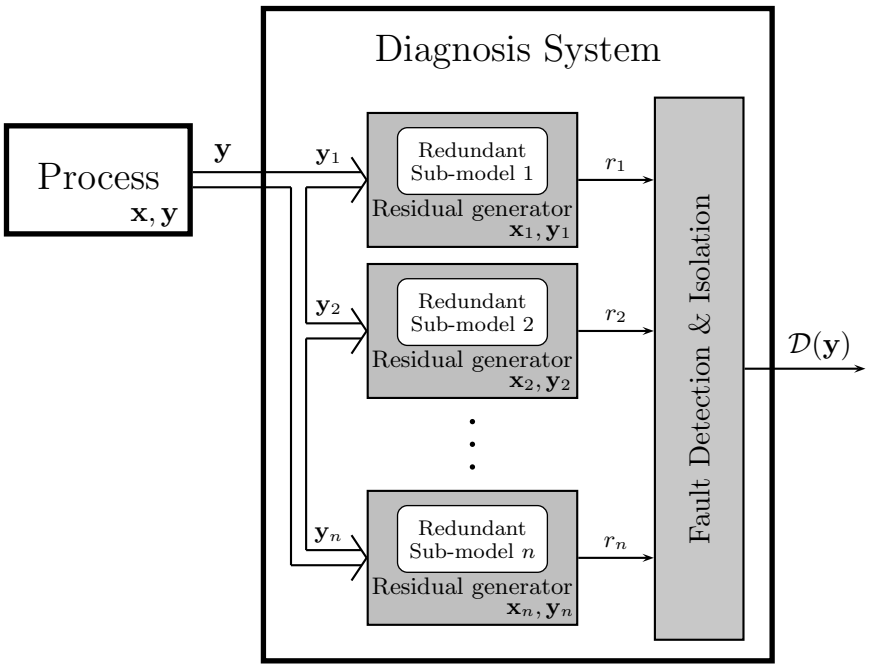


Figure 1.1: Diagnosis system diagram.

ables. However, more observations are needed to diagnose faults than the required ones for computing the unknown variables (i.e. the model must contain analytical redundancy). Hence, the expected behaviour computed with the model is used to validate the actual process behaviour. This is known in the diagnosis community as *consistency checking*.

Analytical redundancy is fundamental for the task of detecting fault occurrences, i.e. *fault detection*. Moreover, it is also important for the task to distinguish which faults are occurring in the system and which are not, i.e. *fault isolation*. In general, the more redundancy the model has, the better diagnosis performance can be achieved. An intuitive explanation is that a model with a high degree of redundancy can be split into several redundant sub-models, where each one of these sub-models can be used to diagnose small components of the process, improving fault detection and isolation.

Analytical redundancy is highly dependent on the number of observations. Furthermore, the diagnosis task is often made difficult due to in-

sufficient, incomplete or useless process observations. The problem on how to obtain the required process observations has been barely treated in the field of diagnosis, despite their importance in the fault diagnosis task. Till now, and before the work developed in the framework of this thesis, just few works had been dedicated to this problem, (Basseville et al., 1987), (Raghuraj et al., 1999), (Bagajewicz, 2000), (Travé-Massuyès et al., 2006). The present dissertation specifically focuses on analysing which sensors should be installed in a process in order to improve the diagnosis capabilities of a model based diagnosis system and at the same time facilitate its design.

1.1 Motivation

Typically, the diagnosis system design consists in first studying the process and identifying a set of faults or behaviour modes to diagnose. Then, a diagnosis model has to be constructed from the process knowledge and the set of faults. This model should be as detailed as possible and should also contain fault information, if available. Once the model is developed, the proper diagnosis system design begins. Since many approaches exist in the literature, the most suitable one has to be chosen according to the type of the model and the faults to diagnose. Finally, the diagnosis system is tested in simulation and, whenever possible, in the real process.

During the diagnosis system design or even afterwards, one may realise that the required diagnosis specifications are not fulfilled (i.e. the diagnosis system does not diagnose the faults as expected). One possible solution involves trying to improve the methods or/and the algorithms on which the diagnosis system is based. Actually, there are a large number of publications devoted to obtaining a better diagnosis system for a given technique. Another possibility involves improving the model. However, developing a good model is usually a hard task. So, it becomes a drawback in many industrial applications. Finally, it is also possible to improve the diagnosis performance by providing the process with extra components in order to increase the redundancy.

Sensors are the system components by means of which process observations are obtained. Therefore, by installing extra sensors in the process, the number of observations increases and consequently so does the analytical redundancy degree of the model. There may be some processes where installing extra sensors is not indicated, (e.g. insufficient space or extra weight in aircrafts, the economical impact on electronic devices). However, for many industrial processes where the presence of a non-diagnosed fault can

cause serious problems, the installation of extra sensors that improve fault diagnosis is fully recommended.

The study of determining which sensors are needed to achieve certain specifications is called *sensor placement* analysis. In fact, there are some research areas where the sensor placement problem has been widely studied (e.g. observability in the control field). This is not the case for the diagnosis field, where there are just few isolated works. However, this topic has gained an increasing interest during the last few years.

The sensor placement analysis for model-based diagnosis will be the main topic on this thesis. Nevertheless, model-based diagnosis concerns a large variety of models and techniques. Fitting all of them in a common framework suitable for the sensor placement analysis is not feasible. Therefore, only a sub-class of models will be considered. Such a sub-class will be justified by the practical case study.

1.1.1 Practical Motivation based on the Case Study

To develop the results obtained and at the same time to test them, a practical case based on a fuel cell stack system will be considered. The model used was developed by (Pukrushpan et al., 2004) and currently has become a reference model in the area of fuel cell stack control. This model comprises several detailed subsystems required to operate a fuel cell stack. Because of this, and the large number of non-linear equations (e.g. look-up tables, piecewise functions, maps, etc) involved in the model, it is considered a complex system. The fuel cell stack system and its model will be thoroughly described in Section 7.3.

Due to the fact that the model has strong non-linearities and that the system may work at several operating points, a diagnosis approach based on a linearised model is not indicated. On the other hand, structural analysis offers a suitable framework to handle such complex systems, since just the structure of the model is considered. Furthermore, structural analysis is based on efficient graph tools which allow to work with such models. Structural model-based diagnosis has been extensively studied by several authors (Blanke et al., 2006), (Krysander, 2006). Therefore, from the diagnosis point of view, there exists a well-grounded framework to cope with this kind of models. For these reasons, the present work is confined to structural model based approaches.

The fuel cell stack system considered in this thesis had originally a few number of sensors installed, which derives in a low level of redundancy and poses serious difficulties for achieving a good diagnosis performance. How-

ever, there are several process variables that could be easily measured which makes appropriate to apply the sensor placement analysis to it. Moreover, works related to fuel cell diagnosis are rather lacking in the literature. This gives another good motivation to work on the diagnosis of this kind of systems.

1.2 Main objectives

As it was pointed out in the previous section, the main goal of this thesis is to develop new methodologies for solving the sensor placement problem in the fault diagnosis field. These methods must be as general as possible, which means that they can be applied not only to the fuel cell stack system, but also to any other complex system.

These methodologies will derive in algorithms capable of handling large-scale structural models where the number of candidate sensors to be installed is typically large. Therefore, computational efficiency of the developed algorithms must be also taken into account.

Due to the fact that the set of solutions could become large (i.e. there may exist a lot of possible sensor configurations that solve the problem), the sensor placement problem will be formulated as an optimisation problem. Only best cost solutions will be sought. Thus, the term *optimal sensor placement* will be used.

Different features will be considered to evaluate and classify the developed algorithms. Such features will be based on the following criteria:

- The need to generate redundant sub-models by the algorithm.
- The flexibility on the diagnosis specifications considered by the algorithm.
- The efficiency of the optimal search strategy used by the algorithm.

Generating redundant sub-models is a computing time demanding task. Therefore, algorithms that do not require them are desirable. Nevertheless, it is easier to define diagnosis specifications from these redundant sub-models. This implies that redundant sub-models are suitable to handle several diagnosis properties for the sensor placement analysis. On the other hand, the characterisation of certain diagnosis specifications by means of model properties and not from redundant sub-models will also be investigated. Fault detectability and isolability will be the basic diagnosis specifications that will be considered in all the approaches presented in this thesis.

The sensor placement problem can be viewed as a combinatorial problem (i.e. find a sensor combination that fulfils some diagnosis specifications). This entails special considerations in the optimal search implementation, which needs to be carefully studied.

Another important goal is that every proposed method should be able to handle both process and sensor faults. Process faults are the initial faults defined on the process components before the sensor placement analysis is performed, whereas sensor faults concern faults on the extra sensors, chosen by the sensor placement analysis. Furthermore, how redundant sensors can be efficiently handled by each algorithm will be also taken into consideration.

After the development of some generic algorithms, the optimal sensor placement problem for the fuel cell stack system will be solved. This will lead to an opportunity to test the algorithms and draw conclusions from a practical point of view. The idea is to prove the importance of the sensor placement analysis in the design of a diagnosis system.

1.3 Outline

The manuscript is organised in eight chapters. Chapter 2 presents background theory. Chapter 3 introduces a formal framework for the sensor placement analysis. Next, three different approaches to solve the sensor placement problem for fault diagnosis are proposed in Chapters 4, 5 and 6. Chapter 7 is devoted to solve the sensor placement problem for the fuel cell stack system. Finally, conclusions are drawn in Chapter 8. The outline of this thesis is next briefly detailed:

- *Chapter 2*: A review of the main concepts needed through this thesis is given. First, the diagnosis based on consistency checking is summarised. Then, the basic issues about structural model representation and graph theory are introduced. Finally, minimal redundant model computation is also detailed, since it will be used in further chapters.

- *Chapter 3*: A framework to investigate the sensor placement problem based on structural models is developed. This framework allows to properly handle process and sensors faults, as well as redundant sensors. The study on how redundant sensors affect the diagnosis capabilities of the system is also investigated. Finally, the sensor placement problem for fault diagnosis is formally presented.

- *Chapter 4*: A methodology to solve the sensor placement problem is presented. In this approach, the required redundant sub-models are

provided at each iteration of the sensor configuration search. This is performed by incrementally computing the set of redundant sub-models.

- *Chapter 5*: In order to improve the search strategy presented in the previous chapter, the sensor placement problem is solved here by means of binary integer programming. Two formulations are presented. The first formulation involves non-linear constraints, whereas the second one just uses linear inequality constraints. Extensions for efficiently handling redundant sensors are proposed. Finally, binary integer programming is also used for an optimal selection of the useful redundant sub-models.

- *Chapter 6*: In this chapter, the computation of the known variables in the residual generators is specially addressed. First, a new framework to handle causalities in the unknown variable computation is presented. Then, the sensor placement problem for fault diagnosis is introduced under the causal computation framework. Two main algorithms are developed in this chapter. The first one allows to solve the sensor placement problem within the causal framework for both process and sensors faults. The second algorithm, based on specific heuristics, is limited to process faults, but its efficiency is better than for the former algorithm.

- *Chapter 7*: The sensor placement problem is applied to a fuel cell stack system. First, a detailed model of the process is presented. Then, the sensor placement problem within the causal framework is solved. Based on the obtained solution, several residuals are implemented and tested. This chapter presents a methodology to design diagnosis systems involving: structural model extraction from the analytical model, sensor placement analysis, redundant sub-model computation and residual generation.

- *Chapter 8*: The main contributions of this thesis are first summarised. Next, the three sensor placement approaches presented in this thesis are compared, and the main conclusions are drawn. Furthermore, some possible future works are suggested.

1.4 Related Publications

Several publications have been derived during the doctoral courses and the thesis development. A list of published works where the thesis author has contributed to is next presented:

- A. Rosich, E. Frisk, J. Åslund, R. Sarrate, F. Nejjari (2010). Fault Diagnosis Based On Causal Computations. *IEEE Transactions on Systems, Man, and Cybernetics–Part A*. Under review.
- A. Rosich, A. A. Yassine, S. Ploix (2010). Efficient Optimal Sensor Placement for Structural Model Based Diagnosis. *21th International Workshop on Principles of Diagnosis, DX'10*. Portland, USA.
- F. Nejjari, R. Sarrate, A. Rosich (2010). Optimal Sensor Placement For Fuel Cell System Diagnosis using BILP Formulation. *18th Mediterranean Conference on Control and Automation*. Marrakech, Morocco.
- A. A. Yassine, A. Rosich, S. Ploix (2010). An Optimal Sensor Placement Algorithm taking into account Diagnosability Specifications. *17th IEEE International Conference on Automation, Quality and Testing, Robotics,*. Cluj-Napoca, Romania.
- A. Rosich, E. Frisk, J. Åslund, R. Sarrate, F. Nejjari (2009). Sensor Placement for Fault Diagnosis Based On Causal Computations. *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Process, Safeprocess'09*. Barcelona, Spain.
- A. Rosich, F. Nejjari, R. Sarrate (2009). Fuel Cell System Diagnosis based on a Causal Structural Model. *7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Process, Safeprocess'09*. Barcelona, Spain.
- A. Rosich, R. Sarrate, F. Nejjari (2009). Optimal Sensor Placement for FDI using Binary Integer Linear Programming. *20th International Workshop on Principles of Diagnosis, DX'09*. Stockholm, Sweden.
- A. Rosich, R. Sarrate, V. Puig, T. Escobet (2007). Efficient Optimal Sensor Placement for Model-Based FDI using an Incremental Algorithm. *Proc. 46th IEE Conference on Decision and Control*. New Orleans, USA.

- R. Sarrate, V. Puig, T. Escobet, A. Rosich (2007). Optimal Sensor Placement for Model-based Fault Detection and Isolation. *Proc. 46th IEE Conference on Decision and Control*. New Orleans, USA.

- V. Puig, A. Rosich, C. Ocampo, R. Sarrate (2007). Fault-Tolerant Explicit MPC of PEM Fuel Cells. *Proc. 46th IEE Conference on Decision and Control*. New Orleans, USA.

- A. Rosich, V. Puig, J. Quevedo (2006). Fault-Tolerant Constrained MPC of PEM Fuel Cells. *IAR Annual Meeting*. Université Henri Poincaré, Nancy, France.

Chapter 2

Review of Background Theory

In this chapter, some theoretical fundamentals are reviewed, focusing on those issues that have a special interest for the comprehension of this thesis. The reader who is acquainted with these topics can skip to the next chapter since no new contribution is here addressed.

In order to present a general framework suitable for sensor placement, the foundations of diagnosis based on consistency are firstly introduced. Then, the structural model is formally introduced, as the modelling representation used in this thesis. Since a structural model representation is a graph representation, some theoretical issues on graph theory are also introduced. Specifically those concerning bipartite graphs and the Dulmage-Mendelsohn decomposition. Finally, this Chapter focuses on how parts of the model can be selected to test consistency. In particular, the computation of Minimal Structural Overdetermined (MSO) sub-models is reviewed, as well as the existing works in the literature devoted to this issue.

Here, only basic concepts on these topics are covered. However, there exists a great number of works extending the topics presented in this chapter. The reader interested in further information is encouraged to resort to the given bibliography.

2.1 Consistency based Diagnosis Framework

The task of diagnosing faults can be explained, regardless of the implemented method, by using the notion of *consistency based diagnosis*. The framework here presented arises as a result of combining approaches from the artificial

intelligence (AI) community and the control theory community (also known as FDI). Main issues in the AI community (e.g. (De Kleer and Williams, 1987) and (Reiter, 1987)) deal with identifying those behaviour modes that may be present in the process, being the inference procedure its key point. On the other hand, FDI community issues (e.g. (Gertler, 1998) and (Chen and Patton, 1999)) are more focused on obtaining signals (called *residuals*) that are sensitive to faults, i.e. residual generator design. The basic ideas developed in this section were first introduced by Nyberg and Krysander (2003) and further detailed in Krysanders' thesis (Krysander, 2006). It is worth mentioning that an introduction to consistency based diagnosis can be also found in (Blanke et al., 2006).

In the area of diagnosis, a system is usually assumed to consist of a set $\mathbf{C}$ of *components*,

$$\mathbf{C} = \{c_1, c_2, c_3, \dots\}$$

For instance consider an air compressor system. It is possible to define some components such as the electric motor, the compressor box or the manifold of the outlet air flow. This allows us to assign different system descriptions to each component, so that the entire system model can be split into several component models. As it will be seen next, this facilitates the task of defining and representing faults in the system, as well as distinguishing between faults during the fault isolation task.

System variables interact with one or several components. A variable that only appears in one component is called *internal variable*, whereas a variable shared by several components and/or observable is called *external variable*. External variables connect a component to its outer world, it can be whether an other system components or other external systems (e.g. a control system or a diagnosis system).

A component $c \in \mathbf{C}$ is associated to one or several behaviour modes, $\mathbf{B}_c$. Typically, a component has at least the non-faulty behaviour mode, $\mathcal{NF}$, also known as "normal" behaviour mode for some authors. However, it is also possible to associate one behaviour mode, $\mathcal{F}_i$, for each fault defined in the component. It is assumed that the real behaviour mode of a component corresponds to exactly one of its associated behaviour modes in $\mathbf{B}_c$. Therefore, for convenience, an unknown-fault behaviour mode $\mathcal{UF}$ is defined, which covers any non-specified behaviour mode. The set of all behaviours modes associated to a component c is denoted as

$$\mathbf{B}_c = \{b_1, b_2, \dots, b_n\}$$

where each b_i represents a different behaviour mode of the component: $b_1 =$

$\mathcal{NF}$, $b_i = \mathcal{F}_{i-1}$ $(i = \{2, \ldots, n-1\})$ and $b_n = \mathcal{UF}$.

In a system, the different behaviour modes are characterised by means of relations among the system variables. So, depending on how system variables relate among them, the system will behave according to a given mode. Here, the term *equation* will be used to denote any expression defining a relationship between two or more system variables. Nevertheless, other authors use other equivalent terms as, for example, the term *relation* used in (Travé-Massuyès et al., 2006) or the term *constraint* used in (Ploix et al., 2005). These equations represent the *a priori* knowledge available for model-based diagnosis and can be expressed by a wide range of different classes: dynamic-algebraic equations, logic laws, look-up tables, etc. Thus, a system *model* can be defined by the set of equations that depend on the system variables and is denoted as

$$\mathbf{M} = \{e_1, e_2, e_3, \cdots\}$$

where e_i represents the i-th equation of the model. Therefore, a behaviour mode $b \in \mathbf{B}_c$ for a given component $c \in \mathbf{C}$ can be characterised by a subset of model equations, $M_b \subseteq \mathbf{M}$. M_b is called the behaviour model of b.

The idea of consistency-based diagnosis is to determine, from the set of model equation $\mathbf{M}$, which behaviour modes are *consistent* with the observations. Thus, for diagnosis purposes, it is important to distinguish between the set $\mathbf{Y}$ of known variables (or observations) and the set $\mathbf{X}$ of unknown variables. The set $\mathbf{Y}$ usually refers to controlled and/or measured variables.

Let $\mathbf{x}$ and $\mathbf{y}$ denote vectors of variables in $\mathbf{X}$ and $\mathbf{Y}$, respectively, and let $\mathbb{X}$ and $\mathbb{Y}$ be its domains. According to (Krysander, 2006), a subset of equations $M \subseteq \mathbf{M}$ is consistent with the observations $\mathbf{y} \in \mathbb{Y}$ if

$$\exists \mathbf{x} \in \mathbb{X} : M(\mathbf{x}, \mathbf{y}) \tag{2.1}$$

where $M(\mathbf{x}, \mathbf{y})$ means that all the equations in M are satisfied for the given values in $\mathbf{y}$. Analogously, the set of known variables consistent with M is defined as

$$\mathcal{O}(M) = \{\mathbf{y} \in \mathbb{Y} \mid \exists \mathbf{x} \in \mathbb{X} : M(\mathbf{x}, \mathbf{y})\} \tag{2.2}$$

In Figure 2.1 a graphical interpretation of the behaviour mode consistency is depicted. The different behaviour modes are represented as space subsets of the observation domain, $\mathbb{Y}$. For instance, considering the observation $\mathbf{y_1} \in \mathbb{Y}$ in the figure, it can be concluded that behaviour modes $\mathcal{F}_1$ and $\mathcal{F}_3$ are not consistent with $\mathbf{y_1}$, whereas behaviour modes $\mathcal{NF}$, $\mathcal{F}_2$ and

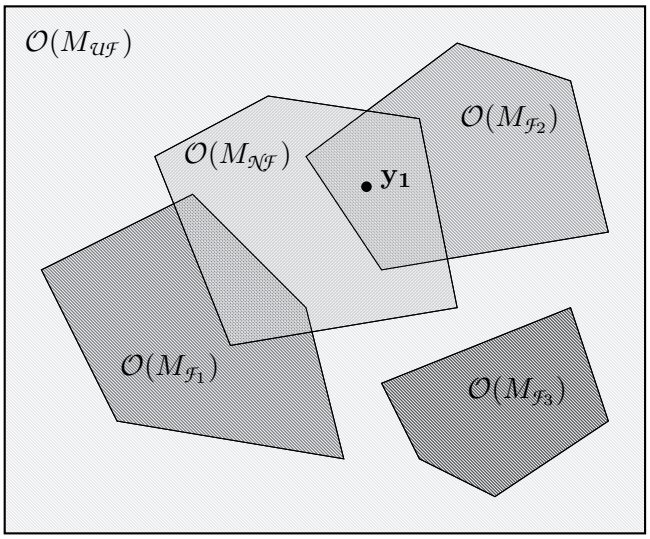


Figure 2.1: Graphical interpretation of the behaviours modes in the domain $\mathbb{Y}$.

$\mathcal{UF}$[1] are consistent with $\mathbf{y_1}$. For practical reasons, it is easier to draw a diagnostic statement when the behaviour mode is not consistent with the observations than when it is (Nyberg and Krysander, 2003).

To test consistency from a set of model equations, M, *analytical redundancy* in the model must exist. In other words, there must be observations such that the model is not consistent with:

$$\exists \mathbf{y} \in \mathbb{Y}, \forall \mathbf{x} \in \mathbb{X} : \neg M(\mathbf{x}, \mathbf{y}) \tag{2.3}$$

where $\neg M(\mathbf{x}, \mathbf{y})$ means that there is at least one equation in M that is not satisfied for the values in $\mathbf{y}$. Note that this is equivalent to

$$\exists \mathbf{y} \in \mathbb{Y} : \mathbf{y} \notin \mathcal{O}(M) \tag{2.4}$$

As said above, any behaviour mode b is characterised by a subset of model equations, M_b. Thus, if the component is in a behaviour mode b then it is assumed that M_b is consistent with any possible set of system observations. Given a model equation $e \in \mathbf{M}$, the operator $\mathsf{assump}(e)$ (Krysander,

[1]The space subset related with the behaviour mode $\mathcal{UF}$ in Figure 2.1 involves the whole observation domain $\mathbb{Y}$, despite it is difficult to see in the figure.

2006) will denote those behaviour modes that are assumed to be consistent with e,

$$\mathsf{assump}(e) = \{b \mid e \in M_b\} \tag{2.5}$$

The $\mathsf{assump}(e)$ operator can be extended to also denote which behaviour modes are assumed to be consistent with a set M of model equations,

$$\mathsf{assump}(M) = \bigcap_{e \in M} \mathsf{assump}(e) \tag{2.6}$$

This implies that, given an inconsistent sub-model M, the conclusion drawn for the diagnosis system is that $\mathsf{assump}(M)$ behaviour modes are inconsistent with the observations. Therefore, the remaining behaviour modes $\mathbf{B}_c \setminus \mathsf{assump}(M)$ are consistent with the observations (i.e. the behaviour modes in $\mathbf{B}_c \setminus \mathsf{assump}(M)$ can explain the observed behaviour of the component). To show how the diagnosis reasoning works within this framework, the following example is proposed.

Example 2.1. Consider a component c with the following behaviour modes:

$$\mathbf{B}_c = \{\mathcal{NF}, \mathcal{F}_1, \mathcal{F}_2, \mathcal{UF}\}$$

The model of the component consists of three equations, $M = \{e_1, e_2, e_3\}$, which describe the different behaviour modes as follows:

$$\begin{aligned} M_{\mathcal{NF}} &= \{e_1, e_2, e_3\} \\ M_{\mathcal{F}_1} &= \{e_2, e_3\} \\ M_{\mathcal{F}_2} &= \{e_1, e_3\} \\ M_{\mathcal{UF}} &= \emptyset \end{aligned}$$

This can be interpreted as equations e_1, e_2 and e_3 describing the free fault component behaviour. A fault related with $\mathcal{F}_1$ affects equation e_1, therefore equations e_2 and e_3 remain consistent in the presence of such fault. The same interpretation can be done for a fault related with $\mathcal{F}_2$, but now the affected equation is e_2, i.e. e_1 and e_3 are consistent with $\mathcal{F}_2$. Finally, note that no equation describes the unknown fault behaviour mode, $\mathcal{UF}$, indicating that any equation may become inconsistent in the presence of an unknown fault.

Now, assume that the sub-model $M' = \{e_1, e_2\}$ becomes inconsistent, which means that $\mathsf{assump}(M')$ is also inconsistent,

$$\mathsf{assump}(M') = \mathsf{assump}(e_1) \cap \mathsf{assump}(e_2) = \{\mathcal{NF}, \mathcal{F}_2\} \cap \{\mathcal{NF}, \mathcal{F}_1\} = \mathcal{NF} \tag{2.7}$$

Therefore the diagnosis inference is that the component is behaving as one of the remaining modes,

$$\mathbf{B}_c \setminus \mathsf{assump}(M') = \{\mathcal{F}_1, \mathcal{F}_2, \mathcal{UF}\} \tag{2.8}$$

□

Improving the diagnosis task in order to detect and isolate faults is possible by testing consistency on several sub-models, $M_i \subseteq \mathbf{M}$. Given a set of sub-models suitable to test consistency, the diagnosis statement, $\mathcal{D}(\mathbf{y})$, is presented as a function from the set of all possible observations $\mathbf{y}$ to all possible combinations of $\mathbf{B}_c$,

$$\mathcal{D}(\mathbf{y}) : \mathbb{Y} \rightarrow \mathcal{P}(\mathbf{B}_c) \tag{2.9}$$

where $\mathcal{P}(\mathbf{B}_c)$ denotes the power set of $\mathbf{B}_c$, and $\mathcal{D}(\mathbf{y})$ is computed as

$$\mathcal{D}(\mathbf{y}) = \mathbf{B}_c \setminus \Big(\bigcup_{M_i \text{ is inconsistent}} \mathsf{assump}(M_i) \Big) \tag{2.10}$$

This shows that the diagnosis performance strongly depends on the sub-models M_i to test consistency. How these sub-models are generated from $\mathbf{M}$ is explained in Section 2.4.

Finally, remark that this framework has been here introduced regarding the behaviour modes of a component. However, the framework can be easily extended to cope with several components in a system and several behaviours modes per component (Krysander, 2006).

2.2 Structural Model

The analysis of the model structure has been widely used in the area of model-based diagnosis (Blanke et al., 2006). Next, it is explained what a *structural model* is and how it can be represented.

The structural model of a system is an abstraction of the analytical model. In fact, the structural model is a coarse model simplification since only the relation between variables and equations is taken into account, neglecting the mathematical expression of this relation. Due to its simple description, it cannot be ensured that the diagnosis performance obtained from structural models will hold for the practical case. However if the required diagnosis performance is not fulfilled for a structural model, neither it will be for the practical case (i.e. only best case results can be computed). It is usually assumed that all the equations in the model are *compatible* and *independent* (Blanke et al., 2006).

Assumption 2.1. *The set of solutions of a model of M equations is not empty, i.e.*

$$\exists(\mathbf{x}, \mathbf{y}) \in \mathbb{X} \times \mathbb{Y} : M(\mathbf{x}, \mathbf{y}) \tag{2.11}$$

We say that the model equations in M are compatible.

Assumption 2.2. *Given an observation $\mathbf{y} \in \mathbb{Y}$, two model equations $e_i, e_j \in M$ do not define the same set of solutions, i.e.*

$$\{\mathbf{x} \in \mathbb{X} \mid e_i(\mathbf{x}, \mathbf{y})\} \not\subseteq \{\mathbf{x} \in \mathbb{X} \mid e_j(\mathbf{x}, \mathbf{y})\} \tag{2.12}$$

We say that both model equations are independent.

One advantage is that the structural model is suitable for an early stage of the system design, when the precise model expressions are not known yet, but it is possible to determine which variables are related to each equation. Furthermore, the diagnosis analysis based on structural models are performed by means of graph-based methods which have no numerical problems and are more efficient, in general, than analytical methods.

The structural model (or the structure) of a system is commonly represented by a bipartite graph according to the following definition.

Definition 2.1 (Structural model)**.** Given a set of model equations, $\mathbf{M}$, that depend on variables $\mathbf{Z} = \{\mathbf{X} \cup \mathbf{Y}\}$, the *structural model* is defined as a bipartite graph $G = (\mathbf{M}, \mathbf{Z}; \mathbf{A})$ where $\mathbf{M}$ and $\mathbf{Z}$ are the set of vertices and $\mathbf{A}$ is the set of edges such that

$$\mathbf{A} = \{(e, z) \mid \text{variable } z \in \mathbf{Z} \text{ appears in equation } e \in \mathbf{M}\}$$

□

Given a structural model $G = (\mathbf{M}, \mathbf{Z}; \mathbf{A})$, it is possible to refer to a sub-model of $M \subseteq \mathbf{M}$ equations and $Z \subseteq \mathbf{Z}$ variables, by its corresponding bipartite sub-graph $G' \subseteq G$, defined as

$$G' = (M, Z; A')$$

where the set $A' \subseteq \mathbf{A}$ of edges is implicitly defined from the sets M and Z as

$$A' = \{(e, z) \in \mathbf{A} \mid e \in M, z \in Z\}$$

For instance, the bipartite sub-graph $(\mathbf{M}, \mathbf{X}; A') \subseteq G$ represents the structural model when just unknown variables are considered. Next example shows the extraction of the structural model from the analytical model equations and its bipartite graph representation.

Example 2.2. An air compressor model driven by an electric motor is used in this example. The analytical model equations that describe the free fault behaviour of the compressor are

$$e_1 \;:\; v = k_v \cdot \omega_c + R \cdot i \tag{2.13a}$$
$$e_2 \;:\; J\dot{\omega}_c = k_t \cdot i - B \cdot \omega_c - \tau \tag{2.13b}$$
$$e_3 \;:\; W = h(\omega_c, p_{in}, p_{out}, T_{in}) \tag{2.13c}$$
$$e_4 \;:\; \eta = \text{Lookup-Table}(W, p_{in}, p_{out}) \tag{2.13d}$$
$$e_5 \;:\; \tau = C_p \frac{T_{in} \cdot W}{\eta \cdot \omega_c} \left(\left(\frac{p_{out}}{p_{in}}^{\frac{\gamma-1}{\gamma}} \right) - 1 \right) \tag{2.13e}$$

The equations involving the electric motor are e_1 and e_2, where k_v, R, J, k_t and B are model parameters like the electrical resistance, the motor inertia or the friction coefficient, etc. The variables are: the motor voltage, v, the motor current, i, the motor angular speed, ω_c, and the compressor torque, τ. The $\dot{\omega}_c$ denotes the derivative of the angular speed, i.e. $\dot{\omega}_c = d\omega_c/dt$. The compressor box is modelled by the equations e_3, e_4 and e_5 where C_p and γ are parameters whereas the air flow, W, the input and output pressures, p_{in} and p_{out}, the input air temperature, T_{in}, and the compressor efficiency, η, are system variables. In equation e_3 all the physics that involve the air flow computation are included in function h to simplify the notation. The two vertex sets of the structural model corresponding to the air compressor are

$$\begin{aligned} \mathbf{M} &= \{e_1, e_2, e_3, e_4, e_5\} \\ \mathbf{Z} &= \{\omega_c, i, \tau, W, \eta, v, p_{in}, p_{out}, T_{in}\} \end{aligned} \tag{2.14}$$

and the bipartite graph, showing the set of edges, is depicted in Figure 2.2. □

Any bipartite graph, and consequently any structural model, can be represented in matrix form by means of the *biadjacency matrix*. Let the two vertex sets, $\mathbf{M} = \{e_1, e_2, \dots\}$ and $\mathbf{Z} = \{z_1, z_2, \dots\}$, be explicitly ordered in a bipartite graph $G = (\mathbf{M}, \mathbf{Z}; \mathbf{A})$, then a biadjacency matrix $Q = [q_{i,j}]$ is defined as

$$q_{i,j} = \begin{cases} 1, & \text{if } (e_i, z_j) \in \mathbf{A} \\ 0, & \text{if otherwise} \end{cases} \tag{2.15}$$

Example 2.3. Following with the Example 2.2, consider the bipartite graph in Figure 2.2 and the vertex sets ordered as in (2.14). The corresponding biadjacency matrix is shown in Table 2.1 where the ones have been replaced

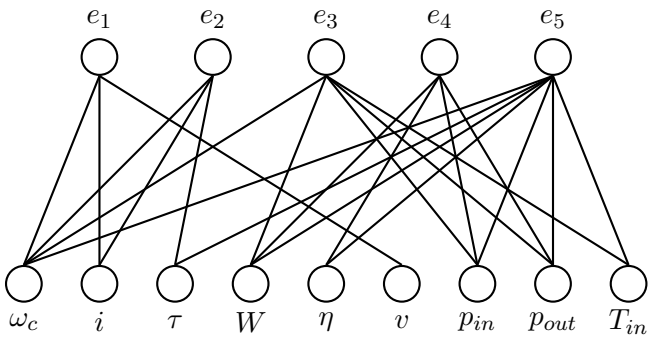


Figure 2.2: Bipartite graph of the structural model of the compressor.

	ω_c	i	τ	W	η	v	p_{in}	p_{out}	T_{in}
e_1	×	×				×			
e_2	×	×	×						
e_3	×			×			×	×	×
e_4				×	×		×	×	
e_5	×		×	×	×		×	×	×

Table 2.1: Biadjacency matrix of the structural model of the compressor.

by "×" symbols and the zeros have been omitted in order to make the matrix clearer.

Note that, assuming that the set of variables is partitioned into known and unknown variables as

$$\begin{aligned} \mathbf{X} &= \{\omega_c, \tau, \eta\} \\ \mathbf{Y} &= \{i, W, v, p_{in}, p_{out}, T_{in}\} \end{aligned}$$

the sub-model $(\mathbf{M}, \mathbf{X}; A') \subset G$ corresponds to the biadjacency sub-matrix consisting of all the equation-rows and the first, third and fifth variable-columns in Table 2.1. □

In terms of graph theory, the set of variables Z contained within the set of equations M is the corresponding set of vertices Z adjacent to the vertices M, and it is denoted as

$$\mathrm{var}_Z(M) = \{z \in Z \mid \exists e \in M : (e, z) \in \mathbf{A}\} \tag{2.16}$$

Analogously, the set M of equations that contains any variable in Z is denoted as

$$\mathrm{equ}_M(Z) = \{e \in M \mid \exists z \in Z : (e, z) \in \mathbf{A}\} \tag{2.17}$$

Example 2.4. From expressions (2.16) and (2.17), and according to the compressor model introduced in the previous Examples 2.2 and 2.3, the unknown variables contained in the electric motor sub-model are

$$\mathrm{var}_{\mathbf{X}}(\{e_1, e_2\}) = \{\omega_c, \tau\}$$

and the compressor model equations that depend on the compressor torque, τ, and the compressor efficiency, η, are

$$\mathrm{equ}_{\mathbf{M}}(\{\tau, \eta\}) = \{e_2, e_4, e_5\}$$

□

In structural model-based fault diagnosis, only sub-models involving unknown variables are of interest for the diagnosis analysis (Blanke et al., 2006). For this reason, we will henceforth assume that the structural model of a set of equations $M \subseteq \mathbf{M}$ is $(M, var_{\mathbf{X}}(M), A')$. Both notations will be used interchangeably.

2.3 Graph Theoretical Preliminaries

This section introduces some basic fundamentals on graph theory, specially those concepts used throughout this thesis. Principal definitions are given, as well as notation concerning graph properties. An important part of this section is devoted to formally present the *Dulmage-Mendelsohn decomposition*.

In graph theory and also in structural model-based diagnosis, the concept of *matching* in a graph is widely used. Let $G = (V; E)$ be a graph where V is the set of vertices and E is the set of edges, a *matching* is a set of edges $\mathcal{M}_G \subseteq E$ where no two edges share a common vertex. The size (cardinality) of a matching, $\mathcal{M}$, denoted as $|\mathcal{M}|$, is the number of edges within the matching. The set of vertices in V incidents to the edges in

$\mathcal{M}$ is denoted by $\partial^V \mathcal{M}$, and $|\partial^V \mathcal{M}|$ denotes the cardinality of such a set. Therefore, in a bipartite graph $G = (V_1, V_2; E)$, $\mathcal{M}_G \subseteq E$ is a matching if and only if

$$|\mathcal{M}_G| = |\partial^{V_1} \mathcal{M}_G| = |\partial^{V_2} \mathcal{M}_G|$$

A vertex $v \in V$ is *covered* by a matching $\mathcal{M}$ if $v \in \partial^V \mathcal{M}$. An uncovered vertex is called a *free* vertex.

A matching $\mathcal{M}_G^{max}$ is called a *maximum matching* or a *maximum size matching* if it is a matching with the largest cardinality. The size of a maximum matching in G is denoted by $\nu(G)$. An edge is called *admissible* if it is contained in some maximum matching. A matching $\mathcal{M}_G^{V_1}$ (respectively $\mathcal{M}_G^{V_2}$) of a bipartite graph $G = (V_1, V_2; E)$ is a *complete matching* of V_1 into V_2 (respectively V_2 into V_1) if and only if

$$\partial^{V_1} \mathcal{M}_G^{V_1} = V_1 \quad \text{(respectively, } \partial^{V_2} \mathcal{M}_G^{V_2} = V_2)$$

Finally, a matching $\mathcal{M}_G^p$ of a bipartite graph is a *perfect matching* if it is a complete matching of V_1 into V_2 and it is, at the same time, a complete matching of V_2 into V_1, i.e.

$$|\mathcal{M}_G^p| = |V_1| = |V_2|$$

Recently, a special bipartite graph decomposition has been used in structural model-based diagnosis. This decomposition is known as the *Dulmage-Mendelsohn decomposition* (Dulmage and Mendelsohn (1958), Murota (2000)), also called DM-decomposition for short. Given a bipartite graph $G = (V_1, V_2; E)$, the DM-decomposition canonically decomposes the graph in a family of sub-graphs:

$$G_k = (V_{1_k}, V_{2_k}; E_k) \qquad (k = 0, 1, \cdots, b, \infty) \tag{2.18}$$

Next, the procedure to obtain such sub-graphs is explained, according to (Murota, 2000). Firstly, let $\Gamma(U)$ denotes the set of vertices in V_2 adjacent to some vertex[2] in $U \subseteq V_1$,

$$\Gamma(U) = \{v \in V_2 \mid \exists u \in U : (u, v) \in E\} \tag{2.19}$$

Then, the following sub-modular function called the *surplus function* (Lovász and Plummer, 1986), is defined as

$$p_0(U) = |\Gamma(U)| - |U| \tag{2.20}$$

[2]The $\Gamma(U)$ operator is equivalent to the $\text{var}_{V_2}(U)$, introduced in Section 2.2, for V_1 being the set of equations and V_2 being the set of variables. However, here the notation used by Murota (2000) is kept.

From this surplus function, the subsets V_{1_k} in the DM-decomposition (2.18) are computed by first determining the family of the minimizers,

$$\mathcal{L}_{min}(p_0) = \{U \subseteq V_1 \mid p_0(U) \leq p_0(W), \forall W \subseteq V_1\} \tag{2.21}$$

and then by looking at any *maximal ascending chain* of $\mathcal{L}_{min}$,

$$U_0 \subset U_1 \subset \cdots \subset U_{b-1} \subset U_b \tag{2.22}$$

The partition on the vertex set V_1 is thereby computed from the sets in the maximal ascending chain according to

$$\begin{aligned} V_{1_0} &= U_0 && \text{(2.23a)} \\ V_{1_k} &= U_k \setminus U_{k-1} \quad (k = 1, \cdots, b) && \text{(2.23b)} \\ V_{1_\infty} &= V_1 \setminus U_b && \text{(2.23c)} \end{aligned}$$

The subsets V_{2_k} are obtained from the subsets V_{1_k} by applying the adjacently operator defined in (2.19). Thus, the partition on the vertex set V_2 is computed as

$$\begin{aligned} V_{2_0} &= \Gamma(U_0) && \text{(2.24a)} \\ V_{2_k} &= \Gamma(U_k) \setminus \Gamma(U_{k-1}) \quad (k = 1, \cdots, b) && \text{(2.24b)} \\ V_{2_\infty} &= V_2 \setminus \Gamma(U_b) && \text{(2.24c)} \end{aligned}$$

Finally, the subsets E_k of edges are intuitively defined as the set of edges that are incident, at the same time, to any vertex $v_1 \in V_{1_k}$ and any vertex $v_2 \in V_{2_k}$,

$$E_k = \{(v_1, v_2) \in E \mid v_1 \in V_{1_k}, v_2 \in V_{2_k}\} \tag{2.25}$$

Some properties can be derived from the DM-decomposition. In the sub-graph G_0, a complete matching of V_{2_0} into V_{1_0} always exists, and each edge in E_0 is admissible in G_0. Similarly, in the sub-graph G_∞, a complete matching of V_{1_∞} into V_{2_∞} exists and each edge in E_∞ is admissible in G_∞. In each sub-graph G_k ($k = 1, \cdots, b$), called a *consistent component* (Murota, 2000), a perfect matching exists, where every edge in E_k is also admissible in G_k. Therefore, for each DM-component, the following expressions hold

$$\begin{aligned} |V_{1_0}| &> |V_{2_0}| && \text{(2.26a)} \\ |V_{1_k}| &= |V_{2_k}| \quad (k = 1, \cdots, b) && \text{(2.26b)} \\ |V_{1_\infty}| &< |V_{2_\infty}| && \text{(2.26c)} \end{aligned}$$

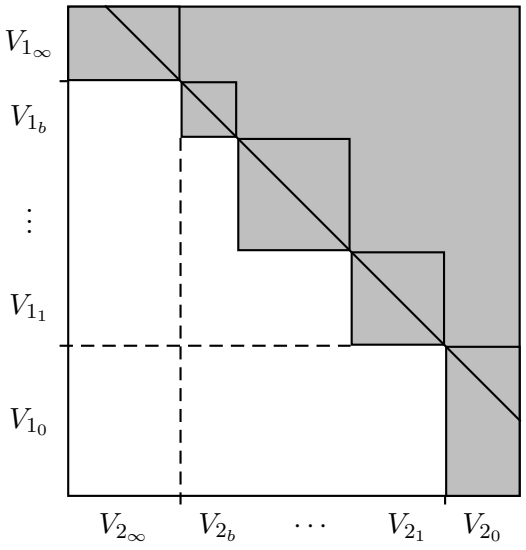


Figure 2.3: DM-decomposition in a biadjacency matrix.

The DM-decomposition can be graphically represented by a triangular form in the biadjacency matrix (see Figure 2.3). The white areas mean that only zeros can appear (there are no edges) while the coloured areas mean that both, zeros and ones, can appear (there may be some edges). The diagonal line represents the maximum matching in the graph, formed from the complete matching of V_{1_∞} into V_{2_∞} within G_∞, the perfect matchings in each G_k ($k = 1, \cdots, b$) and the complete matching of V_{2_0} into V_{1_0} within G_0.

Clearly, the DM-decomposition can be applied on a structural model since it is based on a bipartite graph. In the structural model-based diagnosis literature (Blanke et al., 2006), the DM-decomposition is used to decompose a structural model M into three parts,denoted as

$$G^+ = (M^+, X^+; A^+) \tag{2.27a}$$
$$G^0 = (M^0, X^0; A^0) \tag{2.27b}$$
$$G^- = (M^-, X^-; A^-) \tag{2.27c}$$

for $M^+, M^0, M^- \subseteq M$ and $X^+, X^0, X^- \subseteq X$. These three parts, G^+, G^0 and G^-, are respectively called *over-determined*, *just-determined* and

under-determined parts of the model, and can be computed from the DM-components as

$$G^+ = G_0 \tag{2.28a}$$
$$G^0 = \cup_{k=1}^{b} G_k \tag{2.28b}$$
$$G^- = G_\infty \tag{2.28c}$$

It is worth remarking that the set of known variables is not required to determine these three parts. In fact, by only regarding the set of equations, the sub-graphs in (2.27) are well defined since

$$X^+ = \mathrm{var}_X(M^+) \tag{2.29a}$$
$$X^0 = \mathrm{var}_X(M^0) \setminus X^+ \tag{2.29b}$$
$$X^- = X \setminus \{X^+ \cup X^0\} \tag{2.29c}$$

Therefore, for the sake of notational convenience, the over-determined, the just-determined and the under-determined parts of a model, M, will be denoted by M^+, M^0 and M^-, respectively.

2.4 Testing Consistency in Structural Models

In Section 2.1, it was seen that it is possible to detect and isolate faulty behaviour modes by testing consistency on different sub-models of the system. Here, it will be explained how to choose such sub-models.

First, the principal properties for sub-models devoted to test consistency are defined and characterised Next, the algorithm presented in (Krysander et al., 2008) to generate this class of sub-models is described. Finally, related works on this issue are reviewed and discussed.

2.4.1 Minimal Sub-models for Testing Consistency

The existence of analytical redundancy in a sub-model is a key property to test consistency, as seen in Section 2.1. Here, it will be referred to *structural redundancy* which can differ from the analytical redundancy if the structural model is not independent and/or compatible.

Given a subset of model equations $M \subseteq \mathbf{M}$ and its corresponding bipartite graph $G = (M, \mathbf{X}; A')$, the structural redundancy can be quantified (Krysander, 2006) as

$$\varphi_s(M) = |M| - \nu(G) \tag{2.30}$$

Note that only the set of unknown variables is again considered. Any equation subset such that $\varphi_s(M) \geq 1$ is said to contain structural redundancy and can be therefore used to test consistency, since the model M fulfils (2.3).

According to (2.30), it is straightforward to see that $\varphi_s(M^- \cup M^0) = 0$ for any equation subset $M \subseteq \mathbf{M}$. In fact, it can be verified that

$$\forall \mathbf{y} \in \mathbb{Y}, \exists \mathbf{x} \in \mathbb{X} : M'(\mathbf{x}, \mathbf{y}) \tag{2.31}$$

for any $M' \subseteq (\mathbf{M}^- \cup \mathbf{M}^0)$. Thus, the only equations containing structural redundancy are those in the over-determined part $\mathbf{M}^+$,

$$\varphi_s(\mathbf{M}^+) = \varphi_s(\mathbf{M}) \tag{2.32}$$

This implies that equations in the *structurally over-determined* set $\mathbf{M}^+$ are the only useful equations to test consistency.

Testing the consistency of the whole set $\mathbf{M}^+$ is not useful for distinguishing, among different behaviour modes, which are consistent and which are not (only *detection* task can be performed). Therefore, the strategy to follow is to derive different sub-models containing analytical redundancy so that each sub-model can test the consistency of different behaviour modes (the *isolation* task is accomplished).

A sufficient requirement to obtain the best isolability capabilities, from a structural point of view, is to find all the *minimal structurally overdetermined* (MSO) sub-models. An MSO set can be seen as a set of M equations that contains structural redundancy and no proper subset $M' \subset M$ has this property. This implies that $\varphi_s(M) = 1$ which motivates the next definition (Krysander et al., 2008).

Definition 2.2 (Minimal Structurally Over-determined)**.** A *Minimal Structurally Over-determined (MSO)* set of equations $M \subseteq \mathbf{M}$ is an over-determined set $M = M^+$ such that $\varphi_s(M) = 1$.

Typically, the consistency of an MSO set $M_k \subseteq \mathbf{M}$ is tested by deriving a *residual generator* (Chen and Patton, 1999). A residual generator is a function $r_k(\mathbf{y})$ from the observations $\mathbf{y}$ to a non-negative scalar value. This value is compared against a threshold J_k according to the following binary decision:

$$\begin{aligned} r_k(\mathbf{y}) \geq J_k &\quad \rightarrow \quad M_k \text{ is not consistent with } \mathbf{y} \\ r_k(\mathbf{y}) < J_k &\quad \rightarrow \quad M_k \text{ is consistent with } \mathbf{y} \end{aligned}$$

Several techniques exist to construct residuals generators, e.g. parameter estimation (Isermann, 2006), state observers (Patton et al., 1989), or parity equations (Gertler, 1998).

In (Krysander et al., 2008) an efficient algorithm to compute all the MSO sets given a structural model was developed. This algorithm is based on a top-down search, beginning with the complete structurally over-determined part $\mathbf{M}^+$ of the system model, and recursively constructing the search tree by removing equations until all the MSO sets are found. A version of such algorithm is depicted in Algorithm 2.1 where the first inputs are set to $M = \mathbf{M}^+$ and $R = \emptyset$. Variable R records the set of equations already removed in order not to compute the same MSO set more than once. During the execution of the algorithm, the MSO sub-model $\omega_i \subseteq \mathbf{M}$ is stored in $\mathbf{\Omega}$ (i.e. $\mathbf{\Omega} = \{\omega_1, \omega_2, \omega_3, \cdots\}$).

Algorithm 2.1 $\mathbf{\Omega} = \text{findMSO}(M, R)$

```
Ω := ∅
if φ_s(M) = 1 then
  Ω := {M}
else
  while R ⊉ M do
    Select an e ∈ M \ R
    E := M \ (M \ {e})^+
    if E ∩ R = ∅ then
      R := R ∪ E
      Ω := Ω ∪ findMSO(M \ E, R)
    else
      R := R ∪ E
    end if
  end while
end if
return Ω
```

To improve efficiency, a model reduction is cleverly done in Algorithm 2.1. Instead of removing equations one by one, the equations are removed by groups such that the structural redundancy is decreased by a rate of one at every recursive call. This group of equations E forms an equivalent class and can be easily computed (Krysander et al., 2008) as

$$E = M \setminus (M \setminus \{e\})^+ \tag{2.33}$$

by taking any equation $e \in M$ as long as M is a *proper structurally overdetermined* (PSO) sub-model (i.e. $M = M^+$). The resulting model reduction

	ω_c	τ	η
e_1	×		
e_2	×	×	
e_3	×		
e_4			×
e_5	×	×	×

Table 2.2: Structural model, $(\mathbf{M}, \mathbf{X}, A')$, of the compressor system.

is therefore $M \setminus E$ and it holds that

$$\varphi_s(M) = \varphi_s(M \setminus E) + 1 \tag{2.34}$$

It is worth noting that the set $M \setminus E$ is still a PSO sub-model, i.e. $M \setminus E = (M \setminus E)^+$.

Example 2.5. Consider the structural model $\mathbf{M} = \{e_1, \ldots, e_5\}$ of the compressor in Example 2.2 with the set $\mathbf{X}$ of unknown variables defined in Example 2.3 (the biadjacency matrix in Table 2.2). Next, how Algorithm 2.1 computes all the MSO sets is shown.

First, note that $\mathbf{M}^+ = \{e_1, e_2, e_3, e_4, e_5\}$, so the inputs to the algorithm are $M = \{e_1, e_2, e_3, e_4, e_5\}$ and $R = \emptyset$. The algorithm steps are:

1. $\varphi_s(M) = 2$. M is not an MSO set.
2. Select e_1. Then $E = \{e_1\}$ and $E \cap R = \emptyset$.
3. $R := \{e_1\}$. Call findMSO(M, R) with $M = \{e_2, e_3, e_4, e_5\}$ and $R = \{e_1\}$.
    1. $\varphi_s(M) = 1$. M is an MSO set, return $\mathbf{\Omega} = \{M\}$.
4. $\mathbf{\Omega} = \{\{e_2, e_3, e_4, e_5\}\}$.
5. Select e_2. Then $E = \{e_2, e_4, e_5\}$ and $E \cap R = \emptyset$.
6. $R := \{e_1, e_2, e_4, e_5\}$. Call findMSO$(M, R)$ with $M = \{e_1, e_3\}$ and $R = \{e_1, e_2, e_4, e_5\}$.
    1. $\varphi_s(M) = 1$. M is an MSO set, return $\mathbf{\Omega} = \{M\}$.

7. $\mathbf{\Omega} = \{\{e_2, e_3, e_4, e_5\}, \{e_1, e_3\}\}$.

8. Select e_3. Then $E = \{e_3\}$ and $E \cap R = \emptyset$.

9. $R := \{e_1, e_2, e_3, e_4, e_5\}$. Call findMSO$(M, R)$ with $M = \{e_1, e_2, e_4, e_5\}$ and $R = \{e_1, e_2, e_3, e_4, e_5\}$.

    1. $\varphi_s(M) = 1$. M is an MSO set, return $\mathbf{\Omega} = \{M\}$.

10. $\mathbf{\Omega} = \{\{e_2, e_3, e_4, e_5\}, \{e_1, e_3\}, \{e_1, e_2, e_4, e_5\}\}$. Stop the *while loop* since now $R = M$. Return $\mathbf{\Omega}$.

The algorithm finds all the three possible MSO sets of equations, $\mathbf{\Omega} = \{\omega_1, \omega_2, \omega_3\}$, within the compressor model:

$$\begin{aligned} \omega_1 &= \{e_2, e_3, e_4, e_5\} \\ \omega_2 &= \{e_1, e_3\} \\ \omega_3 &= \{e_1, e_2, e_4, e_5\} \end{aligned}$$

□

2.4.2 Review of Algorithms to Generate MSO Sets

As pointed out in Section 2.1, there are in the literature several approaches devoted to find what it could be generically called minimal redundant sub-models. In some works, this issue is refereed as generating *Analytical Redundancy Relations* (ARR) (Blanke et al., 2006). An ARR is a relation deduced from a system sub-model which only contains observed variables.Therefore, it can be directly used as a residual generator. To obtain an ARR from a structural sub-model M, each unknown variable must be matched with one equation (i.e. there exists a complete matching of the unknown variables into the set of equations) and an extra unmatched equation must exist in the model,

$$|M| = |var_{\mathbf{X}}(M)| + 1 = |\mathcal{M}_M^{max}| + 1 \tag{2.35}$$

Usually, this is interpreted as each equation solving its matched unknown variable and the extra equation (named redundant equation) testing the consistency. This requirement fits with the MSO set definition. The main difference between an ARR and an MSO set is that the computation of the unknown variables must be explicitly ensured in an ARR whereas in an MSO set is not (Armengol et al., 2009). Thus, it can be concluded that an ARR is a particular case of an MSO set.

One of the early algorithms to generate ARRs was proposed in (Blanke et al., 2006). This algorithm starts with a maximum matching in the set of unknown variables and subsequently a family of ARR is found by adding extra non-matched equations to the set of equations involved in the matching. However the result is not complete, i.e. not all the possible ARRs are guaranteed. Later, in (Gelso et al., 2008) the algorithm was improved such that all possible ARRs are found. This was accomplished by iteratively combining the set of ARRs computed in the former algorithm. Furthermore, the algorithm in (Blanke et al., 2006) was extended in (Düştegör et al., 2004), where the best possible matching according to computability criteria is used to generate the resulting family of ARRs.

In (Travé-Massuyès et al., 2006) another algorithm to generate all the ARRs was proposed. It first assumes that all the unknown variables are measured and thereby every equation becomes an ARR. Then, it combines these ARRs by eliminating variables and taking into account whether the eliminated variables can be computed. In (Ploix et al., 2005), another method based on variable elimination was presented.

A modification of the (Krysander et al., 2008) method was presented in (Rosich et al., 2009b). The objective of this modification is to guarantee that all the unknown variables in an MSO set can be easily computed when both, linear and non-linear equations, are involved in the MSO set. This algorithm will be presented and further detailed in Section 6.7.

The artificial intelligence community uses the concept of *minimal conflict* to test the consistency of a model. In (Cordier et al., 2004) the existence of an equivalence between ARRs and conflicts was shown. A method to compute the set of pre-compiled minimal conflicts was presented in (Pulido and Gonzalez, 2004). Its strategy is based on choosing one equation and then finding all the required equations to compute the unknown variables within the chosen equation. Another similar method based on the same strategy was also developed in (Krysander and Nyberg, 2002).

A detailed comparison of some of these methods presented above was done in (Armengol et al., 2009). A general conclusion drawn from all these methods, above mentioned, is that searching for all the MSO sets (or the ARRs) has exponential time complexity. Furthermore, all the algorithms that are complete (i.e. that compute all possible MSO sets) are equivalent when unknown variable computability is not considered (all of them return the same result). However, the computation time and the memory required to achieve the same result can significantly vary from one method to another.

Chapter 3

Introduction to Optimal Sensor Placement

This chapter introduces the foundations for a proper treatment of the optimal sensor placement problem for fault diagnosis. First, some assumptions are adopted in order to simplify the framework presented in Section 2.1. The purpose of these simplifications is to facilitate the study of the sensor placement problem by only focusing on the main issues of fault diagnosis systems. Then, fault detectability and isolability are defined within the consistency based diagnosis framework when structural models are used.

Once the fault diagnosis foundations used throughout this thesis are well established, the concept of sensor is introduced within the framework of fault diagnosis based on structural models. It will be shown how sensors are handled in the structural model as well as some basic properties relating the diagnosis performance with the sensors installed in the system. Furthermore, the optimal sensor placement problem for model based fault diagnosis is presented by means of a generalised formulation. Finally, a review of some related works on sensor placement for fault diagnosis is given.

3.1 Basic Assumptions for Sensor Placement

As mentioned above, some assumptions are here introduced in order to simplify the consistency based diagnosis framework. These assumptions can be briefly summarised as:

- There is no explicit faulty behaviour model.
- Every fault is related to one model equation.

- Multiple faults are not considered.

First of all, it is assumed that no explicit description of a faulty behaviour is available. This implies that only equations describing the non-faulty behaviour are allowed in the system model. Therefore, given the system model equations, $\mathbf{M}$, and the non-faulty behaviour mode, $\mathcal{NF}$, the following expression holds

$$M_{\mathcal{NF}} = \mathbf{M} \tag{3.1}$$

This assumption has been proposed in several works, usually from the AI community, e.g. (De Kleer and Williams, 1987) and (Reiter, 1987). These works claim that using information of the faulty behaviour modes can be sometimes warranted but is never guaranteed. Observe that obtaining fault models from a real system implies an expert knowledge of the system, or intentionally causing faults in the system for fault model identification. However, it is worth pointing out that using fault models may improve the diagnosis capabilities of the diagnosis system, (Frisk et al., 2003).

The second assumption concerns the model equations used for describing faulty behaviours modes. Given a fault f_i associated with the behaviour mode $\mathcal{F}_i \in \mathbf{B}$, the corresponding model equations describing such behaviour will henceforth assumed to be

$$M_{\mathcal{F}_i} = \mathbf{M} \setminus \{e_{f_i}\} \tag{3.2}$$

for $e_{f_i} \in \mathbf{M}$. This means that all model equations, except e_{f_i}, describe the behaviour of the system when fault f_i is present. In other words, any fault is limited to only affect one equation in the model. Note that this does not hold for the non-faulty behaviour mode $\mathcal{NF}$ due to (3.1), or for the unknown-fault behaviour mode, $\mathcal{UF}$, since $M_{\mathcal{UF}} = \emptyset$.

This second assumption directly implies that the behaviour modes assumed consistent with equation e_{f_i} are

$$\mathsf{assump}(e_{f_i}) = \{\mathcal{NF}, \mathcal{F}_1, \cdots, \mathcal{F}_{i-1}, \mathcal{F}_{i+1}, \cdots\} = \mathbf{B} \setminus \{\mathcal{F}_i, \mathcal{UF}\} \tag{3.3}$$

Therefore, when e_{f_i} becomes inconsistent implies that $\mathbf{B} \setminus \{\mathcal{F}_i, \mathcal{UF}\}$ are also inconsistent, i.e. either f_i or an unknown fault are a possible explanation of the current system behaviour.

To motivate this second assumption, consider an MSO set of equations $\omega \subseteq \mathbf{M}$. Then, according to (2.6) and (3.3), the behaviour modes assumed consistent with ω are

$$\mathsf{assump}(\omega) = \bigcap_{e \in \omega} \mathsf{assump}(e) = \mathcal{NF} \cup \{\mathcal{F}_i \mid e_{f_i} \notin \omega\} \tag{3.4}$$

Therefore, when ω becomes inconsistent, only the remaining behaviour modes can explain the actual behaviour of the system. Thus, the following behaviour modes are consistent with the observations:

$$\mathbf{B} \setminus \mathsf{assump}(\omega) = \{\mathcal{F}_i \mid e_{f_i} \in \omega\} \cup \mathcal{UF} \tag{3.5}$$

Now, it can be easily seen that, under this second assumption, diagnosis capabilities can be determined by just tracking equations e_{f_i} related with fault behaviour modes $\mathcal{F}_i$. Therefore, the diagnosis analysis can be simplified since it is only required to take into account the model equation while the behaviour modes remain in a second level. This assumption is also used in other works devoted to sensor placement for fault diagnosis, such as (Krysander and Frisk, 2008) and (Yassine et al., 2008).

Finally, it is also assumed that only single faults can occur in the system. This is a normal assumption introduced in many works due to the fact that the number of possible behaviours modes exponentially grows when multiple faults are considered. However, it is possible to extend the diagnosis from single faults to multiple faults (Nyberg, 2006). It can be argued, in support of this assumption, that single faults are usually more probable in the system than multiple faults.

The three assumption presented above will hold throughout this thesis and can be roughly stated as:

Assumption 3.1. *The system model only describes the non-faulty behaviour mode, i.e. expression in (3.1) always holds. No explicit fault models are considered.*

Assumption 3.2. *Every fault can only affect one model equation, i.e. expression in (3.2) always holds for all faulty behaviour modes $\mathcal{F}_i \in \mathbf{B}$, except for $\mathcal{NF}$ and $\mathcal{UF}$.*

Assumption 3.3. *Only single faults can occur in the system. Faulty behaviour modes corresponding to multiple faults are not considered.*

3.2 Fault Detectability and Isolability

In fault diagnosis, the two main tasks are first determining a fault occurrence in the system and second identifying which fault is present. These two tasks are respectively known as *fault detection* and *fault isolation*. In the following, basic definitions of fault detectability and fault isolability are given from the consistency based diagnosis perspective. These definitions are similar to the ones proposed in (Blanke et al., 2006) and (Krysander, 2006).

Definition 3.1 (Detectable fault)**.** A fault f_i represented by the behaviour mode $\mathcal{F}_i \in \mathbf{B}$ is *detectable* if there exists some observation $\mathbf{y} \in \mathbb{Y}$ such that $\mathcal{F}_i$ is consistent with $\mathbf{y}$ but the non-faulty behaviour mode, $\mathcal{NF}$, is not.

Determining the consistent observation set for each behaviour mode is not a trivial task when non-linear equations are involved in the model. Detectability analysis would be greatly simplified if structural model properties could be used instead of analytical properties. This, however, entails relating analytical and structural properties which is also not trivial. Hence, to mitigate such problem, the following assumption is also considered.

Assumption 3.4. *Given a behavioural structural model M_b, it must hold that*

$$\mathcal{O}(M_b) = \mathcal{O}(M_b^+) \tag{3.6}$$

Assumption 3.4 means that just the equations in the over-determined part of the model M_b constrain the set of consistent observations. Thus, M_b^+ is the only useful part of the model for diagnosis. It should be noted, however, that if (3.6) is not fulfilled, the model can still be used for diagnosis. The drawback is that the set $\mathcal{O}(M_b) \setminus \mathcal{O}(M_b^+)$ of consistent observations will not be taken into account.

Now detectability analysis can be rather simplified when structural models are considered under Assumptions 3.1, 3.2 3.3 and 3.4. Theorem 3.1 shows a necessary condition for a structural models to detect faults.

Theorem 3.1. *Given the structural model, $\mathbf{M}$, of the system, a necessary condition for detecting a fault f_i is*

$$e_{f_i} \in \mathbf{M}^+ \tag{3.7}$$

Proof. We will prove the theorem by showing that $e_{f_i} \notin \mathbf{M}^+$ implies that fault f_i is not detectable. First note that $e_{f_i} \notin \mathbf{M}^+$ is equivalent to

$$\mathbf{M}^+ \subseteq \mathbf{M} \setminus \{e_{f_i}\} \tag{3.8}$$

This implies that

$$\mathcal{O}(\mathbf{M} \setminus \{e_{f_i}\}) \subseteq \mathcal{O}(\mathbf{M}^+) \tag{3.9}$$

where $\mathbf{M} = M_{\mathcal{NF}}$ and $\mathbf{M} \setminus \{e_{f_i}\} = M_{\mathcal{F}_i}$ (Assumptions 3.1 and 3.2, respectively). Then, since (3.6) must hold, we obtain that

$$\mathcal{O}(M_{\mathcal{F}_i}) \subseteq \mathcal{O}(M_{\mathcal{NF}}) \tag{3.10}$$

which, according to Definition 3.1, means that the fault f_i is not detectable. □

The same result was obtained in Krysander (2006) for linear structured models where (3.6) also holds. In fact, under such assumption, the minimal checking model of M_b is M_b^+ which entails to not detect fault f_i as long as (3.8) holds (see Section 12.2 in Krysander (2006), specially Corollary 12.2).

Recall that due to model uncertainty only best case results can be obtained by using structural models. This explains the lack of a sufficient condition in Theorem 3.1. Nevertheless, it is clear that (3.7) can be efficiently used to determine detectability under the aforementioned assumptions, which motivates the next detectability definition on structural models.

Definition 3.2 (Structurally detectable fault)**.** A fault f_i represented by the behaviour mode $\mathcal{F}_i \in \mathbf{B}$ is *structurally detectable* if (3.7) holds.

There exist several works in the literature, (Travé-Massuyès et al., 2006) and (Ploix et al., 2005), that characterise structural fault detectability by means of the MSO sets. The next lemma shows that, according to (3.7), it is also possible to determine structurally detectable faults by regarding the MSO sets.

Lemma 3.1. *Given a set Ω of MSO sets obtained from $\mathbf{M}$, a fault f_i is structurally detectable if there exists at least one MSO set $\omega \in \Omega$ such that*

$$e_{f_i} \in \omega \tag{3.11}$$

Proof. From Algorithm 2.1, it is easy to verify that $\omega \subseteq \mathbf{M}^+$, which together with (3.11), implies (3.7). □

Determining the set of faults that can be isolated from a given fault is also an important issue in diagnosis analysis. In the following definition, fault isolability is introduced.

Definition 3.3 (Isolable fault)**.** A fault f_i represented by the behaviour mode $\mathcal{F}_i \in \mathbf{B}$ is *isolable* from another fault f_j represented by the behaviour mode $\mathcal{F}_j \in \mathbf{B}$ if there exist some observations $\mathbf{y} \in \mathbb{Y}$ such that $\mathcal{F}_i$ is consistent with $\mathbf{y}$ but $\mathcal{F}_j$ is not.

According to this definition, the non-faulty behaviour mode $\mathcal{NF}$ cannot be isolated from the other behaviour modes since it always becomes inconsistent for any consistency test. On the other hand, no faulty behaviour mode can be isolated from the unknown-fault behaviour mode, $\mathcal{UF}$, since it never becomes inconsistent. In fact, as previously mentioned, $\mathcal{UF}$ is used as a last resort when no other defined behaviour models can explain the actual

system behaviour. This highlights that, for diagnosis analysis, it suffices to only consider those modes concerning faults, i.e. $\mathcal{F}_i$, while the $\mathcal{NF}$ and $\mathcal{UF}$ behaviour modes can be neglected, without loss of generality.

Note also that fault detection becomes a particular case of fault isolation from the non-faulty behaviour mode, $\mathcal{NF}$. Therefore, fault isolability can also be characterised, by extending Theorem 3.1 to any couple of faulty behaviour modes, as a structural model property under the same assumptions as for fault detectability.

Theorem 3.2. *Given a structural model,* $\mathbf{M}$*, of the system, a necessary condition for isolating a fault* f_i *from a fault* f_j *is*

$$e_{f_i} \in (\mathbf{M} \setminus \{e_{f_j}\})^+ \tag{3.12}$$

Proof. The proof is similar to the one in Theorem 3.1. We will prove that $e_{f_i} \notin (\mathbf{M} \setminus \{e_{f_j}\})^+$ implies that f_i is not isolable from f_j. Such a condition is equivalent to

$$(\mathbf{M} \setminus \{e_{f_j}\})^+ \subseteq \mathbf{M} \setminus \{e_{f_i}\} \tag{3.13}$$

and therefore

$$\mathcal{O}(\mathbf{M} \setminus \{e_{f_i}\}) \subseteq \mathcal{O}((\mathbf{M} \setminus \{e_{f_j}\})^+) \tag{3.14}$$

Now, regarding that $\mathbf{M} \setminus \{e_{f_i}\} = M_{\mathcal{F}_i}$ and $\mathbf{M} \setminus \{e_{f_j}\} = M_{\mathcal{F}_j}$ (Assumption 3.2), together with Assumption 3.4, we obtain that

$$\mathcal{O}(M_{\mathcal{F}_i}) \subseteq \mathcal{O}(M_{\mathcal{F}_j}) \tag{3.15}$$

which, according to Definition 3.3, ends the proof. □

As for the fault detection case, structural isolability will be defined based on Theorem 3.2.

Definition 3.4 (Structurally isolable fault)**.** A fault f_i represented by the behaviour mode $\mathcal{F}_i \in \mathbf{B}$ is *structurally isolable* from another fault f_j represented by the behaviour mode $\mathcal{F}_j \in \mathbf{B}$ if (3.12) holds.

It is also possible to determine whether two faults are structurally isolable by regarding the MSO sets, as the next lemma shows.

Lemma 3.2. *Given a set* Ω *of MSO sets obtained from* $\mathbf{M}$*, a fault* f_i *is structurally isolable from a fault* f_j *if at least one MSO set* $\omega \in \Omega$ *exists such that*

$$e_{f_i} \in \omega \wedge e_{f_j} \notin \omega \tag{3.16}$$

Proof. Again, it can be verified that such MSO set fulfils $\omega \subseteq (\mathbf{M} \setminus \{e_{f_2}\})^+$. Therefore, (3.16) implies that condition (3.12) is also fulfilled. □

From Definitions 3.2 and 3.4, it is clear that the set of (structurally) detectable and isolable faults can be determined before implementing the diagnosis system and computing any MSO set. The same results could be obtained from Lemmas 3.1 and 3.2 as long as all possible MSO sets, $\mathbf{\Omega}$, computed by Algorithm 2.1 are provided (Krysander et al., 2008).

To characterise fault isolability, there are two possibilities. Let $\mathbf{F}$ be the set of (structurally) detectable faults. The first possibility is based on defining a family of sets $\mathbb{F}_I^1 = \{\dots, F_I(f_i), \dots\}$ where the set $F_I(f_i)$ is defined such that

$$F_I(f_i) = \{f_j \in \mathbf{F} \mid f_i \text{ is isolable from } f_j\} \tag{3.17}$$

for all $f_i \in \mathbf{F}$.

The second possibility to characterise (structurally)isolability is by means of a partition $(M_1, M_2, \cdots, M_n)$ of the overdetermined part M^+. In this partition it holds that

$$(M \setminus \{e\})^+ = M^+ \setminus M_k \qquad (k = 1, \cdots, n) \tag{3.18}$$

for any $e \in M_k$. In (Krysander et al., 2008), it is proved that each set of such a partition defined in (3.18) is an equivalence class. Indeed, this is the same equivalent class presented in (2.33) for improving the MSO set computation. Then, the set of isolable faults can be characterised according to the following theorem, presented in (Krysander and Frisk, 2008).

Theorem 3.3. *Given a model M, let f_i and f_j be two (structurally) detectable faults in M. The faulty behaviour mode f_i is (structurally) isolable from f_j if and only if e_i and e_j belong to different sets in the partition defined in (3.18).*

Proof. See *Theorem 3* in (Krysander and Frisk, 2008). □

This means that (structurally) fault isolability can be represented by means of a partition $\mathbb{F}_I^2 = (F_1, F_2, \cdots, F_m)$ on the detectable faults $\mathbf{F}$, where for any two faults $f_1, f_2 \in F_i$ with $F_i \in \mathbb{F}_I^2$ it holds that

$$e_1, e_2 \in M_k \tag{3.19}$$

with e_1 and e_2 being the corresponding fault equations for f_1 and f_2, and M_k a partition set, according to (3.18).

Observe that each set $F_i \in \mathbb{F}_I^2$ denotes the set of non (structurally) isolable faults among them. Moreover, a key property derived from Theorem 3.3 is that fault isolability is symmetric as long as is defined according to Definition 3.4, i.e. if f_i is (structurally)isolable from f_j then f_j is also (structurally)[1] isolable from f_i. This also means that $f_j \in F_I(f_i)$ implies that $f_i \in F_I(f_j)$ for $F_I(f_i), F_I(f_j) \in \mathbb{F}_I^1$. This makes the isolability characterisation $\mathbb{F}_I^2$ more compact than $\mathbb{F}_I^1$. However, both characterisations are fully equivalent.

3.3 Sensors for Fault Diagnosis

A sensor is a system component that allows us to obtain observations from the process. This means that by adding more sensors to the system, the set of observations can be easily increased. Therefore, better diagnosis performance can be achieved since the capability of diagnosing faults significantly depends on the observations, as it has been seen in the previous section. In this section, how sensors are handled within the structural model will be explained, and some important properties of adding sensors for diagnosis will also be introduced.

It is important to remark that sensors link internal process information with other external systems. Therefore, sensors can be installed in the system for many purposes, e.g. control, monitoring or fault diagnosis. Here, however, only sensors for diagnosis, specially for sensor placement analysis, will be regarded. This implies that all sensors installed for purposes other than diagnosis will be out of the scope of this thesis. Hence, in the following, the term *sensor* will be limited to a non-installed sensor that may be installed in the process in order to improve the diagnosis capabilities.

3.3.1 Adding Sensors in the Structural Model

Performing the sensor placement analysis implies that not already installed sensors are candidate sensors for installation. Since the analysis is based on model, the system model has to be adapted for every extra sensor chosen for installation. Therefore, a handy treatment of such sensors in the model is required.

Representing the sensors by means of the known variable set would be a straightforward approach. If a sensor measuring some unknown variable

[1]Henceforth, detectability and isolability are based on Definitions 3.2 and 3.4. Therefore, the term "structurally" will be intentionally neglected.

$x \in \mathbf{X}$ is chosen for installation in the system, then the corresponding variable, x, becomes known. This variable is removed from the set $\mathbf{X}$ of unknown variables and is added into the set $\mathbf{Y}$ of known variables. Nevertheless, this strategy is not desirable when handling faults in sensors since no sensor behavioural model is explicitly defined. Instead of this, an explicit sensor model will be used. Henceforth, any candidate sensor in the sensor placement analysis is modelled by means of one single equation which is called *sensor equation*, represented by

$$e_s : x = y \tag{3.20}$$

for $x \in \mathbf{X}$ and $y \in \mathbf{Y}$. Thereby, when a sensor is chosen for installation in the original system, its corresponding sensor equation is added to the system model (i.e. $\mathbf{M} \cup \{e_s\}$ is the new set of model equations).

Usually, sensors have a signal conditioning stage that may include, for example, signal filtering, linearisation or/and digitisation. Moreover, some measurements involve more than one unknown process variables, e.g. a wattmeter, that measures the electric power which depends on the electric current and the voltage. This means that the sensor model could be generally modelled as

$$e_s : y = h(x_1, \ldots, x_n) \tag{3.21}$$

where h represents the signal conditioning process and $x_1, \ldots, x_n$ the unknown process variables involved in the measurement.

To handle sensor models as in (3.21) the following strategy is adopted. The sensor model is split in two equations by adding an extra unknown variable x' such that

$$e'_s \quad : \quad x' = h(x_1, \ldots, x_n) \tag{3.22a}$$
$$e_s \quad : \quad y = x' \tag{3.22b}$$

Then, the former equation e'_s (3.22a) can be fitted in the structural system model, as part of the original model, whereas equation (3.22b) is now the corresponding sensor equation.

Remark that adding equation e'_s into the model preserves the structural redundancy of the model,

$$\varphi_s(\mathbf{M}) = \varphi_s(\mathbf{M} \cup \{e'_s\}) \tag{3.23}$$

Furthermore, when the sensor is installed (i.e. equation e_s is added to the model) both equations, e'_s and e_s, belongs to the same DM-decomposition

part. In fact, this can be checked by considering the surplus function in (2.20). It then holds that

$$|\mathrm{var}_{\mathbf{X}}(M \cup \{e'_s\})| - |M \cup \{e'_s\}| > |\mathrm{var}_{\mathbf{X}}(M \cup \{e'_s, e_s\})| - |M \cup \{e'_s, e_s\}| \quad (3.24)$$

for any set $M \subseteq \mathbf{M}$. Therefore, the set $M \cup \{e'_s, e_s\}$ is in the family of the minimizers defined in (2.21), while the set $M \cup \{e'_s\}$ is not (i.e. equations e'_s and e_s always come together in the DM-decomposition).

Summarising the above explanation, to handle a sensor for the sensor placement analysis it is only required to consider the sensor equation (3.20). Any other equations involved in the sensor model can be considered, without loss of generality, in the equation set of the structural system model, $\mathbf{M}$. Now, a one-to-one correspondence between the sensor equation and the measured variable exists, which motivates the following definition.

Definition 3.5 (Set of candidate sensors)**.** The set $\mathbf{S}$ of all candidate sensors is defined as a subset of unknown system variables $\mathbf{S} \subseteq \mathbf{X}$ such that each variable $s \in \mathbf{S}$ can be measured by installing a sensor in the system.

In the following, according to this definition, a sensor $s \in \mathbf{S}$ will denote the variable to be measured, as well as the physical measuring device to be installed in the system. Now, it is clear that sensors can be handled by pointing out the unknown variables $s \in \mathbf{S}$ and then adding the corresponding sensors equation e_s to the model. Given a set of sensors $S \subseteq \mathbf{S}$ the set of their corresponding sensor equations is denoted by M_S,

$$M_S = \bigcup_{s \in S} e_s \quad (3.25)$$

Adding extra sensors in the system may improve diagnosis capabilities. To motivate this idea, the following lemma shows that the overdetermined part of the model is never reduced by adding new equations.

Lemma 3.3. *Let M_1 and M_2 be two arbitrary sets of equations such that $M_1 \subseteq M_2$, then it holds that*

$$M_1^+ \subseteq M_2^+ \quad (3.26)$$

Proof. First recall that the overdetermined part is the minimal subset with minimum surplus function p_0. Hence, since $M_1 \subseteq M_2$, it holds that

$$p_0(M_2^+) \leq p_0(M_1^+ \cup M_2^+) \quad (3.27)$$

Furthermore, the surplus function p_0 introduced in (2.20) is a sub-modular function (Murota, 2000) since

$$p_0(A \cup B) + p_0(A \cap B) \leq p_0(A) + p_0(B) \tag{3.28}$$

for A and B being subsets of one vertex set in the bipartite graph.

By replacing A with M_1^+ and B with M_2^+ in (3.28), the following expression is obtained

$$p_0(M_1^+ \cup M_2^+) + p_0(M_1^+ \cap M_2^+) \leq p_0(M_1^+) + p_0(M_2^+) \tag{3.29}$$

This, together with (3.27), implies that

$$p_0(M_1^+) \geq p_0(M_1^+ \cap M_2^+) \tag{3.30}$$

Next, since M_1^+ is the minimal subset in M_1 with minimum surplus function p_0 (i.e. $p_0(M_1^+) < p_o(M')$ for any $M' \subset M_1^+$), expression (3.30) is only satisfied if $M_1^+ \subseteq M_2^+$. □

The same can be intuitively seen by regarding the structural redundancy, φ_s. In fact, adding sensor equations to the model increases the number of equations but not the number of unknown variables. Therefore the redundancy is never decreased by installing extra sensors,

$$\varphi_s(\mathbf{M}) \leq \varphi_s(\mathbf{M} \cup M_S) \tag{3.31}$$

for any candidate sensor $S \subseteq \mathbf{S}$. The next corollary formalises the idea of improving the diagnosis capabilities by installing extra sensors.

Corollary 3.1. *Let $S \subseteq \mathbf{S}$ be a candidate sensor set, then it holds that*

$$\mathbf{M}^+ \subseteq (\mathbf{M} \cup M_S)^+ \tag{3.32}$$

Proof. It directly holds from Lemma 3.3. □

On the one hand, Corollary 3.1 can be used to show that the more sensors are installed in the system the more equations may appear in the overdetermined part of the model, i.e. diagnosis performance may be improved. On the other hand, it proves that an fault equation, $e_{f_i} \in \mathbf{M}$, never becomes non-detectable by installing extra sensors. Thus the diagnosis performance is never degraded by installing sensors[2].

[2]This is only true as long as sensor faults are not considered (see next Section 3.3.2).

3.3.2 Faults in Sensors

As mentioned before, the system consists of a set of installed components that can present several faults represented by its corresponding behaviour modes. Sensors can be considered part of such set of components. Hence, installing extra sensors to improve diagnosis can lead to an increase in the number of possible faults to be considered in the diagnosis analysis. Furthermore, a sensor fault is a special case of behaviour mode since it is only present in the model when the sensor is installed. In the following, it will be shown how to handle sensor faults in the sensor placement analysis.

Let $S \subseteq \mathbf{S}$ be a candidate sensor set that is installed in the system, i.e. the current system model is $(\mathbf{M} \cup M_S)$. Assume that every sensor $s \in S$ has a faulty behaviour mode, $\mathcal{F}_s$, associated to it. Then, the faulty behaviour model corresponding to sensor $s \in S$ is

$$M_{F_s} = \mathbf{M} \cup (M_S \setminus e_s) \tag{3.33}$$

which coincides with equation (3.2), with the sensor equation being used instead. This can be understood as a fault in sensor $s \in S$ affecting equation e_s, which seems reasonable. Furthermore, by tracking sensor equations in the model, it is possible to known which sensor faults need to be considered (i.e. if $e_s \in (\mathbf{M} \cup M_S)$ then faulty behaviour mode $\mathcal{F}_s$ needs to be taken into account in the analysis, otherwise not).

In order to distinguish between the original faults defined before the sensor placement analysis and the faults in the extra sensors after the analysis, the former are called *process faults* (denoted as f^p) whereas the latter are called *sensor faults* (denoted as f^s). Remark that a process fault may refer to a sensor as long as it is installed in the original system and not considered in the sensor placement analysis as a candidate sensor (e.g. a sensor for control purposes).

3.3.3 Physical Redundant Sensors

Depending on the process, it may be possible to install in the system several sensors measuring the same process variable. For instance, consider a pipe with two sensors measuring the same inside flow. The sensor that measures an already measured variable is named *redundant sensor*. Given a set of candidate sensors, $\mathbf{S}$, the set of candidate redundant sensor will be denoted by $\mathbf{S}'$, where each redundant sensor in $\mathbf{S}'$ measures the same unknown variable as a sensor in $\mathbf{S}$.

A redundant sensor is modelled by means of its corresponding sensor equation. Let $s' \in \mathbf{S}'$ be the redundant sensor of $s \in \mathbf{S}$. Then, the redundant sensor equation has the form

$$e_{s'} : x = y' \tag{3.34}$$

where both equations, e_s and $e_{s'}$, share the same unknown variable $x \in \mathbf{X}$, but different known variables $y, y' \in \mathbf{Y}$. It is important to note that sensor s must be installed in the system before s' is chosen for installation. Otherwise, s' would not behave as a true redundant sensor. This requirement should be ensured in the following.

Thus, the model of a system with all candidate sensors (non-redundant as well as redundant) installed, is represented by

$$\mathbf{M} \cup M_{\mathbf{S}} \cup M_{\mathbf{S}'} \tag{3.35}$$

Intuitively, considering redundant sensors should improve the diagnosis performance, i.e. more observations are obtained. Next lemma shows how diagnosis is improved by adding redundant sensors in the system.

Lemma 3.4. *Let* $(\mathbf{M} \cup \{e_s\} \cup \{e_{s'}\})$ *be a system model with a sensors* $s \in \mathbf{S}$ *and its corresponding redundant sensor* $s' \in \mathbf{S}'$*. Then, it holds that*

$$(\mathbf{M} \cup \{e_s\} \cup \{e_{s'}\})^+ = (\mathbf{M} \cup \{e_s\})^+ \cup \{e_s\} \cup \{e_{s'}\} \tag{3.36}$$

Proof. Fist, since the sub-model $\{e_s, e_{s'}\}$ consists of two equations and one sharing variable, it follows from (2.20) that

$$p_0((\mathbf{M} \cup \{e_s\})^+ \cup \{e_s\} \cup \{e_{s'}\}) = p_0((\mathbf{M} \cup \{e_s\})^+) - 1 \tag{3.37}$$

Then, note that according to (2.21) and (2.22), $(\mathbf{M} \cup \{e_s\})^+$ is the smaller subset with the minimum surplus function in $(\mathbf{M} \cup \{e_s\})$. From this and (3.37), it follows that $(\mathbf{M} \cup \{e_s\})^+ \cup \{e_s\} \cup \{e_{s'}\})$ is the smallest subset with the minimum surplus function in $(\mathbf{M} \cup \{e_s\} \cup \{e_{s'}\})$, which completes the proof.

□

According to this lemma, installing redundant sensors does not improve diagnosis on process faults since no new system model equation appears in the overdetermined part. The only possible new equations appearing in the overdetermined part are the redundant sensor equations. Thus, redundant sensors can only improve diagnosis on sensor faults, which means that regarding redundant sensors for process fault diagnosis makes no sense.

3.3.4 Useful Sensors for Fault Diagnosis

Given a set of candidate sensors, it may happen that some of them do not improve the diagnosis performance. For instance, consider a sensor that measures a variable not related with the system. Then, it is clear that such sensor can not give any useful observation for diagnosis. This section is devoted to determine which sensors can contribute to the diagnosis, so that those sensors that are not useful can be rejected. By doing this, the complexity of the sensor placement analysis can be reduced since less sensors will be considered.

From the DM-decomposition introduced in Section 2.3, it can be seen that removing equations from the just-determined or under-determined parts of the model has no effect on the overdetermined part. Thus, given any set M of model equations, it holds that

$$M^+ = (M \setminus M')^+ \tag{3.38}$$

for any $M' \subseteq (M^0 \cup M^-)$. This motivates the fact that all sensor equations appearing in the just-determined or under-determined parts can be removed since no diagnosis capability will be lost. Therefore, the set $S_0 \subseteq \mathbf{S}$ of useful sensors can be defined as

$$S_0 = \{s \in \mathbf{S} \mid e_s \in (\mathbf{M} \cup M_{\mathbf{S}})^+\} \tag{3.39}$$

According to (3.38), it holds that the overdetermined part of the system model with all the candidate sensors, $\mathbf{S}$, remains unaltered when only useful sensors S_0 are used,

$$(\mathbf{M} \cup M_{\mathbf{S}})^+ = (\mathbf{M} \cup M_{S_0})^+ \tag{3.40}$$

meaning that sensors in $\mathbf{S} \setminus S_0$ do not improve diagnosis capabilities.

To cope with useful redundant sensors, it suffices to look for redundant sensors in the useful sensors set S_0. The set $S_0' \subseteq \mathbf{S}'$ of useful redundant sensors is therefore defined as

$$S_0' = \{s' \in \mathbf{S}' \mid s' \text{ is the redundant sensor of } s \in S_0\} \tag{3.41}$$

Therefore, given a set of sensors, $\mathbf{S}$, and a set of redundant sensors, $\mathbf{S}'$ to be installed in the system, the only sensors to be considered for fault diagnosis are the sensors in S_0 according to (3.39) and the redundant sensors in S_0' according to (3.41). Remark that the diagnosis performance on process faults is not degraded by choosing S_0 and S_0' instead of $\mathbf{S}$ and $\mathbf{S}'$,

$$\mathbf{M} \cap (\mathbf{M} \cup M_{\mathbf{S}} \cup M_{\mathbf{S}'})^+ = \mathbf{M} \cap (\mathbf{M} \cup M_{S_0} \cup M_{S_0'})^+ \tag{3.42}$$

In fact, as proved in Lemma 3.4, useful redundant sensors in S_0' neither contribute to improve diagnosis performance on process faults since

$$\mathbf{M} \cap (\mathbf{M} \cup M_{\mathbf{S}} \cup M_{\mathbf{S}'})^+ = \mathbf{M} \cap (\mathbf{M} \cup M_{S_0})^+ \tag{3.43}$$

However, Theorem 3.4 shows how fault detectability and isolability for sensor faults can be guaranteed by adding redundant sensors.

Theorem 3.4. *Consider a system* $\mathbf{M}$ *with a set of useful sensors,* S_0*, installed, as well as a redundant sensor* s' *of* $s \in S_0$*. Then, the sensor faults corresponding to* s *and* s' *are detectable and fully isolable from any other process or sensor fault.*

Proof. It suffice to prove that the faults in sensors s and s' are detectable and fully isolable in

$$\mathbf{M} \cup M_{S_0} \cup \{e_{s'}\} \tag{3.44}$$

According to Lemma 3.4, both sensor faults are detectable,

$$e_s, e_{s'} \in (\mathbf{M} \cup M_{S_0} \cup \{e_s\} \cup \{e_{s'}\})^+ \tag{3.45}$$

Then, according to (3.39), it is straightforward to see that a fault in s is isolable from a fault in s' since

$$e_s \in (\mathbf{M} \cup M_{S_0})^+ \tag{3.46}$$

Now, a fault in s is also isolable from any other fault as long as

$$e_s \in ((\mathbf{M} \cup M_{S_0} \cup \{e_{s'}\}) \setminus \{e\})^+ \tag{3.47}$$

for any $e \in (\mathbf{M} \cup M_{S_0}) \setminus \{e_s\}$. First, note that $(\{e_s, e_{s'}\})^+ = \{e_s, e_{s'}\}$. Then according to Lemma 3.3, it holds that

$$(\{e_s, e_{s'}\})^+ \subseteq ((\mathbf{M} \cup M_{S_0} \cup \{e_{s'}\}) \setminus \{e\})^+ \tag{3.48}$$

which fulfils (3.47).

Equations (3.46) and (3.47) prove that the fault in s are fully isolable. Since both equations, e_s and $e_{s'}$, are structurally equivalent, it suffices to swap e_s for $e_{s'}$ to prove that the fault in s' is also fully isolable. □

According to the above theorem, fault detectability and full isolability for a sensor $s \in S_0$ and its corresponding redundant sensor $s' \in S_0'$ come for free when S_0 and s' are installed in the system. Moreover, this also shows that there is no need to consider more than two sensors measuring

the same variable, even though it is physically possible, since fault detection and isolation on sensors s and s' is already ensured with two sensors.

In the following, for notational convenience, it will be assumed that only useful sensors have been previously selected as candidate sensors before the sensor placement analysis,

$$\mathbf{S} \triangleq S_0 \tag{3.49}$$

The same assumption will hold for redundant sensors, $\mathbf{S}' \triangleq S'_0$.

3.4 Maximum Diagnosis Specifications

Installing all candidate sensors in the system does not guarantee that the desired diagnosis specifications will be achieved. For this reason, it is important to know beforehand all the faults that could be detectable and all the faults that could be isolable by placing extra sensors in the system.

Let $\mathbf{F_P}$ be the set of process faults to be detected and isolated. According to Assumption 3.2, there is one model equation $e_{f^p} \in \mathbf{M}$ related to every fault $f^p \in \mathbf{F_P}$. Therefore it is possible to characterise the faults by their corresponding fault equations. Furthermore, according to Corollary 3.1, maximum detectability specifications of process faults can be achieved when the whole set of candidate sensors, $\mathbf{S}$, is installed in the system. Therefore, the set of all detectable process faults is defined as

$$F_{D_P} = \{f^p \in \mathbf{F_P} \mid e_{f^p} \in (\mathbf{M} \cup M_{\mathbf{S}})^+)\} \tag{3.50}$$

As mentioned in Section 3.3.2, sensor faults are directly related to their corresponding sensor equations. Furthermore, recall from (3.49) that all sensor faults can be detected since all sensors are useful for diagnosis. Then, if $\mathbf{F_S}$ is the set of sensor faults, it directly holds that the set of detectable sensor faults is

$$F_{D_S} = \mathbf{F_S} \tag{3.51}$$

The same holds for redundant sensor faults. Let $\mathbf{F_{S'}}$ be the set of redundant sensor faults, then the set of detectable redundant sensor faults is

$$F_{D_{S'}} = \mathbf{F_{S'}} \tag{3.52}$$

Thus, the maximum set of detectable faults, involving both process and sensor faults, is

$$F_{D_{max}} = F_{D_P} \cup F_{D_S} \cup F_{D_{S'}} \tag{3.53}$$

Maximum fault isolability is achieved when all candidate sensors are installed in the system. Thus, according to (3.12), a fault f_i is isolable from a fault f_j if

$$e_{f_i} \in (\mathbf{M} \cup M_{\mathbf{S}} \cup M_{\mathbf{S}'}) \setminus \{e_{f_j}\})^+ \tag{3.54}$$

where e_{f_i} and e_{f_j} are either fault equations or sensor equations, depending on f_i and f_j being a process or a sensor fault. In the following, either $\mathbb{F}^1_{I_{max}}$ or $\mathbb{F}^2_{I_{max}}$ (see Section 3.2) will be used to characterise maximum fault isolability according to (3.54).

Typically, maximum diagnosis specifications will be assumed (i.e. detectability of faults in $F_{D_{max}}$ and isolability of faults according to $\mathbb{F}^1_{I_{max}}$). Recall that no better diagnosis specifications can be achieved. However, it is possible to define other specifications than the maximum ones. Any diagnosis specifications, F_D and $\mathbb{F}^1_I$, will be feasible as long as

$$F_D \subseteq F_{D_{max}} \tag{3.55}$$

and

$$F'_I(f) \subseteq F_I(f) \tag{3.56}$$

for all $f \in F_D$, where $F'_I(f) \in \mathbb{F}^1_I$ and $F_I(f) \in \mathbb{F}^1_{I_{max}}$.

3.5 Optimal Sensor Placement for Fault Diagnosis

Usually, there are several different sensor configurations that solve the sensor placement problem when large-scale systems with many candidate sensors are considered. Nevertheless, only one of these possible solutions has to be finally chosen to design a diagnosis system. Here, in order to cope with this selection decision, the best candidate sensor configuration will be sought. Thus, the optimal sensor placement problem is addressed. This implies to consider a cost function to evaluate the best (the optimal) solution among all possible configurations.

3.5.1 Sensor Cost Function

The cost of installing a sensor in the system can be motivated by many reasons and may depend on the system application. For instance, the purchase price or the mounting facilities can be good reasons for devices which are mass produced. On the other hand, for critical systems (e.g. aerospace devices) the objective may focus on reliability and error robustness. Also, for applications where the mounting space or the weight is limited, the number of sensors could be optimised.

For a general use of the sensor cost and to not going into details in the different purposes for installing sensors, the cost of installing a sensor will be evaluated by a positive real number. This number generically evaluates the properties to place the sensor (e.g. price, reliability, weight, etc.) with respect to the other sensors.

Definition 3.6 (Sensor Cost)**.** Let $s \in (\mathbf{S} \cup \mathbf{S}')$ be a sensor. Then, the cost associated to the installation of this sensor is $C(s) > 0$.

Thus, the cost of installing a set of candidate sensors is the sum of the cost of installing each sensor in the candidate set. This is formally introduced in the following definition.

Definition 3.7 (Sensor Set Cost)**.** Let $S \subseteq (\mathbf{S} \cup \mathbf{S}')$ be a subset of sensors. Then, the cost of installing all the sensors in S is defined by

$$C(S) \triangleq \sum_{s \in S} C(s) \tag{3.57}$$

A key property of the cost function defined in (3.57) is that it is a strictly monotonic function since for $S_i, S_j \subseteq (\mathbf{S} \cup \mathbf{S}')$, if $S_i \subset S_j$ then $C(S_i) < C(S_j)$. Observe that, according to Definition 3.6, two redundant sensors can have different sensor costs.

3.5.2 Optimal Sensor Placement Problem Set-up

The optimal sensor placement problem for fault diagnosis is formulated in this section from a general perspective. Here, it is assumed that an algorithm (or a function), P, must be designed in order to test whether the diagnosis requirements are fulfilled for a given set of sensors. This algorithm is fed with the process information, typically represented by the model $\mathcal{M}$ of the process, the set of candidate sensors S to be installed in the system and the required diagnosis specifications, $\mathcal{F}$. The algorithm returns a boolean value indicating whether the diagnosis requirements are achieved,

$$P(\mathcal{M}, S, \mathcal{F}) = \begin{cases} 1 \text{ if diagnosis specifications } \mathcal{F} \text{ are fulfilled} \\ 0 \text{ otherwise} \end{cases} \tag{3.58}$$

This algorithm is developed mainly depending on the system description, $\mathcal{M}$, and the diagnosis requirements, $\mathcal{F}$. For instance, the system could be described by means of linear equations or by a logic statement. In the former case the algorithm should be based on linear algebra tools and in

the later case, relational algebra should be used. Concerning the diagnosis requirements $\mathcal{F}$, the algorithm will be different if, for example, the diagnosis specifications are based on model properties or on residual generator properties. In the former case, the test is straight since the model is provided, while in the later case, it is required to generate all the MSO sets and design the residual generators. However, sometimes it is possible to reformulate the same diagnosis property based on MSO sets as a model property. For instance, see fault detectability and isolability introduced in Section 3.2 (compare Theorem 3.1 with Lemma 3.1 and Theorem 3.2 with Lemma 3.2).

Now, let $2^{\mathbf{S}}$ be the power set of the candidate sensor set $\mathbf{S}$, i.e. $2^{\mathbf{S}}$ is the family of sensor sets that contains all the possible sensor configurations that can be installed in the system. Then, a two-class partition can be introduced in $2^{\mathbf{S}}$, according to whether the diagnosis specifications are verified,

$$[2^{\mathbf{S}}]_+ = \{S \in 2^{\mathbf{S}} \mid P(\mathcal{M}, S, \mathcal{F}) = 1\}$$
$$[2^{\mathbf{S}}]_- = \{S \in 2^{\mathbf{S}} \mid P(\mathcal{M}, S, \mathcal{F}) = 0\}$$

The class $[2^{\mathbf{S}}]_+$ is the set of all sensor configurations that are feasible solutions for the sensor placement problem for diagnosis. Since each $S \in [2^{\mathbf{S}}]_+$ has an associated cost according to Definition 3.7, then there exists a $S^* \in [2^{\mathbf{S}}]_+$ such that it corresponds to the optimal solution.

Another important issue to be addressed in the the optimal sensor placement problem is how the search of the optimal solution is performed. Due to the combinatorial nature of the problem (i.e. find among all the possible sensor configurations, the one which is the best solution) the search strategy is in general computationally time demanding.

A general setup of the problem, valid for all the approaches introduced in this thesis, is presented in the next definition.

Definition 3.8 (Optimal sensor placement problem for fault diagnosis)**.** Given a system model, $\mathcal{M}$, a set of candidate sensors $\mathbf{S}$, the cost function $C(S)$ and the diagnosis specifications, $\mathcal{F}$. The optimal sensor placement problem for fault diagnosis is formulated as

$$\begin{aligned} \min_{S \subseteq \mathbf{S}} \quad & C(S) \\ \text{subject to:} \quad & P(\mathcal{M}, S, \mathcal{F}) = 1 \end{aligned} \tag{3.59}$$

The following chapters introduce several approaches to solve such a problem where different algorithms P and different search strategies are developed. All of them are based on a structural model and the required diagnosis specifications are fault detectability and isolability (i.e. F_D and $\mathbb{F}_I^1$).

3.6 Related Works on Sensor Placement for Fault Diagnosis

In this section, a review of the existing works devoted to sensor placement for fault diagnosis is done. First, it is worth noting that there are only few works devoted to sensor placement for fault diagnosis, and by no means a general methodology that suits for all of them exists. Those works related with structural models are specially highlighted since all the approaches presented in this thesis are based on this kind of models. On the other hand, works devoted to sensor placement for purposes other than fault diagnosis are omitted.

One of the earliest works devoted to the sensor placement problem for fault diagnosis was presented in (Basseville et al., 1987). Sensors are optimally placed for monitoring the eigenvalues of a linear dynamic system. Another more recent work, also focused on linear dynamic systems, was proposed in (Frisk et al., 2009). Here the detectability and isolability capabilities are defined as model properties, instead of residual generator properties. Besides, contrary to many other works, faults in the placed sensors are taken into account.

Another work that handles a linear dynamic model is presented in (Commault et al., 2008). This work is limited to linear structured systems. It uses the structure of the model to work with graph based tools, specifically minimal input separators. However, the computation of minimal input separator presents serious computational complexity issues when the model becomes large. Moreover, as it was pointed out in (Frisk and Krysander, 2007), there are some strong assumptions that are difficult to fulfil in practical situations: the residual must be sensitive to just one fault and the fault could be measurable.

Most of the works aiming at the sensor placement problem for fault diagnosis are based on structural models or use other graph representations to cope with the process description. For instance, in (Raghuraj et al., 1999) a directed graph is used to represent which part of the model is affected by each fault. From this "cause-effect" graph, the sensors needed to diagnose the corresponding fault can be found. In this work, an ad hoc algorithm is presented to find the optimal sensors such that fault detectability and isolability are guaranteed. Another example of works based on graphs, can be found in (Khemliche et al., 2006), where the approach is based on bond graph models.

Next, those approaches based on structural models are discussed. A

distinction will be made between those works that need the set of MSO sets previously generated, and those that need no MSO sets. Remember that generating the set of MSO sets, specially when all possible sensors are assumed to be installed in the system, is a high time consuming task, since the redundancy degree is usually high.

In (Travé-Massuyès et al., 2006) besides the ARR generation algorithm, a method to find the sensors for a best diagnosability degree is developed. This is done by first computing the set of all ARR assuming all possible sensors installed, and then obtaining the diagnosability degree. Next, sensors and their corresponding ARRs are removed until the minimal cardinality sensor set that fulfils the same diagnosability degree is found. The sensor search is improved in (Spanache et al., 2004) by means of a genetic optimisation algorithm. Based on these works, in (Rosich et al., 2007) the search strategy is modified by starting with no sensors and iteratively adding sensors while the MSO sets are incrementally generated. This approach will be presented in Chapter 4. Its main improvement is uselessness of generating all the MSO sets, except for the worst case.

As previously mentioned, the sensor placement problem is in fact a combinatorial problem. Thus, if we are just interested in the best combination according to some cost criterion, the problem can be formulated using binary integer programming. This was first noted in (Bagajewicz et al., 2004) where an extension of the works done by Raghuraj et al. (1999) is performed by applying binary optimisation. On the other hand, the works introduced by Travé-Massuyès et al. (2006) are also improved by means of binary integer programming. In (Sarrate et al., 2007), the problem is redefined as an optimal non-linear binary approach (similar to the one presented by Bagajewicz et al. (2004)) where the main contribution is that the set of MSO sets are involved in the constraint formulation. Similar approaches are presented in (Fijany and Vatan, 2006) and (Rosich et al., 2009c). Both present a linear representation of the same problem formulated in (Sarrate et al., 2007). This means that the detectability and isolability properties are represented from the set of MSO sets as linear constraints, being this its main contribution. The difference between both approaches is that an ad-hoc branch and bound algorithm is used in (Fijany and Vatan, 2006), whilst a standard branch and bound algorithm for mixed integer linear programing are used in (Rosich et al., 2009c). The methods presented in (Sarrate et al., 2007) and (Rosich et al., 2009c) are detailed in Chapter 5.

As previously mentioned, generating the complete MSO set requires a high computation time. So, some works have appeared recently without the requirement of computing any MSO set. These methods ensure that the

MSO sets with the corresponding diagnosis capabilities can be generated after solving the sensor placement problem. Thus, the drawback of computing MSO sets is avoided. In (Krysander and Frisk, 2008), an efficient method to find all the minimal sensors set[3] for maximum fault detectability an isolabilty from a structural model is proposed. The method is based on the partial order defined by the well-constrained part, G_k $(k = 1, \ldots, b)$, of the DM-decomposition (see Section 2.3). On the other hand, in (Yassine et al., 2008) an alternative structural model decomposition, based on gathering equations that can not be isolable (i.e. equations that always come together in an MSO set) is proposed. From this decomposition, it is possible to determine the set of detectable faults and the set of isolable faults. Then, the optimal search on the candidate sensor set is performed. Finally, in (Rosich et al., 2009a), a method that takes into account the causal computability of the unknown variable in the residual generation is developed. This method and the causal computability framework are presented in Chapter 6.

[3] *minimal set* means that no proper subset is a solution.

Chapter 4

Sensor Placement by Incremental MSO Sets Generation

4.1 Introduction

The main objective of the approach presented in this chapter is to propose a method for sensor placement based on the computation of MSO sets. This method improves the efficiency of similar approaches by trying not to compute all the MSO sets when all candidate sensors are installed. This is achieved by developing an algorithm that solves the optimal sensor placement problem by incrementally generating, at each iteration, only those MSO sets that are required. Therefore, a solution can be found before generating the whole set of MSO sets. Moreover, since the search strategy is based on the sensor set cost, the global optimal solution, using a desired cost criterion, is ensured.

Since the corresponding MSO sets are provided, the method is suitable to handle diagnosis specification other than fault detectability and isolability. Actually, any property derived from the set of MSO sets can be used. For example the highest derivative order required in a residual generator, the causality (see Chapter 6) in the variable computation, model uncertainties, the fault sensitivity, etc. Such properties can be verified automatically by an ad-hoc algorithm as long as the required set of MSO sets and some extra model information are provided.

Let $\mathcal{F}$, $\mathcal{M}$ and Ω_S denote, respectively, the diagnosis specifications defined on the MSO sets, the system model containing all the information

required to validate the diagnosis specifications and the set of MSO sets generated from the set of sensors $S \subseteq \mathbf{S}$. Then, a procedure P', similar to (3.58), should be designed:

$$P'(\mathcal{M}, \Omega_S, \mathcal{F}) = \begin{cases} 1 \text{ if diagnosis specifications } \mathcal{F} \text{ are fulfilled} \\ 0 \text{ otherwise} \end{cases} \tag{4.1}$$

The difference with respect to procedure P in (3.58) is that the MSO sets required for a given sensor configuration are now provided.

To find the optimal solution S^*, a coarse algorithm description that solves the problem can be stated as

1. Choose the best sensor configuration, $S_i \subseteq \mathbf{S}$, not previously chosen.

2. Compute the set Ω_{S_i} (e.g. $\Omega_{S_i} = \text{findMSO}((\mathbf{M} \cup M_{S_i})^+, \emptyset)$, according to Algorithm 2.1).

3. If $P'(\mathcal{M}, \Omega_{S_i}, \mathcal{F}) = 0$ then return to step 1. If $P'(\mathcal{M}, \Omega_{S_i}, \mathcal{F}) = 1$ then S_i is the optimal solution, $S^* := S_i$.

Step 1 could be implemented by enumerating all $2^{|\mathbf{S}|}$ possible sensor configurations (if the number of sensors is not large). Step 2 is rather inefficient since generating MSO sets is time demanding and here repeated MSO sets are generated at each iteration. Thus, improving the efficiency in Step 2 will be the main objective of the present chapter. Finally, Step 3 depends on the procedure P', which is especially designed according to the diagnosis specifications. This means that an ad-hoc algorithm must be designed according to the considered diagnosis specifications. For instance, if the diagnosis specifications were fault detectability and isolability for structural models, a procedure based on Corollaries 3.1 and 3.2 could be implemented. Other diagnosis specifications would require the study of other properties in order to develop a suitable procedure P'.

4.2 Relation between Sensors and MSO Sets

As mentioned in Chapter 2, a residual only depends on known variables and can be derived from an MSO set. Then, the sensors needed to compute the corresponding residual can be determined by observing the sensor equations that are included in the corresponding MSO set of equations. Given a set Ω of MSO sets, it is possible to define a relation between two MSO sets

indicating that both MSO sets depend on the same set of sensors. This relation is formally defined as

$$\omega \sim \omega' : \omega \cap M_{\mathbf{S}} = \omega' \cap M_{\mathbf{S}} \tag{4.2}$$

for $\omega, \omega' \in \Omega$

It follows from the definition that this relation is an equivalence relation. Any equivalence relation must fulfil reflexivity, symmetry and transitivity properties. Furthermore, the equivalence relation defines a partition on the set of Ω into equivalent classes. The equivalence class, $[\omega]_S$, is defined from this equivalence relation as,

$$[\omega]_S = \{\omega' \in \Omega \mid \omega \sim \omega'\} \tag{4.3}$$

where the index S tells us which sensors equations are involved in each equivalent class. Thus, according to (4.2) and (4.3) it holds that

$$M_S = \omega \cap M_{\mathbf{S}} \tag{4.4}$$

for any MSO set $\omega \in [\omega]_S$. Furthermore, if there is no MSO set that contains the sensor equations M_S then, for notational convenience, it will be assumed that $[\omega]_S \triangleq \emptyset$.

Let $\mathbf{\Omega_S}$ be the set of all possible MSO sets when all the candidate sensors are installed in the system, i.e. the MSO sets computed from $\mathbf{M} \cup M_{\mathbf{S}}$. Then, the quotient set of $\mathbf{\Omega_S}$ is the set of all possible equivalent classes of $\mathbf{\Omega_S}$ by $\sim$, according to (4.3), and it is denoted as

$$\mathbf{\Omega_S}/\sim = \{[\omega]_S \mid \omega \in \mathbf{\Omega_S}\} \tag{4.5}$$

Next lemma shows that the set of MSO sets obtained from a sensor configuration $S \subseteq \mathbf{S}$ installed in the system can be well defined by the elements of $\mathbf{\Omega_S}/\sim$.

Lemma 4.1. *Let Ω_S be the set of all possible MSO sets obtained from the model $\mathbf{M} \cup M_S$ for $S \subseteq \mathbf{S}$. Then, it holds that*

$$\Omega_S = \bigcup_{S_i \subseteq S} [\omega]_{S_i} \tag{4.6}$$

for $[\omega]_{S_i} \in \mathbf{\Omega_S}/\sim$.

Proof. The proof is straightforward by first noting that $\Omega_S \subseteq \mathbf{\Omega_S}$ since Ω_S is the family of all the MSO sets in $(\mathbf{M} \cup M_S)^+$ and $\mathbf{\Omega_S}$ is the family of all the MSO sets in $(\mathbf{M} \cup M_\mathbf{S})^+$, where $(\mathbf{M} \cup M_S)^+ \subseteq (\mathbf{M} \cup M_\mathbf{S})^+$.

Then, according to (4.2) and (4.3), the set defined by

$$\bigcup_{S_i \subseteq S} [\omega]_{S_i} \tag{4.7}$$

comprises all the MSO sets in $\mathbf{\Omega_S}$ such that no MSO set contains other sensor equations than the ones in M_S, which is the same as saying that they are all the MSO sets in $(\mathbf{M} \cup M_S)^+$, i.e. $\mathbf{\Omega}_S$. □

In the following, it will be stated that the set of sensors, S, generates the set Ω_S of MSO sets. Lemma 4.1 also implies that some of the MSO sets in Ω_S can also be generated by a sensor set other than S. In fact, given two sensor sets, S_1 and S_2, such that $S_1 \subseteq S_2$, it holds that

$$\Omega_{S_2} = (\bigcup_{S_i \subseteq S_1} [\omega]_{S_i}) \cup (\bigcup_{S_i \in 2^{S_2} \setminus 2^{S_1}} [\omega]_{S_i}) = \Omega_{S_1} \cup (\bigcup_{S_i \in 2^{S_2} \setminus 2^{S_1}} [\omega]_{S_i}) \tag{4.8}$$

which implies that Ω_{S_1} can be generated from either S_1 or S_2.

This shows that generating the MSO sets in Ω_S can be efficiently done by regarding the MSO sets generated by subsets of S. Due to the properties of the cost function presented in Definition 3.7, the MSO sets that can be generated by sensor sets with a smaller cost than the actual one can be determined. This is shown in the next theorem.

Theorem 4.1. *Let Ω_{S_1} be the family of MSO sets generated by $S_1 \subseteq \mathbf{S}$. An MSO set*

$$\omega \in \Omega_{S_1} \setminus [\omega]_{S_1} \tag{4.9}$$

can be generated by a sensor set $S_2 \subseteq \mathbf{S}$ such that $C(S_2) < C(S_1)$.

Proof. According to Lemma 4.1 and since two equivalent classes are disjoint sets, i.e. $[\omega]_{S_i} \cap [\omega]_{S_j} = \emptyset$ for $i \neq j$, the set of MSO sets $\Omega_{S_1} \setminus [\omega]_{S_1}$ can be expressed as

$$\Omega_{S_1} \setminus [\omega]_{S_1} = \bigcup_{S_2 \subset S_1} [\omega]_{S_2} \tag{4.10}$$

where regarding the strictly monotonic property of the cost functions, it holds that

$$C(S_2) < C(S_1) \tag{4.11}$$

□

Now, when a sensor configuration S_k is chosen to test the diagnosis specifications, if all sensor configurations with a smaller cost than $C(S_k)$ have been previously chosen, then it can be stated, according to Theorem 4.1, that the MSO sets in $\Omega_{S_k} \setminus [\omega]_{S_k}$ have already been generated in previous iterations. Hence, just the set $[\omega]_{S_k}$ needs to be computed at iteration k. This motivates the fact that the sensor search strategy introduced in step 1 of the coarse description algorithm in Section 4.1 can be performed as follows:

1. Choose the minimal cost sensor configuration, $S_i \subseteq \mathbf{S}$, not previously chosen.

4.3 Algorithm for Optimal Sensor Placement

In this section, the Algorithm 4.1 which solves the optimal sensor placement by incrementally generating the set of MSO sets is presented. The inputs required by the algorithm are the structural system model $\mathbf{M}$ of the system, the set $\mathbf{S}$ of possible sensors to be installed in the system, the cost $C(s)$ of each sensor $s \in \mathbf{S}$ as well as the diagnosis specifications $\mathcal{F}$ and the extra model information $\mathcal{M}$ to call the procedure P'.

First, the algorithm chooses a sensor subset by solving the following combinatorial optimization problem at each k iteration,

$$\min_{S_k \subseteq \mathbf{S}} \{C(S_k) : S_k \in (2^{\mathbf{S}} \setminus \overset{k-1}{\underset{l=1}{\cup}} \{S_l\})\} \tag{4.12}$$

Note that $C(S_k)$ refers to the sensor configuration cost function defined in Definition (3.57). Furthermore, the argument of this optimization problem is the sensor configuration such that it has a minimal cost and it has not been chosen in the previous iterations $\{1, \ldots, k-1\}$.

After the sensor configuration, S_k, has been chosen, the MSO sets in $[\omega]_{S_k}$ are generated. In this step, the structural model $\mathbf{M}$ and the corresponding sensor equations M_{S_k} are provided to compute the required MSO sets, $[\omega]_{S_k}$. To compute this set of MSO sets, one can adapt any of the existing lgorithms mentioned in the introduction chapter, specifically in Section 2.4.2. Here, however, the Algorithm developed in (Krysander and Frisk, 2008) is used. As it was mentioned before, the original algorithm (see Algorithm 2.1) was based on a top-down search, starting with all the equations of the model and then removing equations step by step until an MSO set is found. Now, the difference is that we are only interested in computing those MSO sets that contain all the equations in M_{S_k}. This is accomplished by firstly ensuring

Algorithm 4.1 $S^* = \text{IncrementalSP}(\mathbf{M}, \mathbf{S}, C, \mathcal{F}, \mathcal{M})$

$k := 1$
repeat
  $S_k := \arg\min_{S_k \subseteq S} \{C(S_k) : S_k \in (2^{\mathbf{S}} \setminus \bigcup_{l=1}^{k-1} \{S_l\})\}$ % *Choose a sensor set*
  if $(\mathbf{M} \cup M_{S_k})^+ \supseteq M_{S_k}$ **then**
    $[\omega]_{S_k} := \text{findMSO}((\mathbf{M} \cup M_{S_k})^+), M_{S_k})$ % *Generate* $[\omega]_{S_k}$
  else
    $[\omega]_{S_k} := \emptyset$
  end if
  Store $[\omega]_{S_k}$
  $\Omega_{S_k} := \cup[\omega]_{S'}$ for $S' \subseteq S_k$ % *Build the* $\mathbf{\Omega}_{S_k}$
  $k := k + 1$
until $P'(\mathcal{M}, \Omega_{S_k}, \mathcal{F}) = 1$ **or** $S_k = \mathbf{S}$ % *Verify the diagnosis specifications*
if $P'(\mathcal{M}, \Omega_{S_k}, \mathcal{F}) = 1$ **then**
  $S^* := S_k$ % *The optimal solution is found*
else
  print "There is no solution"
end if

that $M_{S_k} \subseteq (\mathbf{M} \cup M_{S_k})^+$. If this is not the case, then no MSO set containing M_{S_k} exists. Secondly setting the input variable R of the algorithm as $R := M_{S_k}$, which forces the algorithm to not remove any equation in M_{S_k} during the MSO sets computation, i.e. all the computed MSO sets contain the M_{S_k} equations. Since the algorithm was proved to be complete (i.e. all the possible MSO sets are found), then all possible MSO sets $[\omega]_{S_k}$ can be ensured to be found, too. Once the $[\omega]_{S_k}$ has been generated, it is stored in memory since it might be needed in further iterations.

In the next step, the set Ω_{S_k} of MSO sets is obtained from the current generated set $[\omega]_{S_k}$ and the remaining sets $[\omega]_{S'}$ for $S' \subset S_k$. All the remaining MSO sets have already been generated according to Theorem 4.1 and the sensor search strategy of the algorithm. Thus, it is enough generating $[\omega]_{S_k}$. The following three statements are fulfilled when the algorithm is at iteration k:

- All the $\mathbf{\Omega}_{S_k} \setminus [\omega]_{S_k}$ MSO sets have been generated in previous iterations.
- No sensor configuration S' such that $S' \subset S_k$ is a solution.
- If the configuration S_k is a solution, then it is a global optimum.

The algorithm continues iterating until the diagnosis specifications $\mathcal{F}$ are fulfilled. This implies the use of function P' according to (4.1). If no solution is found, the algorithm stops iterating after the sensor configuration $S_k = \mathbf{S}$ has been tested and concludes that there is no solution.

In Section 4.6, issues concerning the efficiency of this algorithm and its applicability to practical cases are discussed.

4.4 Application to the compressor model

An academic example based on the compressor model (see Example 2.2) is presented. It shows how Algorithm 4.1 solves the sensor placement problem for fault diagnosis when the required MSO sets are incrementally generated.

The system model used in this example consists of equations (2.13), where its corresponding structural model is represented in Table 2.1. It is assumed that all system variables are unknown, e.g. no sensors are installed in the original system. The following candidate sensors are defined:

$$\mathbf{S} = \{\omega_c, i, \tau, W, v, p_{in}, p_{out}, T_{in}\} \tag{4.13}$$

Note that all system variables can be measured except for the compressor efficiency η. Some of these variables are easier to measure than others, so a cost to each variable is set accordingly (see Table 4.1).

sensor	ω_c	i	τ	W	v	p_{in}	p_{out}	T_{in}
cost	100	50	200	120	30	20	20	10

Table 4.1: Compressor sensor costs.

Two process faults $\mathbf{F_P} = \{f_1, f_2\}$ are defined on the system model. Fault f_1 refers to a mechanical fault in the electric motor and is related to expression (2.13b) (i.e. $e_{f_1} = e_2$), whereas fault f_2 refers to a compressor box fault in the outlet flow. Thus, it is related to expression (2.13c) (i.e $e_{f_2} = e_3$). The diagnosis specifications are structural detectability and isolability for all process faults in $\mathbf{F_P}$ and all sensors faults as well. As mentioned earlier, fault detectability and isolability can be defined by means of a set of MSO sets, according to Corollaries 3.1 and 3.2. Thus, the algorithm P' should be implemented to check whether expressions (3.11) and (3.16) are fulfilled for

k	S_k	C_{S_k}	$[\omega]_{S_k}$	$\mathbf{\Omega}_{S_k}$
37	$\{i, v, p_{in}, p_{out}, T_{in}\}$	130	$\{\omega_1\}$	$\{\omega_1\}$
67	$\{\omega_c, v, p_{in}, p_{out}, T_{in}\}$	180	$\{\omega_2\}$	$\{\omega_2\}$
70	$\{\omega_c, i, v\}$	180	$\{\omega_3\}$	$\{\omega_3\}$
75	$\{\omega_c, i, v, T_{in}\}$	190	$\emptyset$	$\{\omega_3\}$
76	$\{W, v, p_{in}, p_{out}, T_{in}\}$	200	$\{\omega_4\}$	$\{\omega_4\}$
81	$\{\omega_c, i, p_{in}, p_{out}, T_{in}\}$	200	$\{\omega_5\}$	$\{\omega_5\}$
82	$\{\omega_c, i, v, p_{out}\}$	200	$\emptyset$	$\{\omega_3\}$
83	$\{\omega_c, i, v, p_{in}\}$	200	$\emptyset$	$\{\omega_3\}$
87	$\{\omega_c, i, v, p_{out}, T_{in}\}$	210	$\emptyset$	$\{\omega_3\}$
88	$\{\omega_c, i, v, p_{in}, T_{in}\}$	210	$\emptyset$	$\{\omega_3\}$
91	$\{i, W, p_{in}, p_{out}, T_{in}\}$	220	$\{\omega_6\}$	$\{\omega_6\}$
85	$\{\omega_c, i, v, p_{in}, p_{out}\}$	220	$\emptyset$	$\{\omega_3\}$
99	$\{i, W, v, p_{out}, T_{in}\}$	230	$\{\omega_7\}$	$\{\omega_7\}$
100	$\{i, W, v, p_{in}, T_{in}\}$	230	$\{\omega_8\}$	$\{\omega_8\}$
102	$\{\omega_c, i, v, p_{in}, p_{out}, T_{in}\}$	230	$\emptyset$	$\{\omega_1, \omega_2, \omega_3, \omega_5\}$
105	$\{i, W, v, p_{in}, p_{out}\}$	240	$\{\omega_9\}$	$\{\omega_9\}$
111	$\{i, W, v, p_{in}, p_{out}, T_{in}\}$	250	$\{\omega_{10}, \omega_{11}\}$	$\{\omega_1, \omega_4, \omega_6, \omega_7, \omega_8, \omega_9, \omega_{10}, \omega_{11}\}$

Table 4.2: Results from Algorithm 4.1 at intermediate iterations.

the set of MSO sets, Ω_{S_k}, regarding the fault equations e_2 and e_3, as well as the sensor equations in M_{S_k}.

Once the diagnosis specifications are stated, Algorithm 4.1 is solves the optimal sensor placement problem. Table 4.2 shows the results for those iterations where some MSO sets are obtained, i.e. $\mathbf{\Omega}_{S_k} \neq \emptyset$. The results for the sensors configurations from $k = 0$ to $k = 111$ such that $[\omega]_{S_k} = \emptyset$ and $\mathbf{\Omega}_{S_k} = \emptyset$ are not shown in the table.

The computed MSO sets during the algorithm execution (see Table 4.2) are:

$$\omega_1 = \{e_1, e_2, e_3, e_4, e_5, e_i, e_v, e_{p_{in}}, e_{p_{out}}, e_{T_{in}}\}$$
$$\omega_2 = \{e_1, e_2, e_3, e_4, e_5, e_{\omega_m}, e_v, e_{p_{in}}, e_{p_{out}}, e_{T_{in}}\}$$
$$\omega_3 = \{e_1, e_{\omega_c}, e_i, e_v\}$$
$$\omega_4 = \{e_1, e_2, e_3, e_4, e_5, e_W, e_v, e_{p_{in}}, e_{p_{out}}, e_{T_{in}}\}$$

$$\begin{aligned}
\omega_5 &= \{e_2, e_3, e_4, e_5, e_{\omega_c}, e_i, e_{p_{in}}, e_{p_{out}}, e_{T_{in}}\} \\
\omega_6 &= \{e_2, e_3, e_4, e_5, e_i, e_W, e_{p_{in}}, e_{p_{out}}, e_{T_{in}}\} \\
\omega_7 &= \{e_1, e_2, e_3, e_4, e_5, e_i, e_W, e_v, e_{p_{out}}, e_{T_{in}}\} \\
\omega_8 &= \{e_1, e_2, e_3, e_4, e_5, e_i, e_W, e_v, e_{p_{in}}, e_{T_{in}}\} \\
\omega_9 &= \{e_1, e_2, e_3, e_4, e_5, e_i, e_W, e_v, e_{p_{in}}, e_{p_{out}}\} \\
\omega_{10} &= \{e_1, e_3, e_i, e_W, e_v, e_{p_{in}}, e_{p_{out}}, e_{T_{in}}\} \\
\omega_{11} &= \{e_1, e_2, e_4, e_5, e_i, e_W, e_v, e_{p_{in}}, e_{p_{out}}, e_{T_{in}}\}
\end{aligned} \tag{4.14}$$

where $\{e_{\omega_c}, e_i, e_\tau, e_W, e_v, e_{p_{in}}, e_{p_{out}}, e_{T_{in}}\}$ denotes the corresponding sensor equation set $M_{\mathbf{S}}$ for $\mathbf{S}$ defined in (4.13). Note that the generated MSO sets, $[\omega]_{S_k}$, only contain the sensor equations corresponding to S_k. For example, look at iteration $k = 70$ where $S_{70} = \{\omega_m, i, v\}$ and the generated MSO sets are $[\omega]_{S_{70}} = \omega_3$, then the corresponding sensor equations are

$$\omega_3 \cap M_{\mathbf{S}} = \{e_{\omega_c}, e_i, e_v\} = M_{S_{70}} \tag{4.15}$$

The algorithm stops at iteration $k = 111$ since a solution is found. This means that the solution, found at iteration $k = 111$, is the optimal solution

$$S^* = S_{111} = \{i, W, v, p_{in}, p_{out}, T_{in}\} \tag{4.16}$$

and the MSO sets that generate such a solution are

$$\Omega_{S_{111}} = \{\omega_1, \omega_4, \omega_6, \omega_7, \omega_8, \omega_9, \omega_{10}, \omega_{11}\} \tag{4.17}$$

that fulfil the required diagnosis specifications. To easily verify this, Table 4.3 shows, by means of a cross symbol, which fault equations (regarding both process and sensor faults) belong to each MSO set in $\Omega_{S_{111}}$.

Remark that generating all the MSO sets with all the possible sensors installed in the system would require to generate 46 MSO sets. Here, the proposed algorithm generates 11 MSO sets and 8 out of these 11 are part of the solution.

4.5 Extension to Redundant Sensors

The approach presented in this chapter can easily be extended to redundant sensors. However, it is worth noting that when redundant sensors are considered, the number of MSO sets to be handled grows exponentially and the problem becomes rapidly intractable. On the other hand, the MSO sets

	e_2	e_3	e_i	e_W	e_v	$e_{p_{in}}$	$e_{p_{out}}$	$e_{T_{in}}$
ω_1	×	×	×		×	×	×	×
ω_4	×	×		×	×	×	×	×
ω_6	×	×	×	×		×	×	×
ω_7	×	×	×	×	×		×	×
ω_8	×	×	×	×	×	×		×
ω_9	×	×	×	×	×	×	×	
ω_{10}		×	×	×	×	×	×	×
ω_{11}	×		×	×	×	×	×	×

Table 4.3: Fault equations contained in the MSO sets of $\Omega_{S_{111}}$.

obtained with redundant sensors can be straightforwardly deduced from previous generated MSO sets. There is no need to compute new MSO sets by means of Algorithm 2.1.

Given a structural model $\mathbf{M} \cup M_S$ with some sensors $S \subseteq \mathbf{S}$ installed in the system, the structure is preserved when a sensor is replaced by its corresponding redundant sensor. This is true since both sensor equations depend on the same unknown variable (see Section 3.3.3). Thus, let Ω_S be all MSO sets generated from $\mathbf{M} \cup M_S$. Then, replacing sensor $s \in \mathbf{S}$ by its redundant sensor s' will produce the same MSO sets in Ω_S but replacing sensor equation e_s by its redundant sensor equation $e_{s'}$. This set will be denoted as $\Omega_{S|s'}$, indicating that sensor s has been replaced by its redundant sensor s'. Therefore, it can be stated that

$$\Omega_S \cup \Omega_{S|s'} \subset \Omega_{S\cup\{s'\}} \tag{4.18}$$

The remaining MSO sets in $\Omega_{S\cup\{s'\}}$, not deduced from the non-redundant sensors are those MSO sets that contain both sensor equations, e_s and $e_{s'}$. However, it can be verified that

$$(\{e_s, e_{s'}\})^+ = \{e_s, e_{s'}\} \tag{4.19}$$

and

$$\varphi_s(\{e_s, e_{s'}\}) = 1 \tag{4.20}$$

which shows that the equation set $\{e_s, e_{s'}\}$ is an MSO set. In fact, this is the only MSO set that depends on both sensors. Therefore, when a redundant

sensor is introduced, the set of MSO sets can be deduced as:

$$\Omega_{S\cup\{s'\}} = \Omega_S \cup \Omega_{S|s'} \cup \{e_s, e_{s'}\} \tag{4.21}$$

According to (4.21), the only set to be generated (using Algorithm 2.1) when redundant sensors are handled is Ω_S, the remaining MSO sets can be directly deduced from Ω_S. Next example shows how it should be proceeded in the case of computing MSO sets with redundant sensors.

Example 4.1. Assume the following set of sensors $\{s_1, s_2\}$ and its corresponding redundant sensors $\{s'_1, s'_2\}$. The cost of each sensor is $C(s_1) = C(s'_1) = 2$ and $C(s'_2) = C(s_2) = 5$. The steps to incrementally compute the MSO sets, according to the search strategy of Algorithm 4.1 including redundant sensors should proceed as follows:

1. Choose no sensor and compute $[\omega]_\emptyset$.

$$\Omega_\emptyset = [\omega]_\emptyset$$

2. Choose sensor $\{s_1\}$ and compute $[\omega]_{\{s_1\}}$.

$$\Omega_{\{s_1\}} = [\omega]_\emptyset \cup [\omega]_{\{s_1\}}$$

3. Choose sensor set $\{s_1, s'_1\}$.

$$\Omega_{\{s_1,s'_1\}} = \Omega_{\{s_1\}} \cup \Omega_{\{s_1\}|s'_1} \cup \{e_{s_1}, e_{s'_1}\}$$

4. Choose sensor $\{s_2\}$ and compute $[\omega]_{\{s_2\}}$.

$$\Omega_{\{s_2\}} = [\omega]_\emptyset \cup [\omega]_{\{s_2\}}$$

5. Choose sensor set $\{s_1, s_2\}$ and compute $[\omega]_{\{s_1,s_2\}}$.

$$\Omega_{\{s_1,s_2\}} = [\omega]_\emptyset \cup [\omega]_{\{s_1\}} \cup [\omega]_{\{s_2\}} \cup [\omega]_{\{s_1,s_2\}}$$

6. Choose sensor set $\{s_1, s'_1, s_2\}$.

$$\Omega_{\{s_1,s'_1,s_2\}} = \Omega_{\{s_1,s_2\}} \cup \Omega_{\{s_1,s_2\}|s'_1} \cup \{e_{s_1}, e_{s'_1}\}$$

7. Choose sensor set $\{s_2, s'_2\}$.

$$\Omega_{\{s_2,s'_2\}} = \Omega_{\{s_2\}} \cup \Omega_{\{s_2\}|s'_1} \cup \{e_{s_2}, e_{s'_2}\}$$

8. Choose sensor set $\{s_1, s_2, s_2'\}$.

$$\Omega_{\{s_1,s_2,s_2'\}} = \Omega_{\{s_1,s_2\}} \cup \Omega_{\{s_1,s_2\}|s_2'} \cup \{e_{s_2}, e_{s_2'}\}$$

9. Choose sensor set $\{s_1, s_1', s_2, s_2'\}$.

$$\Omega_{\{s_1,s_1',s_2,s_2'\}} = \Omega_{\{s_1,s_1',s_2\}} \cup \Omega_{\{s_1,s_1',s_2\}|s_2'} \cup \{e_{s_2}, e_{s_2'}\}$$

The set $[\omega_\emptyset]$, $[\omega_{\{s_1\}}]$, $[\omega_{\{s_2\}}]$ and $[\omega_{\{s_1,s_2\}}]$ are computed with Algorithm 2.1, while the remaining MSO sets are deduced from these sets. Note also that, step 9 could be performed from s_1' instead of s_2',

$$\Omega_{\{s_1,s_1',s_2,s_2'\}} = \Omega_{\{s_1,s_2,s_2'\}} \cup \Omega_{\{s_1,s_2,s_2'\}|s_1'} \cup \{e_{s_1}, e_{s_1'}\}$$

which would provide the same result.

Note also that for s_1' to be a redundant sensor, it must be ensured that s_1 is already installed, which means that s_1' comes together with s_1. The same holds for s_2' and s_2. □

4.6 Conclusions

The approach presented in this chapter can be viewed as an improvement of the work in (Travé-Massuyès et al., 2006) since not all the MSO sets need to be computed. Only in the worst case, when the full set of candidate sensors $\mathbf{S}$ is the optimal solution or there is no solution, all the MSO sets are computed. However, since the MSO sets are incrementally computed and no MSO set is computed more than once, it can be said that, in the worst case, Algorithm 4.1 is as efficient as the approach presented in (Travé-Massuyès et al., 2006). Therefore, the efficiency is improved whenever the optimal solution is a subset of $\mathbf{S}$.

In fact, the time required to find the optimal solution depends on the cost of this optimal solution with respect to the other sensor configurations, i.e. the smaller the cost of the optimal solution is, the faster the solution is found. This means that large scale models can be efficiently handled as long as the optimal solution has a low cost. Of course, the cost of the optimal solution can not be known beforehand and intractable problems may be expected. Note that, both the MSO set generation and the sensor set search are time demanding procedures.

On the other hand, the main advantage is that all the required MSO sets are provided for each possible solution. So, this method is suitable to handle

other diagnosis specifications beyond fault detectability and isolability where the computation of MSO sets is required.

Finally, Algorithm 4.1 has been extended to handle redundant sensors. However, the number of MSO sets rapidly grows when redundant sensors are considered and moreover the number of sensor combination to test becomes larger, which makes this approach not efficient for redundant sensors.

Chapter 5

Binary Integer Optimisation for Sensor Placement

5.1 Introduction

The optimal sensor placement can be formulated as a *Binary Integer Programming* (BIP) problem. This class of problems are a particular case of what is known in the literature as *Mixed Integer Programming*. Specifically, a BIP optimisation is based on restricting the value of the optimisation variable to $\{0, 1\}$. Because of this, it is also known as *0-1 Integer Programming*. When solving the optimal sensor placement problem, the installation of a sensor configuration is represented by a binary vector, $\mathbf{q}$, such that each sensor $s_i \in \mathbf{S}$ corresponds to a binary element q_i of $\mathbf{q}$, with $q_i \in \{0, 1\}$. Therefore, if the entry q_i equals 1, the corresponding sensor must be installed whereas if q_i equals 0, the sensor does not have to be installed.

There are some works that have already solved the sensor placement problem by means of BIP formulation. For instance, the problem of placing sensors for process monitoring and fault detection and isolation is addressed in (Bagajewicz et al., 2004). This work departs from a directed graph that indicates whether a fault affects the measurement of a sensor. This information is codified into a BIP formulation in order to obtain the optimal set of sensors such that every fault affects, at least, to one sensor ("fault observability") and two different faults affect to different sensors ("fault resolution"). A similar approach is introduced in this chapter but, here, the MSO sets are considered. The direct relation between faults and sensors is

not as straightforward as in (Bagajewicz et al., 2004). Fault detectability and isolability depend on the available MSO set, and at the same time the MSO set depends on the sensors installed in the system. Therefore, the MSO information has to be codified as constraints in the optimization problem.

The main advantage of using this approach is that, once the constraints are formulated, the optimisation is performed by means of standard algorithms available in applications dedicated to mathematical programming. Thus, we do not need to cope with the search of the optimal solution since it is accomplished by the standard tools of such applications.

Next, the constraint formulation, which is the main part of this approach, is first presented. Then, the sensor placement problem for fault diagnosis will be formally presented as a BIP problem with non-linear constraints. Later, the same problem is transformed into a linear optimisation problem, where the same constraints are reformulated.

5.2 Constraint Formulation for BIP

5.2.1 Preliminary Notation

The present approach accounts for fault detectability and isolability for process and sensors faults. To accomplish this, it will be assumed that the complete set $\mathbf{\Omega_S}$ of all the MSO sets, when all the candidate sensors $\mathbf{S}$ are installed in the system, is available. Recall that such a set can be computed by Algorithm 2.1 with the following inputs,

$$\mathbf{\Omega_S} = \text{findMSO}((\mathbf{M} \cup M_{\mathbf{S}})^+, \emptyset) \tag{5.1}$$

Let n be the number of MSO sets available, i.e. $\mathbf{\Omega_S} = \{\omega_1, \omega_2, \ldots, \omega_n\}$, and let k be the number of candidate sensors to be installed, i.e. $k = |\mathbf{S}|$. Then, a binary matrix $\mathbf{W} = [w_{ij}]$ of size $n \times k$ that relates MSO sets (the row set) and sensors (the column set) is defined. Hence, matrix $\mathbf{W}$ is built from the set of MSO sets as

$$w_{ij} = \begin{cases} 1 & \text{if } e_{s_j} \in \omega_i \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

for all $\omega_i \in \mathbf{\Omega_S}$ and all $s_j \in \mathbf{S}$. Remark that $w_{ij} = 1$ means that ω_i depends on the measurement of the sensor s_j.

A binary matrix $\mathbf{V}$ which relates the MSO sets with the process faults is also defined. Let l be the number of process faults that have to be detected and isolated (i.e. $l = |F_{D_P}|$). Then, the binary matrix $\mathbf{V} = [v_{i,j}]$ of size

$n \times l$ relates process faults (the column set) and MSO sets (the row set), according to

$$v_{ij} = \begin{cases} 1 & \text{if } e_{f_j^p} \in \omega_i \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

for all $\omega_i \in \mathbf{\Omega_S}$ and all process fault $f_j^p \in F_{D_P}$. Note that matrix $\mathbf{V}$ is known in the literature as the *Fault Signature Matrix* (FSM), (Cordier et al., 2004). Since the diagnosis specifications also concern sensor faults , an extra matrix relating MSO sets and sensor faults must be introduced. However, according to the discussion in Section 3.3.2, the equation affected by a sensor fault is the corresponding sensor equation. Then, it turns out that such a matrix is equivalent to $\mathbf{W}$. Thus, the FSM involving both process and sensor faults is built as

$$\begin{pmatrix} \mathbf{V} & \mathbf{W} \end{pmatrix} \tag{5.4}$$

where the first l columns concern process faults and the remaining k columns concern sensor faults.

5.2.2 MSO Set Selector

Not all the MSO sets in $\mathbf{\Omega_S}$ can be used to test the diagnosis specifications when a sensor configuration is selected for installation. This motivates the definition of the *MSO set selector* vector, $\boldsymbol{\rho}$, which is a binary vector of n elements (one for each MSO set) that indicates whether the MSO set is valid, given a candidate sensor set $S \subseteq \mathbf{S}$, i.e.

$$\rho_i = 1 \longleftrightarrow \text{the MSO set } \omega_i \text{ is valid} \tag{5.5}$$

$$\rho_i = 0 \longleftrightarrow \text{the MSO set } \omega_i \text{ is not valid} \tag{5.6}$$

Let $\omega_i \in \mathbf{\Omega_S}$ be an MSO set and let $s_j \in \mathbf{S}$ be any of the sensors that depends on this MSO set ω_i (i.e. $e_{s_j} \in \omega_i$). The MSO set is not valid as long as any of its corresponding sensors is not installed (i.e. $q_j = 0$). Otherwise the MSO set is valid. Thus, ρ_i can be computed as

$$\rho_i = \prod_{j=1}^{k} \Big(w_{ij} \cdot q_j + (1 - w_{ij}) \Big) \tag{5.7}$$

The possible values of each term within the product are shown in the truth Table 5.1. If each product term in (5.7) equals 1 for all the sensors (i.e. for all columns in $\mathbf{W}$) then the MSO set ω_i is valid and ρ_i equals 1, otherwise the MSO set ω_i is not valid and ρ_i equals 0.

w_{ij}	q_j	$w_{ij} \cdot q_j + (1 - w_{ij})$
0	0	1
0	1	1
1	0	0
1	1	1

Table 5.1: Truth table of a valid MSO set.

Example 5.1. Given the compressor model presented in Example 2.2. Now, to reduce the number of MSO sets, assume that the variable set is partitioned into $\mathbf{Y} = \{\omega_m, p_{in}, p_{out}, T_{in}\}$, and $\mathbf{X} = \{i, \tau, W, \eta, v\}$. Therefore, the structural model to consider is the sub-graph

$$G' = (\{e_1, \ldots, e_5\}, \{i, \tau, W, \eta, v\}; A') \tag{5.8}$$

in the biadjacency matrix depicted in Table 2.1. Let the candidate sensor set be $\mathbf{S} = \{i, \tau, W, v\}$. Then, the MSO sets in $\mathbf{\Omega_S}$ are

$$\begin{array}{ll} \omega_1 = \{e_4, e_5, e_\tau, e_W\} & \omega_6 = \{e_2, e_3, e_4, e_5, e_i\} \\ \omega_2 = \{e_3, e_W\} & \omega_7 = \{e_1, e_i, e_v\} \\ \omega_3 = \{e_3, e_4, e_5, e_\tau\} & \omega_8 = \{e_1, e_2, e_\tau, e_v\} \\ \omega_4 = \{e_2, e_i, e_\tau\} & \omega_9 = \{e_1, e_2, e_4, e_5, e_W, e_v\} \\ \omega_5 = \{e_2, e_4, e_5, e_i, e_W\} & \omega_{10} = \{e_1, e_2, e_3, e_4, e_5, e_v\} \end{array} \tag{5.9}$$

Matrix $\mathbf{W}$ is built, according to (5.2), from the MSO sets in (5.9) and represented in Table 5.2, where ones are replaced by $\times$ symbols, and zeros are blank spaces. Assume that the binary vector $[q_1, q_2, q_3, q_4]$ is related to the sensors vector $[i, \tau, W, v]$. Then, according to (5.7), the MSO set selector $\boldsymbol{\rho}$ is computed as

$$\begin{array}{ll} \rho_1 = q_2 q_3 & \rho_6 = q_1 \\ \rho_2 = q_3 & \rho_7 = q_1 q_4 \\ \rho_3 = q_2 & \rho_8 = q_2 q_4 \\ \rho_4 = q_1 q_2 & \rho_9 = q_3 q_4 \\ \rho_5 = q_1 q_3 & \rho_{10} = q_4 \end{array} \tag{5.10}$$

□

In the following, vector $\boldsymbol{\rho} = [\rho_1, \rho_2, \cdots, \rho_n]$ will be used to implicitly select, in the BIP optimisation, those MSO sets that are valid to test the diagnosis specifications for a given sensor configuration.

	i	τ	W	v
ω_1		×	×	
ω_2			×	
ω_3		×		
ω_4	×	×		
ω_5	×		×	
ω_6	×			
ω_7	×			×
ω_8		×		×
ω_9			×	×
ω_{10}				×

Table 5.2: **W** matrix for the compressor model.

5.2.3 Fault Detectability Constraints for BIP

Process Fault Detectability

To detect a process fault, at least one MSO set that contains the corresponding fault equation (see Corollary 3.1) must exist. But now, this property is restricted to valid MSO sets.

Given a process fault $f_j^p \in F_D$ and the MSO set selector $\boldsymbol{\rho}$,the detectability constraint for a process fault is defined as

$$\sum_{i=1}^{n} v_{ij}\rho_i \geq 1 \tag{5.11}$$

This constraint ensures that the corresponding column j of matrix **V** has at least one 1 associated to a valid MSO set. In other words, there exists at least one valid MSO set that makes fault f_j^p detectable.

Example 5.2. Following with Example 5.1, consider now two process faults, f_1^p and f_2^p, which are respectively related to equations e_2 and e_3. The corresponding matrix **V** is represented in Table 5.3, from the MSO sets in (5.9).

According to (5.11), the detectable constraint for fault f_1^p is

$$q_1q_2 + q_1q_3 + q_1 + q_2q_4 + q_3q_4 + q_4 \geq 1 \tag{5.12}$$

whereas for fault f_2^p is

$$q_3 + q_2 + q_1 + q_4 \geq 1 \tag{5.13}$$

	f_1^p	f_2^p
ω_1		
ω_2		$\times$
ω_3		$\times$
ω_4	$\times$	
ω_5	$\times$	
ω_6	$\times$	$\times$
ω_7		
ω_8	$\times$	
ω_9	$\times$	
ω_{10}	$\times$	$\times$

Table 5.3: **V** matrix for the compressor model.

Note, for example, that f_1^p becomes detectable as long as either sensor $i, v \in \mathbf{S}$ are installed and fault f_2^p becomes detectable for any candidate sensor $s \in \mathbf{S}$ installed in the compressor. □

Sensor Fault Detectability

Similar to process fault detectability, a sensor fault is detectable if its corresponding sensor equation appears at least in one valid MSO set. Thus, the elements of matrix **W** are now used instead of those of matrix **V**.

The main difference with respect to a process fault is that now the sensor fault detectability constraint only needs to be active when the corresponding sensor is installed. In other words, if the sensor is not installed then no sensor fault is expected. This is achieved by introducing the optimisation variable, q_j, in the inequality constraint

$$\sum_{i=1}^{n} w_{ij}\rho_i \geq q_j \tag{5.14}$$

Note that when the sensor $s_j \in \mathbf{S}$ is installed ($q_j = 1$) the inequality constraint is equivalent to (5.11). However, when the sensor is not installed ($q_j = 0$) the inequality holds for any value of the corresponding MSO set selector.

Example 5.3. Consider the four sensors in Example 5.1, $\{i, \tau, W, v\}$, and the **W** matrix depicted in Table 5.2, according to (5.9). Then, from (5.14),

the corresponding sensor fault constraints for detectability are respectively:

$$q_1q_2 + q_1q_3 + q_1 + q_1q_4 \geq q_1 \tag{5.15a}$$
$$q_2q_3 + q_2 + q_1q_2 + q_2q_4 \geq q_2 \tag{5.15b}$$
$$q_2q_3 + q_3 + q_1q_3 + q_3q_4 \geq q_3 \tag{5.15c}$$
$$q_1q_4 + q_2q_4 + q_3q_4 + q_4 \geq q_4 \tag{5.15d}$$

□

5.2.4 Fault Isolability Constraints for BIP

Fault Isolability between Process Faults

The fault isolability characterisation based on the set of MSO sets can be derived from Corollary 3.2. Recall that a fault f_1 is isolable from another fault f_2, if there exists at least one MSO set that contains e_{f_1} while it does not contain e_{f_2}. Furthermore, due to the symmetric property of fault isolability (see Section 3.2), if such condition holds and the two faults f_1 and f_2 are detectable, it can also be stated that there exist another MSO set that contains e_{f_2} and does not contain e_{f_1} (i.e. fault f_2 is isolable from fault f_1). Therefore, since all isolable faults must be detectable according to the fault isolability specifications in (3.56), if we consider all valid MSO sets, two process faults $f^p_{j_1} \in F_{k_1}$ and $f^p_{j_2} \in F_{k_2}$ for $F_{k_1}, F_{k_2} \in \mathbb{F}^2_I$ and $k_1 \neq k_2$, are isolable if

$$\sum_{i=1}^{n} \rho_i |v_{ij_1} - v_{ij_2}| \geq 1 \tag{5.16}$$

This inequality ensures that both columns j_1 and j_2 in matrix $\mathbf{V}$ (corresponding to $f^p_{j_1}$ and $f^p_{j_2}$, respectively) are at least different in one row associated to a valid MSO set. In other worlds, there exists at least one MSO set such that it contains one fault equation but not the other one.

Example 5.4. Following with Example 5.2, the isolability constraint for the two process faults, f^p_1 and f^p_2, is

$$q_3 + q_2 + q_1q_2 + q_1q_3 + q_2q_4 + q_3q_4 \geq 1 \tag{5.17}$$

according to (5.16), matrix $\mathbf{V}$ in Table 5.3 and the MSO set selectors in (5.10). □

Fault Isolability between Process and Sensor Faults

Similar constraints can be derived for isolability between a process fault and a sensor fault. However, the constraint only needs to be active when the corresponding sensor is installed. Therefore, given a process fault $f^p_{j_1}$ and a sensor fault $f^s_{j_2}$, both faults are isolable as long as

$$\sum_{i=1}^{n} \rho_i |v_{ij_1} - w_{ij_2}| \geq q_{j_2} \tag{5.18}$$

This inequality is always fulfilled when sensor s_j is not installed (i.e. $q_j = 0$). Otherwise the two faults must be isolable.

Example 5.5. Consider, from Example 5.2, process fault f^p_1 and a sensor fault f^s_3 of the flowmeter measuring the air flow, W. The isolability constraint between these two faults is defined, according to (5.18), as

$$q_2 q_3 + q_3 + q_1 q_2 + q_1 + q_2 q_4 + q_4 \geq q_3 \tag{5.19}$$

where matrix $\mathbf{V}$ is in Table 5.3, matrix $\mathbf{W}$ is in Table 5.2 and the MSO set selectors are in (5.10). □

Fault Isolability between Sensor Faults

Isolability between two sensor fault can be easily deduced from the other fault isolability constraints. In this case, the constraint needs to be active only when both sensors are installed. Let s_{j_1} and s_{j_2} be two arbitrary sensors, such that their corresponding sensor faults are isolable in $\mathbb{F}^2_I$. Then, sensor faults $f^s_{j_1}$ and $f^s_{j_2}$ are isolable if

$$\sum_{i=1}^{n} \rho_i |w_{ij_1} - w_{ij_2}| \geq q_{j_1} q_{j_2} \tag{5.20}$$

Note that, only when both sensors are installed ($q_{j_1} = 1$ and $q_{j_2} = 1$), the right hand side of this inequality becomes 1, and the constraint is fulfilled as long as both faults are isolable. Otherwise, the right hand side becomes 0 and the constraint is always fulfilled.

Example 5.6. From Example 5.3, consider the pair of sensor faults corresponding to W and v (i.e. f^s_3 and f^s_4, respectively). Therefore, from (5.20), matrix $\mathbf{W}$ and matrix $\mathbf{V}$ (depicted in Tables 5.2 and 5.3) and the MSO set selector in (5.10), the isolability constraint for sensor faults f^s_3 and f^s_4 is

$$q_2 q_3 + q_3 + q_1 q_3 + q_1 q_4 + q_2 q_4 + q_4 \geq q_3 q_4 \tag{5.21}$$

□

5.3 BIP Optimisation for Sensor Placement

Now, the sensor placement problem for fault diagnosis can be formulated as a BIP problem using the constraints previously introduced. Given the diagnosis specifications F_{D_P} and $\mathbb{F}_I^2$, the number of constraint concerning fault detectability is equal to the number of faults in F_{D_P} plus the number of sensor faults in F_{D_S},

$$|F_{D_P} \cup F_{D_S}| = l + k \tag{5.22}$$

On the other hand, fault isolability constraints are defined from pairs of faults. This means that all possible detectable faults must be combined in pairs, such that every pair of faults does not belong to the same set $F \in \mathbb{F}_I^2$. Assume that $\mathbb{F}_I^2$ is partitioned in m sets, $\mathbb{F}_I^2 = \{F_1, F_2, \cdots, F_m\}$, then the number of constraints concerning fault isolability is

$$\sum_{i=1}^{m-1} |F_i| \sum_{j=i+1}^{m} |F_j| \tag{5.23}$$

In order to correctly apply expressions (5.16), (5.18) or (5.20), we need to take into account whether the involved faults in each isolability constraint are process or sensor faults.

A cost vector $\mathbf{c}$ is defined based on the cost of each sensor in $\mathbf{S}$: $c_i = C(s_i)$, with $s_i \in \mathbf{S}$. Then, given the cost vector $\mathbf{c}$, matrices $\mathbf{W}$ and $\mathbf{V}$ extracted from the set of MSO sets $\mathbf{\Omega_S}$ and the diagnosis specifications F_{D_P} and $\mathbb{F}_I^2$, the sensor placement problem for fault diagnosis based on BIP optimisation is formulated as:

$$\min_{\mathbf{q}} \quad \mathbf{c}^T\mathbf{q} \quad \text{subject to:} \tag{5.24}$$

$$\rho_i = \prod_{j=1}^{k} \Big(w_{ij} \cdot q_j + (1 - w_{ij}) \Big) \qquad i = \{1, \ldots, n\}$$

$$\sum_{i=1}^{n} v_{ij}\rho_i \geq 1 \qquad j = \{1, \ldots, l\}$$

$$\sum_{i=1}^{n} w_{ij}\rho_i \geq q_j \qquad j = \{1, \ldots, k\}$$

$$\sum_{i=1}^{n} \rho_i |v_{ij_1} - v_{ij_2}| \geq 1 \qquad j_1, j_2 \in \{1, \ldots, l\} \text{ and } f_{j_1}^p, f_{j_2}^p \text{ are isolable in } \mathbb{F}_I^2$$

$$\sum_{i=1}^{n} \rho_i |v_{ij_1} - w_{ij_2}| \geq q_{j_2} \qquad j_1 \in \{1,\dots,l\}, j_2 \in \{1,\dots,k\} \text{ and } f^p_{j_1}, f^s_{j_2} \text{ are isolable in } \mathbb{F}^2_I$$

$$\sum_{i=1}^{n} \rho_i |w_{ij_1} - w_{ij_2}| \geq q_{j_q} q_{j_2} \qquad j_1, j_2 \in \{1,\dots,l\} \text{ and } f^s_{j_1}, f^s_{j_2} \text{ are isolable in } \mathbb{F}^2_I$$

$$q_i \in \{0,1\} \qquad i = \{1,\dots,k\}$$

This problem can be solved, for example, by IBM ILOG CPLEX[1], which is a software application for solving both, linear and non-linear, optimisation problems.

Example 5.7. To show the complete formulation of the problem, consider again the compressor model presented in Examples 5.1 and 5.2. Recall that there are two process faults f^p_1 and f^p_2 and 4 possible sensors to be placed $\{i, \tau, W, v\}$ which can be faulty. Therefore, four sensor faults, f^s_1, f^s_2, f^s_3 and f^s_4 are defined.

The set of MSO sets $\mathbf{\Omega_S} = \{\omega_1, \dots, \omega_{10}\}$ was introduced in (5.9). The cost of the sensors is $C(i) = 50$, $C(\tau) = 200$, $C(W) = 120$ and $C(v) = 30$, therefore the cost vector is

$$\mathbf{c} = \begin{pmatrix} 50 & 200 & 120 & 30 \end{pmatrix}^T \tag{5.25}$$

All faults must be detectable and isolable (which coincides with the maximum diagnosis specifications). Thus, diagnosis specifications are

$$\begin{aligned} F_{D_P} &= \{f^p_1, f^p_2\} \\ F_{D_S} &= \{f^s_1, f^s_2, f^s_3, f^s_4\} \\ \mathbb{F}^2_I &= \{\{f^p_1\}, \{f^p_2\}, \{f^s_1\}, \{f^s_2\}, \{f^s_3\}, \{f^s_4\}\} \end{aligned}$$

The optimisation problem is formulated as

$$\min_{\mathbf{q}} \quad 50q_1 + 200q_2 + 120q_3 + 30q_4 \qquad \text{subject to:} \tag{5.26a}$$

$$q_1q_2 + q_1q_3 + q_1 + q_2q_4 + q_3q_4 + q_4 \geq 1 \tag{5.26b}$$

$$q_3 + q_2 + q_1 + q_4 \geq 1 \tag{5.26c}$$

$$q_1q_2 + q_1q_3 + q_1 + q_1q_4 \geq q_1 \tag{5.26d}$$

$$q_2q_3 + q_2 + q_1q_2 + q_2q_4 \geq q_2 \tag{5.26e}$$

$$q_2q_3 + q_3 + q_1q_3 + q_3q_4 \geq q_3 \tag{5.26f}$$

$$q_1q_4 + q_2q_4 + q_3q_4 + q_4 \geq q_4 \tag{5.26g}$$

[1] Visit `www.ibm.com` for further information.

$$q_3 + q_2 + q_1q_2 + q_1q_3 + q_2q_4 + q_3q_4 \geq 1 \tag{5.26h}$$
$$q_1q_4 + q_2q_4 + q_3q_4 + q_4 \geq q_1 \tag{5.26i}$$
$$q_2q_3 + q_2 + q_1q_3 + q_1 + q_3q_4 + q_4 \geq q_2 \tag{5.26j}$$
$$q_2q_3 + q_3 + q_1q_2 + q_1 + q_2q_4 + q_4 \geq q_3 \tag{5.26k}$$
$$q_1q_2 + q_1q_3 + q_1 + q_1q_4 \geq q_4 \tag{5.26l}$$
$$q_3 + q_2 + q_1q_2 + q_1q_3 + q_1q_4 + q_4 \geq q_1 \tag{5.26m}$$
$$q_2q_3 + q_3 + q_1q_2 + q_1 + q_2q_4 + q_4 \geq q_2 \tag{5.26n}$$
$$q_2q_3 + q_2 + q_1q_3 + q_1 + q_3q_4 + q_4 \geq q_3 \tag{5.26o}$$
$$q_3 + q_2 + q_1 + q_1q_4 + q_2q_4 + q_3q_4 \geq q_4 \tag{5.26p}$$
$$q_2q_3 + q_2 + q_1q_3 + q_1 + q_1q_4 + q_2q_4 \geq q_1q_2 \tag{5.26q}$$
$$q_2q_3 + q_3 + q_1q_2 + q_1 + q_1q_4 + q_3q_4 \geq q_1q_3 \tag{5.26r}$$
$$q_1q_2 + q_1q_3 + q_1 + q_2q_4 + q_3q_4 + q_4 \geq q_1q_4 \tag{5.26s}$$
$$q_3 + q_2 + q_1q_2 + q_1q_3 + q_2q_4 + q_3q_4 \geq q_2q_3 \tag{5.26t}$$
$$q_2q_3 + q_2 + q_1q_2 + q_1q_4 + q_3q_4 + q_4 \geq q_2q_4 \tag{5.26u}$$
$$q_2q_3 + q_3 + q_1q_3 + q_1q_4 + q_2q_4 + q_4 \geq q_3q_4 \tag{5.26v}$$
$$q_i \in \{0,1\} \qquad i = \{1,2,3,4\} \tag{5.26w}$$

Constraints (5.26b) and (5.26c) concern detectability of process faults and constraints (5.26d)-(5.26g) concern detectability of sensor faults. Constraint (5.26h) concerns isolability between process faults, constraints (5.26i)-(5.26p) concern isolability between process and sensors faults and finally constraints (5.26q)-(5.26v) concern isolability between sensor faults.

After solving this optimisation problem with ILOG OPL, the optimal solution is

$$\mathbf{q}^* = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix}^T$$

which corresponds to the set of optimal sensors $S^* = \{i, W, v\}$. By substituting this result into (5.10), the following MSO set selectors are obtained:

$$\begin{array}{ll} \rho_1 = 0 & \rho_6 = 1 \\ \rho_2 = 1 & \rho_7 = 1 \\ \rho_3 = 0 & \rho_8 = 0 \\ \rho_4 = 0 & \rho_9 = 1 \\ \rho_5 = 1 & \rho_{10} = 1 \end{array} \tag{5.27}$$

So the MSO sets needed to fulfil the diagnosis specifications are ω_2, ω_5, ω_6, ω_7, ω_9 and ω_{10}. Remark also that the selected MSO sets are indeed the set of MSO sets generated by the solution S^* (i.e. $\mathbf{\Omega}_{S^*}$). Table 5.4 is built from

the selected MSO sets. Note that all process and sensor faults are detectable and isolable (i.e. the diagnosis specifications are fulfilled). □

	e_2	e_3	e_i	e_W	e_v
ω_2		×		×	
ω_5	×		×	×	
ω_6	×	×	×		
ω_7			×		×
ω_9	×			×	×
ω_{10}	×	×			×

Table 5.4: Fault equations contained in the selected MSO sets.

5.4 Redundant Sensor Extension

The formulation presented in (5.24) can be extended to also involve redundant sensors $\mathbf{S}'$. First, the variable vector is extended to $(\mathbf{q}^T \quad \mathbf{q'}^T)^T$ where $\mathbf{q'} = (q'_1 \cdots q'_k)^T$ represents the redundant sensors. The same sub-index indicates that q'_i represents the redundant sensor of q_i.

A redundant sensor can only be installed when the non-redundant sensor is already installed. This can be formulated by the following constraint,

$$q_i - q'_i \geq 0 \tag{5.28}$$

for all redundant sensors. The only combination that does not fulfil this inequality is for $q_i = 0$ and $q'_i = 1$ which is the case that must be avoided.

Recalling the conclusions drawn in Section 3.3.3, it is known that redundant sensors do not improve process fault specifications. Recall also that when a redundant sensor is placed, both, the non-redundant and the redundant sensor faults, become directly detectable and isolable from the other possible faults. These properties will help to efficiently handle redundant sensors in the BIP formulation.

First, since redundant sensors do not affect process faults, only those constraints that are affected by sensor faults must be redefined. Recall, from Theorem 3.4, that a non-redundant sensor fault becomes directly detectable

when its redundant sensor is placed. Then, constraint (5.14) is redefined as

$$q'_j + \sum_{i=1}^{n} w_{ij}\rho_i \geq q_j \tag{5.29}$$

Now, when the redundant sensor is installed ($q'_j = 1$), the detectability constraint is fulfilled, no matter which MSO sets are valid.

Similar modifications can be applied for isolability between process and sensors faults, and for isolability between sensor faults as well. Thus, the isolability constraint (5.18) between a process and a sensor fault is redefined as

$$q'_{j_2} + \sum_{i=1}^{n} \rho_i (v_{ij_1} - w_{ij_2})^2 \geq q_{j_2} \tag{5.30}$$

and the isolability constraint (5.20) between two sensors faults is redefined as

$$q'_{j_1} + q'_{j_2} + \sum_{i=1}^{n} \rho_i (w_{ij_1} - w_{ij_2})^2 \geq q_{j_1} q_{j_2} \tag{5.31}$$

In conclusion, dealing with redundant sensors entails: extending the optimisation vector to also include the redundant sensors in $\mathbf{S}'$, adding the constraints (5.28), and modifying constrains (5.14), (5.18) and (5.20) according to (5.29), (5.30) and (5.31), respectively. Therefore, no new MSO sets are computed and the set of constraints is not significantly increased. However, the set of optimisation variables is increased, being this the main drawback, since the time required to find a solution for such problems is exponential with the number of optimisation variables.

5.5 Binary Integer Linear Programming for Sensor Placement

Up to now, in this chapter, it has been seen how the sensor placement problem can be formulated as a BIP problem with non-linear constraints. This is basically due to the fact that the MSO set selector is a product of optimisation variables and therefore non-linear. Here, the same problem is transformed into a linear optimisation problem and thus refereed as *Binary Integer Linear Programming* (BILP). So, a standard branch and bound algorithm, which is in general more efficient than the algorithms used for a non-linear optimisation, may be used.

This section is therefore devoted to reformulate the non-linear inequalities, previously introduced, into linear inequalities. For the sake of simplicity, in the continuation it will be assumed that all detectable faults are fully isolable. This means that each set $F_i \in \mathbb{F}_I^2$ involves only one fault (i.e. $|\mathbb{F}_I^2| = l + k$). How to relax such an assumption will be explained at the end of this section.

5.5.1 Standard Binary Integer Linear Programming

The BILP formulation (Wosley, 1998) is very similar to the well known linear programming formulation. A standard optimisation problem using BILP is formulated as a linear objective function and a set of linear inequality constraints in matrix form. This is expressed as:

$$\min_{\mathbf{x}} \quad \mathbf{c}^T\mathbf{x} \quad \text{subject to:} \tag{5.32}$$

$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{5.33}$$

$$\mathbf{x} \text{ is binary} \tag{5.34}$$

where $\mathbf{x}$ is the vector of variables, $\mathbf{c}$ is the vector of variable costs, $\mathbf{A}$ is a matrix and $\mathbf{b}$ is a vector. The difference with linear programming is that the vector of variables is constrained to be binary, i.e. $x \in \{0, 1\}$ for all $x \in \mathbf{x}$.

An standard the branch and bound algorithm developed to solve linear binary optimisation problems consists in an implicitly enumeration of all the possible combinations. Then, an upper and a lower bound are iteratively computed in order to efficiently discard combinations that are not a solution. Usually the upper and the lower bound are based on a constraint relaxation by means of linear programming, where the binary restriction $x \in \{0, 1\}$ is omitted. In general, a solution is found, despite the exponential computational complexity of these algorithms.

In the continuation, the $\mathbf{I}_n$ matrix will denote the identity matrix of order n, whereas $\mathbf{0}_{n\times m}$ and $\mathbf{1}_{n\times m}$ will denote respectively a matrix of zeros and ones, of n rows and m columns.

5.5.2 Linear MSO Set Selector

Clearly, the MSO set selector constraint presented in (5.7) is not linear and therefore it cannot be casted as a linear inequality in (5.33). However, replacing the product by a summation implies that the sum equals k when

the MSO set is valid ($\rho_i = 1$),

$$\sum_{j=1}^{k} [w_{ij} q_j + (1 - w_{ij})] = k \longleftrightarrow \text{The MSO set } \omega_i \text{ is valid} \tag{5.35}$$

On the other hand, a non-valid MSO set ω_i implies that there is at least one term of the summation which equals zero and therefore the summation is smaller than k. The following linear inequality holds as long as the MSO set ω_i is not valid ($\rho_i = 0$),

$$\sum_{j=1}^{k} [w_{ij} q_j + (1 - w_{ij})] < k \longleftrightarrow \text{The MSO set } \omega_i \text{ is not valid} \tag{5.36}$$

Now, a binary variable λ_i is introduced in the inequality (5.36) as follows:

$$\sum_{j=1}^{k} [w_{ij} q_j + (1 - w_{ij})] - k\lambda_i \geq 0 \tag{5.37}$$

Given a non-valid MSO set, inequality (5.37) holds if and only if $\lambda_i = 0$ (according to (5.36)), whereas if the corresponding MSO set is valid then the same inequality holds for both $\lambda_i = 0$ and $\lambda_i = 1$. This implies that

$$\text{MSO set } \omega_i \text{ is not valid} \longrightarrow \lambda_i = 0 \tag{5.38}$$

where the reverse of the implication clause is not necessary true. Hence, λ_i can be viewed as a *dummy* variable in the optimisation problem where it is forced to zero as long as the corresponding ω_i is a non-valid MSO set. Variable λ_i will be called the *linear MSO set selector* and plays the same role as ρ_i in the BIP formulation.

Because equation (5.37) is linear, it can be extended to all the n MSO sets, and written in matrix-vector form as

$$\begin{pmatrix} w_{i1} & \cdots & w_{ik} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nk} \end{pmatrix} \begin{pmatrix} q_1 \\ \vdots \\ q_k \end{pmatrix} - k \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{pmatrix} + \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix} \geq \mathbf{0}_{n \times 1} \tag{5.39}$$

where $\beta_i = \sum_{j=1}^{k} (1 - w_{ij})$ for $i = \{1, \cdots, n\}$. Finally, equation (5.39) can be written in a compact form as

$$\begin{pmatrix} -\mathbf{W} & k\mathbf{I}_n \end{pmatrix} \begin{pmatrix} \mathbf{q} \\ \boldsymbol{\lambda} \end{pmatrix} \leq \boldsymbol{\beta} \tag{5.40}$$

where vector $\boldsymbol{\lambda} = \begin{pmatrix} \lambda_1 & \cdots & \lambda_n \end{pmatrix}^T$ is the set of linear MSO set selectors and $\boldsymbol{\beta} = \begin{pmatrix} \beta_1 & \cdots & \beta_n \end{pmatrix}^T$ is a vector of independent coefficients.

5.5.3 Linear Fault Detectability and Isolability Constraints

Similar constrains to those presented in Section 5.2.3 are also defined here. The main difference is that now there is a linear MSO set selector $\boldsymbol{\lambda}$ in the optimisation vector, which is constrained by a linear inequality. Since inequality constraints (5.11), (5.14), (5.16) and (5.18) are already linear, there is no needed to modify them. Inequality (5.20) is not linear but a straightforward linear equivalent representation will be introduced.

Moreover, since the constraints are linear, they can be written in matrix form. Therefore, in this section, it is shown how to built such matrices based on the elements of matrices $\mathbf{W}$ and $\mathbf{V}$. It is assumed that the pair of indices used for isolability constraints are lexicographically ordered[2].

Lexicographical order is used to sort the Cartesian product of two ordered sets. Given two sets of ordered indices, then any pair of indices can be ordered, according to the lexicographical order, as

$$(i_1, i_2) \prec (j_1, j_2) \leftrightarrow i_1 < j_1 \text{ or } (i_1 = j_1 \text{ and } i_2 < j_2) \tag{5.41}$$

Process Fault Detectability

The constraint introduced in (5.11) for fault detectability is extended to all l process faults in F_{D_P} and written in compact form using the linear MSO set selector,

$$\begin{pmatrix} \mathbf{0}_{l\times k} & -\mathbf{V}^T \end{pmatrix} \begin{pmatrix} \mathbf{q} \\ \boldsymbol{\lambda} \end{pmatrix} \leq -\mathbf{1}_{l\times 1} \tag{5.42}$$

Sensor Fault Detectability

Since the constraint (5.14) for sensor fault detectability is linear, it can be extended to all sensors and written in compact form as

$$\begin{pmatrix} \mathbf{I}_k & -\mathbf{W}^T \end{pmatrix} \begin{pmatrix} \mathbf{q} \\ \boldsymbol{\lambda} \end{pmatrix} \leq \mathbf{0}_{k\times 1} \tag{5.43}$$

Isolability between Process Faults

Let $\mathcal{C}_k^n$ be the number of k-combinations of a set with n elements,

$$\mathcal{C}_k^n = \frac{n!}{(n-k)!k!} \tag{5.44}$$

[2]The term lexicographical order is also known as dictionary order or alphabetic order.

To introduce the isolability constraints involving all possible pairs of process faults in F_{D_P}, let matrix $\mathbf{V_{ff}} = [v_{ff_{im}}]$ be an $n \times \mathcal{C}_2^l$ matrix such that

$$v_{ff_{im}} = |t_{ij_1} - t_{ij_2}| \quad \forall j_1, j_2 \in \{1, \ldots, l\} : j_1 < j_2 \tag{5.45}$$

where m indexes, in lexicographical order, the set of pairs (j_1, j_2).

From matrix $\mathbf{V_{ff}}$, constraint (5.16) can be extended to all combinations of two process faults, and written in compact form as

$$\begin{pmatrix} \mathbf{0}_{\mathcal{C}_2^l \times k} & -\mathbf{V_{ff}}^T \end{pmatrix} \begin{pmatrix} \mathbf{q} \\ \boldsymbol{\lambda} \end{pmatrix} \leq -\mathbf{1}_{\mathcal{C}_2^l \times 1} \tag{5.46}$$

Isolability between Process and Sensor Faults

The constraint (5.18) for isolability between process and sensor faults can be extended to all possible pairs involving one process fault and one sensor fault, and written in compact form as

$$\begin{pmatrix} \mathbf{G_2} & -\mathbf{V_{fs}}^T \end{pmatrix} \begin{pmatrix} \mathbf{q} \\ \boldsymbol{\lambda} \end{pmatrix} \leq \mathbf{0}_{l \cdot k \times 1} \tag{5.47}$$

where $\mathbf{G_2}$ is an $(l \cdot k) \times k$ matrix, built from the concatenation of identity matrices of order k as follows

$$\mathbf{G_2} = \begin{pmatrix} I_k & I_k & \cdots & I_k \end{pmatrix}^T \tag{5.48}$$

and $\mathbf{V_{fs}} = [v_{fs_{ip}}]$ is an $n \times (l \cdot k)$ matrix such that

$$v_{fs_{ir}} = (v_{ij_1} - w_{ij_2})^2 \quad \forall j_1 \in \{1, \ldots, l\}, \forall j_2 \in \{1, \ldots, k\} \tag{5.49}$$

where r indexes, in lexicographical order, the Cartesian product of the process fault set and the sensor fault set. For formulation convenience, the indices concerning process faults are assumed to be lower ordered than the indices concerning sensor faults.

Note that matrix $\mathbf{G_2}$ involves the sensor variable q_{j_2} with the process fault f_{j_1}, according to the index r used in (5.49).

Isolability between Sensor Faults

Linear constraints for isolability between sensor faults are derived from inequality (5.20). Remark that the right hand side of the inequality is not linear. This can be easily solved by transforming that inequality into

$$\sum_{i=1}^{n} (w_{ij_1} - w_{ij_2})^2 \lambda_i \geq q_{j_1} + q_{j_2} - 1 \tag{5.50}$$

Now, both expressions are equivalent in the sense that the right hand side of this expression equals 1 when both sensors are installed ($q_{j_1} = 1$ and $q_{j_2} = 1$). Otherwise, the constraint is always fulfilled. So, the constraint in (5.50) for isolability between two sensor faults is now linear.

To extend constraint (5.50) to all possible combinations of two sensors, let the matrix $\mathbf{V_{ss}} = [v_{ss_{it}}]$ of size $n \times \mathcal{C}_2^k$ be

$$v_{ss_{it}} = (w_{ij_1} - w_{ij_2})^2 \quad \forall j_1, j_2 \in \{1, \ldots, k\} : j_1 < j_2 \tag{5.51}$$

where t indexes in lexicographical order the set of the $\mathcal{C}_2^k$ sensor faults pairs.

Now, equation (5.50) can be extended to any combination of two sensor faults, and written in compact form as

$$\begin{pmatrix} \mathbf{G_3} & -\mathbf{V_{ss}}^T \end{pmatrix} \begin{pmatrix} \mathbf{q} \\ \boldsymbol{\lambda} \end{pmatrix} \leq \mathbf{1}_{\mathcal{C}_2^k \times 1} \tag{5.52}$$

where matrix $\mathbf{G_3}$ involves the corresponding pair of sensors, according to the indices j_1 and j_2. Thus, the matrix $\mathbf{G_3}$ is an $\mathcal{C}_2^k \times k$ matrix, built as

$$\mathbf{G_3} = \begin{pmatrix} 1 & & & & & \\ \vdots & & & \mathbf{I}_{k-1} & & \\ 1 & & & & & \\ 0 & 1 & & & & \\ \vdots & \vdots & & & \mathbf{I}_{k-2} & \\ 0 & 1 & & & & \\ & & & \vdots & & \\ 0 & \cdots & & 0 & 1 & 1 \end{pmatrix} \tag{5.53}$$

5.5.4 BILP Optimisation for Sensor Placement

Once linear detectability and isolability constraints have been introduced as linear inequalities, the optimal sensor placement for fault detection and isolation is formulated as

$$\min_{(\mathbf{q}^T \quad \boldsymbol{\lambda}^T)^T} \quad (\mathbf{c}^T \quad \mathbf{0}_{1\times n}) \begin{pmatrix} \mathbf{q} \\ \boldsymbol{\lambda} \end{pmatrix} \quad \text{subject to:} \tag{5.54}$$

$$\begin{pmatrix} -\mathbf{W} & k\mathbf{I}_n \\ \mathbf{0}_{l\times k} & -\mathbf{V}^T \\ \mathbf{I}_k & -\mathbf{W}^T \\ \mathbf{0}_{\mathcal{C}_2^l\times k} & -\mathbf{V_{ff}}^T \\ \mathbf{G_2} & -\mathbf{V_{fs}}^T \\ \mathbf{G_3} & -\mathbf{V_{ss}}^T \end{pmatrix} \begin{pmatrix} \mathbf{q} \\ \boldsymbol{\lambda} \end{pmatrix} \leq \begin{pmatrix} \boldsymbol{\beta} \\ -\mathbf{1}_{l\times 1} \\ \mathbf{0}_{k\times 1} \\ -\mathbf{1}_{\mathcal{C}_2^l\times 1} \\ \mathbf{0}_{l\cdot k\times 1} \\ \mathbf{1}_{\mathcal{C}_2^k\times 1} \end{pmatrix} \tag{5.55}$$

$$(\mathbf{q}^T \quad \boldsymbol{\lambda}^T) \text{ is binary} \tag{5.56}$$

where constraint (5.55) is the concatenation of (5.40), (5.42), (5.43), (5.46), (5.47) and (5.52) respectively. For n MSO sets, l process faults and k sensors, the number of rows (i.e. constraints) in (5.55) is $n+l+k+\mathcal{C}_2^{l+k}$, according to:

- the *MSO set selector* constraints (5.40) involve n rows,
- the detectability constraints (5.42) and (5.43) involve $l+k$ rows,
- the isolability constraints (5.46), (5.47) and (5.52) involve

$$\mathcal{C}_2^l + l\cdot k + \mathcal{C}_2^k = \mathcal{C}_2^{l+k}$$

  rows.

This formulation is completely equivalent to the non-linear formulation (5.24) presented in Section 5.3. It is worth noting that now the vector of variables is extended to include the MSO set selector $\boldsymbol{\lambda}$. This implies that the cost vector is also extended. However, since the purpose of the optimisation is to find the set of optimal sensors, the cost related to the $\boldsymbol{\lambda}$ is set to zero. By doing this, the selected valid MSO sets in the optimisation problem have no effect in the solution cost. Hence, $\boldsymbol{\lambda}$ is regarded as a *dummy* vector.

Example 5.8. Consider the same problem as the one presented in Example 5.7. Then, the corresponding matrices $\mathbf{W}$ and $\mathbf{V}$ are

$$\mathbf{W} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}; \qquad \mathbf{V} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Note that the number of MSO sets is $n = 10$, the number of sensors to install is $k = 4$ and the number of process fault is $l = 2$. To pose the MSO set selector constraints according to (5.40) vector $\boldsymbol{\beta}$ is computed as

$$\boldsymbol{\beta} = \begin{pmatrix} 2 & 3 & 3 & 2 & 2 & 3 & 2 & 2 & 2 & 3 \end{pmatrix}^T$$

Now, matrices $\mathbf{V_{ff}}$, $\mathbf{V_{fs}}$ and $\mathbf{V_{ss}}$ are computed according to (5.45), (5.49) and (5.51),

$$\mathbf{V_{ff}} = \begin{pmatrix} 0 \\ -1 \\ -1 \\ -1 \\ -1 \\ 0 \\ 0 \\ -1 \\ -1 \\ 0 \end{pmatrix}; \mathbf{V_{fs}} = \begin{pmatrix} 0 & -1 & -1 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & -1 & 0 & -1 \\ 0 & -1 & 0 & 0 & -1 & 0 & -1 & -1 \\ 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & -1 & 0 & -1 & 0 \\ 0 & -1 & -1 & -1 & 0 & -1 & -1 & -1 \\ -1 & 0 & 0 & -1 & -1 & 0 & 0 & -1 \\ -1 & 0 & -1 & 0 & 0 & -1 & 0 & -1 \\ -1 & -1 & 0 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & -1 & 0 & -1 & -1 & -1 & 0 \end{pmatrix}; \mathbf{V_{ss}} = \begin{pmatrix} -1 & -1 & 0 & 0 & -1 & -1 \\ 0 & -1 & 0 & -1 & 0 & -1 \\ -1 & 0 & 0 & -1 & -1 & 0 \\ 0 & -1 & -1 & -1 & -1 & 0 \\ -1 & 0 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & -1 & -1 \\ -1 & 0 & -1 & -1 & 0 & -1 \\ 0 & -1 & -1 & -1 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{pmatrix}$$

Finally, matrices $\mathbf{G_2}$ and $\mathbf{G_3}$ are built according to (5.48) and (5.53),

$$\mathbf{G_2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}; \qquad \mathbf{G_3} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

From all these matrices, and according to (5.54)-(5.56), the optimal sensor placement problem for the compressor system is formulated as

$$\min_{(q_1 \cdots q_4 \quad \lambda_1 \cdots \lambda_n)^T} \quad 50q_1 + 200q_2 + 120q_3 + 30q_4 + 0\lambda_1 + \cdots + 0\lambda_{10}$$

subject to:

$$\left(\begin{array}{cccc:cccccccccc}
0 & -1 & -1 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\
0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\
0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\
\hdashline
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & -1 & -1 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\
\hdashline
1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 \\
\hdashline
0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & 0 \\
\hdashline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\
0 & 1 & 0 & 0 & -1 & 0 & -1 & 0 & -1 & -1 & 0 & 0 & -1 & -1 \\
0 & 0 & 1 & 0 & -1 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & -1 & 0 & 0 & -1 \\
0 & 1 & 0 & 0 & -1 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 \\
0 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & -1 & -1 & 0 & 0 & -1 & -1 \\
0 & 0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & -1 & -1 & -1 & -1 & 0 \\
\hdashline
1 & 1 & 0 & 0 & -1 & 0 & -1 & 0 & -1 & -1 & -1 & -1 & 0 & 0 \\
1 & 0 & 1 & 0 & -1 & -1 & 0 & -1 & 0 & -1 & -1 & 0 & -1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & -1 & -1 & -1 \\
0 & 1 & 1 & 0 & 0 & -1 & -1 & -1 & -1 & 0 & 0 & -1 & -1 & 0 \\
0 & 1 & 0 & 1 & -1 & 0 & -1 & -1 & 0 & 0 & -1 & 0 & -1 & -1 \\
0 & 0 & 1 & 1 & -1 & -1 & 0 & 0 & -1 & 0 & -1 & -1 & 0 & -1
\end{array}\right)
\left(\begin{array}{c}
q_1 \\ q_2 \\ q_3 \\ q_4 \\ \hdashline \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \\ \lambda_7 \\ \lambda_8 \\ \lambda_9 \\ \lambda_{10}
\end{array}\right)
\leq
\left(\begin{array}{c}
2 \\ 3 \\ 3 \\ 2 \\ 2 \\ 3 \\ 2 \\ 2 \\ 2 \\ 3 \\ \hdashline -1 \\ -1 \\ \hdashline 0 \\ 0 \\ 0 \\ 0 \\ \hdashline -1 \\ \hdashline 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \hdashline 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1
\end{array}\right) \tag{5.57}$$

$$\begin{pmatrix} q_1 & \cdots & q_4 & \lambda_1 & \cdots & \lambda_{10} \end{pmatrix} \text{ is a binary vector} \tag{5.58}$$

where the dashed lines show the different matrices and vectors used in this formulation.

This optimisation problem can be solved, for example, by either IBM ILOG CPLEX or MATLAB. In this case the *bintprog* command of the Optimization Toolbox in MATLAB is used. The result returned is

$$\mathbf{q} = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix}^T \tag{5.59a}$$

$$\boldsymbol{\lambda} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}^T \tag{5.59b}$$

The result obtained is the same as in Example 5.7 (i.e. $S^* = \{i, W, v\}$). The difference is that the selected MSO sets are now ω_2, ω_7 and ω_9. This is due to the one-way implication in (5.38), which ensures that a non-valid MSO set is never selected. However, not all valid MSO sets will be selected. □

At the beginning of this section it was assumed that any detectable fault was fully isolable from the others. If this is not the case (i.e. there are some pairs of faults that are non-isolable in the diagnosis specifications $\mathbb{F}_I^2$), then it suffices to remove those row-constraints in (5.55) that are related to the non-isolable pairs of faults.

The linear approach presented here can be easily extended to also cover redundant sensors. In fact, the extra constraints introduced in Section 5.4 to handle redundant sensors are linear. Thus, the same constraints are also applicable to the BILP formulation.

5.5.5 Sensor Placement and MSO Set Optimisation

Since the MSO set selector variable is now in the optimisation vector, it is possible to extend the optimisation to the family of MSO sets. There may exist several reasons for optimising the set of MSO sets. As an example of different criteria for optimizing the set of MSO sets, consider the following:

- Minimising the number of the chosen MSO sets: set the same cost for each MSO set.
- Minimising the complexity of the chosen MSO sets: set a cost proportional to the number of equations involved in each MSO set.
- Maximising the isolability properties of the chosen MSO sets: set a cost proportional to the number of (process and sensor) fault equations contained in each MSO set.

Now, $\boldsymbol{\lambda}$ is not treated as *dummy* vector. Instead, cost for each MSO set is included in the cost vector. Hence, the optimisation problem is reformulated as

$$\min_{(\mathbf{q}^T \quad \boldsymbol{\lambda}^T)^T} \quad (\mathbf{c}^T \quad \mathbf{c}_{\boldsymbol{\lambda}}^T) \begin{pmatrix} \mathbf{q} \\ \boldsymbol{\lambda} \end{pmatrix} \quad \text{subject to: (5.55) and (5.56)} \tag{5.60}$$

where vector $\mathbf{c}_{\boldsymbol{\lambda}} = \begin{bmatrix} c_{\lambda_1} & \cdots & c_{\lambda_n} \end{bmatrix}^T$ is a column-vector of n elements stating the cost of the MSO sets.

Fault Isolability Constraints for MSO Set Optimisation

A problem arises in the fault isolability constraints when both, sensors and MSO sets, are optimised at the same time (as in (5.60)). Since not all the possible MSO sets may be used (only the optimal MSO sets will be used), the symmetric property of fault isolability (see Section 3.2) is no longer ensured. This can be easily showed in the following example.

Example 5.9. Example 5.8 is used, but now the cardinality of the set of selected MSO sets is also optimised (i.e. each MSO set has an associated cost of 1). The problem is formulated as

$$\min_{(q_1 \cdots q_4 \quad \lambda_1 \cdots \lambda_{10})} \quad 50q_1 + 200q_2 + 120q_3 + 30q_4 + \lambda_1 + \cdots + \lambda_{10}$$

$$\text{subject to: (5.57) and (5.58)}$$

The result is the same as in (5.59), $S^* = \{i, W, v\}$ and the set of optimal MSO sets $\Omega^* = \{\omega_2, \omega_7, \omega_9\}$. Table 5.5 shows which fault equations are included in each selected MSO set. In this table, it can be seen, for example, that process fault f_2^p (related with e_3) is not isolable from a fault in sensor W since there is no MSO set ω in Ω^* such that

$$e_3 \in \omega \wedge e_W \notin \omega \tag{5.61}$$

However, a fault in sensor W is isolable from process fault f_2^p since ω_9 contains e_W but not e_3. □

	e_2	e_3	e_i	e_W	e_v
ω_2		×		×	
ω_7			×		×
ω_9	×			×	×

Table 5.5: Fault equations contained in the optimal selected MSO sets.

As the example shows, the symmetric property does not hold as long as residual exoneration (Travé-Massuyès et al., 2006) is not assumed. The residual exoneration assumption is adopted when it can be guaranteed that a fault always affects the residual by exceeding its threshold. However, this

is not very realistic in practical situations, since a fault may not always exceed the residual generator threshold under noise or model uncertainty. Then, the isolability constraints must be reformulated in order to handle the isolability specification when MSO set optimisation is desired within the BILP formulation.

The new isolability constraint formulation can be deduced directly from Corollary 3.2. Let $f^p_{j_1}$ and $f^p_{j_2}$ be two process faults, then $f^p_{j_1}$ is isolable from $f^p_{j_2}$ if

$$\sum_{i=1}^{n} v_{ij_1}(1 - v_{ij_2})\lambda_i \geq 1 \tag{5.62}$$

This constraint is satisfied as long as there is at least one valid MSO set ($\lambda_i = 1$) such that it contains the fault $f^p_{j_1}$ equation ($v_{ij_1} = 1$) but not the fault $f^p_{j_2}$ equation ($v_{ij_2} = 0$). Constraint (5.62) is the counterpart of constraint (5.16). From this, it can be easily deduced the following constraints:

- A process fault $f^p_{j_1}$ is isolable from a sensor fault $f^s_{j_2}$ fault if

$$\sum_{i=1}^{n} v_{ij_1}(1 - w_{ij_2})\lambda_i \geq q_{j_2} \tag{5.63}$$

- A sensor fault $f^s_{j_1}$ fault is isolable from a process fault $f^p_{j_2}$ if

$$\sum_{i=1}^{n} w_{ij_1}(1 - v_{ij_2})\lambda_i \geq q_{j_1} \tag{5.64}$$

  These two constraints, (5.63) and (5.64), are the counterpart of (5.18).

- A sensor fault $f^s_{j_1}$ fault is isolable from a sensor fault $f^s_{j_2}$ fault if

$$\sum_{i=1}^{n} w_{ij_1}(1 - w_{ij_2})\lambda_i \geq q_{j_1} + q_{j_2} - 1 \tag{5.65}$$

  which is the counterpart of constraint (5.20).

Now, since the symmetric property does not hold, given two faults, f_1 and f_2, it is required to check whether f_1 is isolable from f_2 as well as f_2 is isolable from f_1. As a consequence, the number of constraints concerning isolability is extended to the number of permutations of two faults from the set of $l + k$ detectable faults, $\mathcal{P}_2^{l+k}$. Recall that, beforehand, the number of

isolability constraints was at most $\mathcal{C}_2^{l+k}$. Now, optimising sensors and MSO sets requires at most two times more isolability constraints,

$$\mathcal{P}_2^{l+k} = \frac{(l+k)!}{(l+k-2)!} = 2\mathcal{C}_2^{k+l} \tag{5.66}$$

5.5.6 BILP for MSO Set Optimisation

The BILP formulation could also be used to only optimise the set of MSO sets by omitting the sensor placement problem. Given a sensor configuration, not all possible MSO sets are needed to implement a diagnosis system. Usually, a reduced set of MSO sets is sufficient. In this situation the best MSO sets can be chosen by performing such optimisation, based on a criterion as those proposed in Section 5.5.5.

Given a set of n MSO sets Ω and a set of l detectable and fully isolable faults in Ω, the optimal MSO sets are sought such that the diagnosis specifications are preserved. This problem can be easily solved by the following BILP formulation,

$$\min_{\boldsymbol{\lambda}} \quad \mathbf{c}_{\boldsymbol{\lambda}}^T \boldsymbol{\lambda} \quad \text{subject to:} \tag{5.67}$$

$$\begin{pmatrix} -\mathbf{V}^T \\ -\mathbf{V}'^T_{\mathbf{ff}} \end{pmatrix} \boldsymbol{\lambda} \leq \begin{pmatrix} -\mathbf{1}_{l\times 1} \\ -\mathbf{1}_{\mathcal{P}_2^l \times 1} \end{pmatrix} \tag{5.68}$$

$$\boldsymbol{\lambda} \text{ is binary} \tag{5.69}$$

where $\mathbf{V}'_{\mathbf{ff}}$ is an $n \times \mathcal{P}_2^l$ matrix, built from (5.62) by using all $\mathcal{P}_2^l$ fault pairs. Note that now there is no distinction between process and sensor faults. A sensor fault is regarded as a process fault since sensors are assumed already installed in the system.

5.6 Conclusions

The optimal sensor placement problem for fault diagnosis has been presented as a binary integer optimisation problem. This allows us to solve the problem by means of standard algorithms included in powerful packages devoted to non-linear programming. Therefore, there is no need to cope with the optimal search since these tools apply internal heuristics in order to make the search as efficient as possible. However, the nature of the problem is combinatorial, which means that the time required to find a solution is, in general, exponential with the number of variables to optimise.

In any case, the main drawback of this approach is the need of generating all the MSO sets with all the sensors installed, i.e. the set $\mathbf{\Omega_S}$. Note that computing $\mathbf{\Omega_S}$ can be troublesome for systems with a large set of candidate sensors, since the number n of MSO sets grows exponentially with the redundancy degree of the system. In fact, this is usually the main limitation of this approach for systems with a large number of candidate sensors, since the number of all possible MSO sets becomes extremely large. In general, it can be stated that if the set $\mathbf{\Omega_S}$ can be generated, then binary integer programing for sensor placement can be applied with no computational restriction. However, when generating all MSO sets is not feasible, the method can still be applied if only a reduced number of MSO sets is considered.

The same problem has been reformulated by using linear constrains. Similar conclusions can be drawn for the linear approach. The difference is that standard algorithms developed for linear integer programming can be used. Furthermore, due to the fact that the MSO set selector variable, $\boldsymbol{\lambda}$, needs to be included in the optimisation vector, the optimisation is extended to cover both sensors and MSO sets. Hence, the optimal set of sensor to be installed for a given diagnosis specifications and the optimal set of MSO sets to be implemented in the diagnosis system can be obtained in one step. However, the computational complexity is increased. Furthermore, to carry out this double optimisation, fault isolability constraints are reformulated since the symmetric property of fault isolability is lost when not all possible MSO sets are considered in the final solution. This can be avoided if residual exoneration is assumed, though Definition 3.3 is no longer valid.

In the present approach, sensor redundancy is introduced without recomputing any new MSO set in both, non-linear and linear, approaches. Moreover, detectability and isolability for both, process and sensor faults, are addressed as diagnosis specifications. However, the method could be extended to other diagnosis specifications. The only requirement is that these extra specifications should be based on MSO sets and expressed as non-linear inequality constraints in the case of non-linear BIP and as linear inequality constraints in the case of BILP.

Chapter 6

Sensor Placement for Causally Computable MSO Sets

In this chapter, a methodology to solve a particular case of the sensor placement problem for fault detection and isolation is addressed. This particular case deals with the causal computability of unknown variables in the residual computation.

The causal computability of unknown variables is a key issue when deriving residuals for complex systems. Those systems usually involve non-linear equations that must be solved when implementing residual generators. Addressing the causal computation of the unknown variables in each MSO set guarantees that a residual generator can be directly implemented from an MSO set. However, the disadvantage of this approach is that only a sub-class of MSO sets can be used for this purpose. The sensor placement problem for causally computable MSO sets is also addressed in order to find the sensors such that this required sub-class of MSO sets can be generated.

6.1 Causal Computability on Residual Generation

In structural model based diagnosis, consistency may be checked by using a set of redundant sub-models (i.e. MSO sets of equations). A residual generator can be implemented from an MSO set by computing the internal unknown variables through a convenient manipulation of the equations and later checking consistency in a redundant equation. This concept is known (Blanke et al., 2006) as a causal interpretation of the computability. The re-

sult is a directed bi-partite graph, named *computation sequence*, that shows how internal values can be computed from the equations (value propagation) in every redundant sub-model. However, to guarantee that the residual is generated by using non-linear equations, the structural model framework needs to be adapted in order to handle causal computability. Few works focus this causal assignment in the fault diagnosis field. For instance, in (Ploix et al., 2008) causality is taken into account in the computation of the set of redundant sub-models. In (Svärd and Nyberg, 2008) causality is treated in derivative and integral computations by considering which solver tools are available, whereas in (de Flaugergues et al., 2009) the causality of invertible function is fitted in the structural analysis.

An MSO set of equations has the property of a complete matching in the unknown variable set, plus an extra non-matched equation named the redundant equation. This redundant equation is used for checking consistency. As an example, assume the following MSO set (6.1), consisting of three equations (e_1, e_2 and e_3), where $\{y_1, y_2\}$ are the known variable set and $\{x_1, x_2\}$ are the unknown variable set. A possible complete matching of the unknown variable set into the equation set is $\{(e_1, x_1), (e_2, x_2)\}$, so that the redundant equation is e_3.

$$\begin{aligned} e_1 &: x_1 = h_1(y_1) \\ e_2 &: x_2 = h_2(x_1, y_2) \\ e_3 &: h_3(x_1, x_2, y_1) = 0 \end{aligned} \tag{6.1}$$

As mentioned above, one way to obtain a residual from this MSO set is building the computation sequence, which tells how to compute the unknown variables. Following with the example, the matching is interpreted as e_1 solves x_1 and e_2 solves x_2. Figure 6.1 shows the computation sequence derived from this matching. Now, the internal values can be straightforwardly propagated, according to the computation sequence, to compute the residual as

$$r(y_1, y_2) = h_3(h_1(y_1), h_2(h_1(y_1), y_2), y1) \tag{6.2}$$

where,

$$\begin{cases} r(y_1, y_2) \simeq 0 \text{ means that there is consistency} \\ r(y_1, y_2) \not\simeq 0 \text{ means that there is no consistency} \end{cases} \tag{6.3}$$

Using this procedure when designing residual generators in complex systems gives an intuitive idea on how the residual can be computed. However, solving a certain variable in a non-linear equation could be a hard task or even could not be possible, which ultimately poses restrictions on the residual generator design. This means that not all matchings can be used to

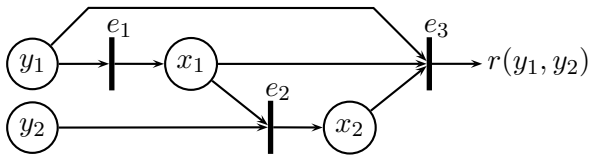


Figure 6.1: Computation sequence.

design a residual generator. For instance, now consider that the matching is $\{(e_3, x_1), (e_2, x_2)\}$ in (6.1). Assume that x_1 cannot be expressed as an explicit variable in e_3. Thus, the value of x_1 cannot be directly computed from x_2 and y_1. So, other tools (e.g. numeric solvers, symbolic manipulation of formulae, etc.) should be used to compute x_1. Therefore no computation sequence can be derived from this matching.

Most of the previous works, that have addressed this topic, state that it suffices the existence of a computation sequence to be able to compute a residual by means of value propagation. However, these works allow the existence of loops inside the computation sequence, which does not guarantee that the computations can be performed. This is shown in next example. Consider the set of non-linear equations (6.4), where x_1 and x_2 are unknown variables and y_1 and y_2 are known variables.

$$\begin{aligned} e_1 &: h_1(x_1, x_2, y_1) = 0 \\ e_2 &: h_2(x_1, x_2, y_2) = 0 \\ e_3 &: h_3(x_1, x_2) = 0 \end{aligned} \tag{6.4}$$

Assume that x_1 can be computed by means of e_1 and x_2 can be computed by means of e_2. A causal matching exists, $\{(e_1, x_1), (e_2, x_2)\}$, and therefore the corresponding computation sequence (see Figure 6.2) can be derived. This means that e_1 and e_2 can be rearranged as

$$\begin{aligned} e_1' &: x_1 = g_1(x_2, y_1) \\ e_2' &: x_2 = g_2(x_1, y_2) \end{aligned} \tag{6.5}$$

Then, propagating variable x_2 into the first equation,

$$x_1 = g_1(g_2(x_1, y_2), y_1) \tag{6.6}$$

where x_1 may not be directly computable as an explicit variable. This situation arises since the computation sequence in Figure 6.2 contains the

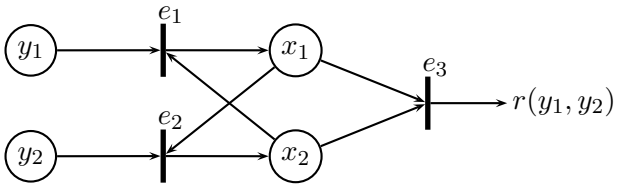


Figure 6.2: Computation sequence with loop.

following loop,

$$e_1 \rightarrow x_1 \rightarrow e_2 \rightarrow x_2 \rightarrow e_1 \tag{6.7}$$

which imposes that variables x_1 and x_2 should be instantiated at the same time since the value of x_1 depends on x_2 and *vice versa*.

This kind of loops is known as *algebraic loops*. It is not always possible to break an algebraic loop and find an analytical expression that solves the unknown variables. An algebraic loop can be solved using other methods like numeric solvers or non-linear optimisation. However, there is no guarantee that these methods find the solution (numeric solvers might not converge and non-linear optimisation might find a local optimum) or the time needed to find the solution can be large. On the other hand, from a structural point of view, it is not possible to know if a given algebraic loop can be broken to derive an explicit analytical expression that permits to solve the unknown variables.

Therefore, residual generators that depend on submodels that imply the inverse computation of non-invertible functions will be excluded. Furthermore, equation subsets that involve both, algebraic and differential, loops in the computation sequence will be excluded as well. So that, no complex solving tools will be needed and the residual computation will be ensured. On the other hand, to keep the simplicity of the approach and at the same time reduce the restrictiveness, sub-models including linear loops will not be excluded, since solving linear equations is not a complex task. In order to cope with causalities in the unknown variable computation, existing structural methods for finding sub-models and computational sequences have to be modified.

The main objective of this approach is to perform diagnosis analysis and sensor placement by taking into account a class of residual generators that

are particularly easy to implement in a diagnosis system based on non-linear system models.

6.2 Causal Structural Model

Knowing when an MSO set can be used to generate a residual (using the computation sequence with no loops, described in the previous section) requires to know which variables can be computed and those that can not be computed in each equation of the model. In this section, it will be shown that this information can be fitted in the structural model by defining a class partition on the set of edges.

In non-linear equations, unknown variables can not always be computed as a function of the others. For instance when non-invertible functions are considered. This leads to introduce the definition of a causally computable variable (or causal variable).

Definition 6.1 (Causal variable)**.** Let $h(X) = 0$ be an equation of the model. Variable $x_i \in X$ is causal in h, if x_i can be computed using h, assuming that the remaining variables, $X \setminus x_i$, are known. It is said that there is a causal relation between x_i and h.

From Definition 6.1 it holds that equation h can never be used in the computation sequence to compute non-causal variables. Furthermore, as discussed in the previous section, causal variables involved in loops are also non-computable, except for loops where all the equations are linear with algebraically independent coefficients. This motivates the following definition.

Definition 6.2 (Linear variable set)**.** Let $h(X) = 0$ be an equation of the model. A set of variables $X_i \subseteq X$ is linear in h if h can be arranged as $\mathcal{L}(X_i) + g(X \setminus X_i) = 0$ and $|X_i| > 1$, where $\mathcal{L}$ is a linear function. It is said that there is a linear relation between X_i and h.

Note that considering one single variable as a linear variable in an equation does not make sense. Linear variables are meant to be considered for identifying linear algebraic loops, and one single linear variable never forms a linear loop. Thus, linear relations are considered whenever two or more linear variables appear in the same equation.

Returning to the set of equations (6.5), assume now that x_1 and x_2 are linear variables in both equations. This means that they can be arranged as $e_1 : \mathcal{L}_{1,1}(x_1) + \mathcal{L}_{1,2}(x_2) = f_1(y_1)$ and $e_2 : \mathcal{L}_{2,1}(x_1) + \mathcal{L}_{2,2}(x_2) = f_2(y_2)$. Then, as long as $[\ \mathcal{L}_{1,1} \quad \mathcal{L}_{2,1}\ ]'$ and $[\ \mathcal{L}_{1,2} \quad \mathcal{L}_{2,2}\ ]'$ are linearly independent,

unknown variables can be easily computed or the loop can be broken (e.g. inverting the matrix or using the Gaussian elimination method).

From the discussion so far, it can be concluded that a set of unknown variables will be computed in a computable sequence as long as the following two conditions hold:

Condition 1: There exists a complete matching on the unknown variables such that there is a causal relation between all equation-variable pairs in the matching.

Condition 2: There are no loops involving causal and/or non-causal variables. The only loops that can appear must just involve linear variables.

Condition 1 ensures the existence of a computation sequence. Condition 2 ensures that the corresponding computation sequence has no loops involving non-linear equations.

The *causal structural model* is now formalised as a bipartite graph where $M = \{\dots, e_i, \dots\}$ is the vertex set of model equations, $X = \{\dots, x_j, \dots\}$ the vertex set of unknown variables, and A the set of edges, such that $(e_i, x_j) \in A$ as long as $x_j \in X$ appears in equation $e_i \in M$. Information on causal and linear relations can be well fitted in the structural model by an equivalent class partition on the set of edges, such that

$$A = A_L \cup A_\times \cup A_\Delta \tag{6.8}$$

where, according to the previous definitions:

- $A_L = \{(e, x) \in A \mid x \text{ is a linear variable in } e\}$. This class of edges will be represented in the biadjacency matrix by an "L" symbol.
- $A_\times = \{(e, x) \in A \mid x \text{ is a causal (but not linear) variable in } e\}$. The cross,"$\times$", symbol will be used to represent this class of edges.
- $A_\Delta = A \setminus (A_L \cup A_\times)$. This class gathers the remaining subset of edges, where a $(e, x) \in A_\Delta$ means that the unknown variable x can not be computed (is non-causal) in e. This relation will be represented by a "Δ" symbol in the biadjacency matrix.

This approach differs from previous works on causal computability in the fact that now loops in the computation sequence are taken into account. This allows rejecting a special class of loops (those that involve non-linear

equations) in the computation sequence, whereas loops involving linear relations are preserved.

Knowing when a variable can be causally computed in an equation may depend on the application and the required specifications. For instance, computing the derivatives of a variable with respect to the time is not desirable whenever measurements contain noise. On the other hand, when measurements are smooth or properly filtered, computing derivatives might be possible. Stating when a variable can be considered computable is out of the scope of this approach. Here generic tools to handle causality are presented and it is assumed that the causal assignment is given.

6.3 Extracting the Causally Computable Part

Given a structural model, it should be known whether a set of unknown variables can be computed when causal and linear relations are considered (according to Conditions 1 and 2). Let $G = (M, X; A)$ be a causal structural model. First, for the sake of simplicity, assume that there are no linear relations (i.e. $A = A_\times \cup A_\Delta$). Condition 1 holds if there exist a complete matching $\mathcal{M}_G^X$ in X, such that

$$\mathcal{M}_G^X \subseteq A_\times \tag{6.9}$$

Let $\partial^M \mathcal{M}_G^X$ be the subset of equations in M incident to edges in $\mathcal{M}_G^X$. Then, Condition 2 holds if the just-determined sub-graph $G' \subseteq G$ such that

$$G' = (\partial^M \mathcal{M}_G^X, X; A') \tag{6.10}$$

has no loops involving edges in $\mathcal{M}_G^X$. This is equivalent to say that there is no *consistent component* (see Section 2.3), $G_k \subseteq G' (= G'^0)$ for $k = 1, \ldots, b$, that contains more than one equation,

$$|M_k| = |X_k| = 1 \qquad k = 1, \ldots, b \tag{6.11}$$

for $G_k = (M_k, X_k; A')$, $M_k \subseteq \partial^M \mathcal{M}_G^X$ and $X_k \subseteq X$.

If a complete matching with such properties exist then the set of unknown variables, X, can be computed using the computation sequence without loops. Note that, this means that the set of equations and the set of variables can be rearranged such that the biadjacency matrix has a triangular form with a diagonal of "$\times$" symbols. Figure 6.3 shows this pattern where all unknown variables can be evaluated.

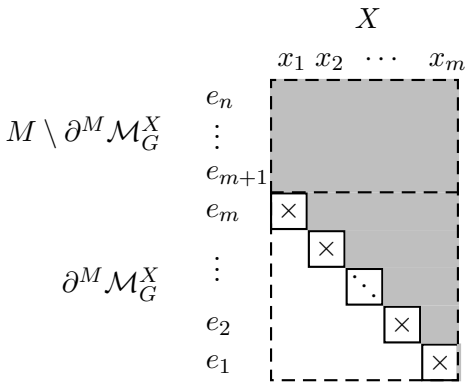


Figure 6.3: Causally computable structure with no linear relations.

Algorithm 6.1 searches for the set of equations $\mathcal{E}_C \subseteq M$ that can be used to compute the causal variables in X. This is iteratively done by finding equations that only contain one causal variable,

$$e \in M : |\mathrm{var}_X(e)| = 1 \wedge \{e, \mathrm{var}_X(e)\} \in A_\times \tag{6.12}$$

Note that, according to (6.9) and (6.12), it holds that

$$(e, \mathrm{var}_X(e)) \in \mathcal{M}_G^X$$

After finding equation e, the graph is pruned and the algorithm continues searching for more equations until no more equation-variable pairs can be found. Finally, the algorithm returns the set of equations, $\mathcal{E}_C$, containing all the equations in M that can be used to compute the set $\mathcal{X}_C$ of causal variables. A similar algorithm to find a matching with no loops, called *ranking algorithm*, was presented in (Blanke et al., 2006).

Here, it is assumed that a set of variables can be solved if every variable can be matched to an equation using a causal edge (Condition 1 holds) and there are no algebraic loops (Condition 2 holds), so any *consistent component* concerning more than one equation-variable pair is rejected.

Now, assume that linear relations are considered (i.e. $A = A_L \cup A_\times \cup A_\Delta$). Since a subset of linear variables can be solved in an algebraic loop, no restrictions are applied for linear relations.

Algorithm 6.1 $\mathcal{E}_C = \text{CausalModel}(M, X)$

$\mathcal{X}_C := \emptyset$
while $\exists e \in M : |\text{var}_X(e)| = 1 \wedge (e, \text{var}_X(e)) \in A_\times$ **do**
  $X := X \setminus \text{var}_X(e)$
  $\mathcal{X}_C := \mathcal{X}_C \cup \text{var}_X(e)$
end while
$\mathcal{E}_C := \{e \in M \mid \text{var}_X(e) \subseteq \mathcal{X}_C\}$

Since linear variables and their relations are of interest, the set of equations, E_L, that depends on these linear variables is first identified

$$E_L = \{e \in M \mid (e, x) \in A_L; \forall x \in \text{var}_X(e)\} \tag{6.13}$$

The Dulmage-Mendelsohn decomposition can be applied to determine the subset of computable variables in a causal structural model. In fact, given the set E_L of linear equations, the equations in the just-determined and over-determined parts can be used to compute linear variables, whereas the equations in the under-determined can not. Thus, the set of equations $\mathcal{E}_L$ to compute linear variables is determined as

$$\mathcal{E}_L = E_L^0 \cup E_L^+ \tag{6.14}$$

Remark that this holds under the assumption that the linear coefficients of these equations are algebraically independent. Note that, according to (2.29), the linear computable variables $\mathcal{X}_L \subseteq X$ can be directly determined as

$$\mathcal{X}_L = \text{var}_X(\mathcal{E}_L) = \text{var}_X(E_L^0) \cup \text{var}_X(E_L^+) \tag{6.15}$$

Algorithm 6.2 summarises how to compute the set of equations, $\mathcal{E}_L$.

Algorithm 6.2 $\mathcal{E}_L = \text{LinearModel}(M, X)$

$E_L := \{e \in M \mid (e, x) \in A_L; \forall x \in \text{var}_X(e)\}$
$\mathcal{E}_L := (E_L^0 \cup E_L^+)$

The diagonal matching presented in Figure 6.3 can be now improved by taking into account the set of linear computable variables. The resulting structural decomposition is shown in Figure 6.4 where the triangular form remains, but now the *consistent components* can take more than one equation-variable relation as long as the variables are linear. The unknown variables $\mathcal{X} \subseteq X$ involved in this matching are computable. The equations in $\mathcal{E} \subseteq M$ can be used to compute $\mathcal{X}$ and are determined as

$$\mathcal{E} = \{e \in M \mid \text{var}_X(e) \subseteq \mathcal{X}\} \tag{6.16}$$

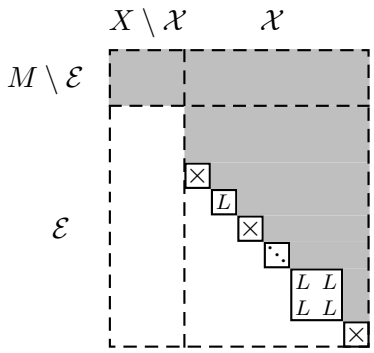


Figure 6.4: Computable structure with causal and linear relations.

The corresponding sub-graph $(\mathcal{E}, \mathcal{X}; A')$ in Figure 6.4 will be called the *causally computable* sub-model.

Algorithm 6.3 finds the causally computable sub-model $\mathcal{E}$ of model M. This is done by iteratively alternating between Algorithm 6.1 and Algorithm 6.2, and pruning the graph in both sets (the set of equations and the set of variables) until no more computable equations, either causal or linear, can be found. Afterwards, the set $\mathcal{X}$ of computable variables can be determined from $\mathcal{E}$ as

$$\mathcal{X} = \mathrm{var}_{\mathbf{X}}(\mathcal{E}) \tag{6.17}$$

Algorithm 6.3 $\mathcal{E} = \mathrm{ComputableModel}(M)$

$\mathcal{E} := \emptyset$
$X = \mathrm{var}_{\mathbf{X}}(M)$
repeat
  $\mathcal{E}_C := \mathrm{CausalModel}(M \setminus \mathcal{E}, X)$
  $X := X \setminus \mathrm{var}_{\mathbf{X}}(\mathcal{E}_C)$
  $\mathcal{E} := \mathcal{E} \cup \mathcal{E}_C$
  $\mathcal{E}_L := \mathrm{LinearModel}(M \setminus \mathcal{E}, X)$
  $X := X \setminus \mathrm{var}_{\mathbf{X}}(\mathcal{E}_L)$
  $\mathcal{E} := \mathcal{E} \cup \mathcal{E}_L$
until $\mathcal{E}_C \cup \mathcal{E}_L = \emptyset$

From the discussion above, it is clear that all remaining equations, $M \setminus \mathcal{E}$, are not useful anymore in this approach, since they contain variables, $X \setminus \mathcal{X}$,

that can not be computed in a computation sequence. Furthermore, remark that extracting the computation sequence given by sub-graph $(\mathcal{E}, \mathcal{X}; A)$, (decomposed as in Figure 6.4) is straightforward since now the matching-diagonal establishes a true interpretation of which equation solves each variable.

6.4 Detectability and Isolability with Causal Relations

This section is devoted to explain how to determine the set of detectable faults and the set of isolable faults when causal computations are taken into account in the residual computation.

Clearly, there exists a complete matching in the variables of the causally computable sub-model $\mathcal{E}$, see Figure 6.4. Hence, there is no under-determined part in $\mathcal{E}$ (i.e. $\mathcal{E}^- = \emptyset$). Remark that this matching is a causal matching, all the edges represent either causal relations with no loops or linear relations with possible loops. Because the matching is complete, it follows that the over-determined set of equations, $\mathcal{E}^+$, contains part of this matching (variables in $\mathcal{E}^+$ are also causally computable). In fact, $\mathcal{E}^+$ is the part of the model that is causally computable and contains redundancy. Fault diagnosis analysis based on the Dulmage-Mendelsohn decomposition can be performed on the equation set $\mathcal{E}$.

6.4.1 Causal Structural Detectability

According to Theorem 3.1, all fault equations in the over-determined part correspond to detectable faults. The same argument can be applied to the causally computable sub-model $\mathcal{E}$ of the model M when causal computations are considered. Thus, fault detectability in the causal framework is next defined.

Definition 6.3 (Causally detectable fault)**.** A fault f is causally detectable in a structural model M if the corresponding fault equation e_f is in the over-determined part of the causally computable sub-model,

$$e_f \in \mathcal{E}^+ \tag{6.18}$$

where $\mathcal{E} = \text{ComputableModel}(M)$, according to Algorithm 6.3.

Given a causal structural model M and a set F of faults, Algorithm 6.4 finds all causally detectable fault equations. The algorithm first computes

the causally computable model $\mathcal{E}$ by means of Algorithm 6.3, and then expression (6.18) is applied to find the causally detectable fault equations, $M_{F_\mathcal{D}} \subseteq \mathbf{M}$.

Algorithm 6.4 $M_{F_\mathcal{D}} = \text{CausalDetectability}(M, M_F)$

$\mathcal{E} := \text{ComputableModel}(M)$
$M_{F_\mathcal{D}} := \mathcal{E}^+ \cap M_F$

The set of causally detectable faults $F_\mathcal{D}$ can be directly determined from its corresponding fault equation set,

$$F_\mathcal{D} = \{f \in F \mid e_f \in M_{F_\mathcal{D}}\} \tag{6.19}$$

6.4.2 Causal Structural Isolability

Isolability analysis is based on Theorem 3.2. Given a model M and a set of detectable faults F_D, a fault $f_i \in F_D$ is structurally isolable from $f_j \in F_D$ if f_j is detectable in the sub-model $M \setminus \{f_i\}$. The same reasoning can be applied to fault isolability when causal computations are considered.

Definition 6.4 (Causally isolable fault)**.** A fault f_i is causally isolable from a fault f_j in a structural model M, if fault f_i is causally detectable in the causally computable sub-model of $M \setminus \{e_{f_j}\}$,

$$e_{f_i} \in \mathcal{E}^+_{f_j} \tag{6.20}$$

where $\mathcal{E}_{f_j} = \text{ComputableModel}(M \setminus \{e_{f_j}\})$, according to Algorithm 6.3.

Now, the symmetric isolability property (see Section 3.2) does not hold for causal isolability (a fault f_i that is causally isolable from a fault f_j does not imply that f_j is causally isolable from f_i) since the causally computable sub-model is not preserved when removing different non-isolable fault equations. Consequently, isolability characterisation $\mathbb{F}^2$ cannot be applied in the causallythe computable framework. This is illustrated in the next example.

Example 6.1. Assume the following causal structural model represented in Table 6.1 with $M = \{e_1, e_2, e_3, e_4\}$, where the causally detectable faults f_1 and f_2 affect respectively equations e_1 and e_2 ($e_{f_1} = e_1$ and $e_{f_2} = e_2$). Applying Algorithm 6.3 for $M \setminus \{e_1\}$ and $M \setminus \{e_2\}$ yields

$$\mathcal{E}_{f_1} = \text{ComputableModel}((M \setminus \{e_1\}) = \{e_2, e_3, e_4\}$$
$$\mathcal{E}_{f_2} = \text{ComputableModel}((M \setminus \{e_2\}) = \{e_4\}$$

	x_1	x_2
$f_1 \rightarrow e_1$	Δ	Δ
$f_2 \rightarrow e_2$	$\times$	$\times$
e_3	Δ	$\times$
e_4		$\times$

Table 6.1: Non-symmetric isolability example.

Then, the over-determined part of each set is computed by means of the DM-decomposition,

$$\mathcal{E}^+_{f_1} = \{e_2, e_3, e_4\}$$
$$\mathcal{E}^+_{f_2} = \emptyset$$

where, according to Definition 6.4, f_2 is isolable from f_1 whereas the reverse does not hold. Thus, the symmetry property is not satisfied.

□

Algorithm 6.5 computes the causally isolable faults for each fault in $F_{\mathcal{D}}$. First, the algorithm computes the causally computable sub-model from the equation set $M \setminus \{e_f\}$ and then finds the causally isolable faults from f by applying (6.20).

Algorithm 6.5 $M_{F_{\mathcal{I}}} = \text{CausalIsolability}(M, M_F)$

$M_{\mathbb{F}^1_{\mathcal{I}}} := \emptyset$
for all $e_f \in M_F$ **do**
  $\mathcal{E}_f = \text{ComputableModel}((M \setminus \{e_f\})$
  $M_{F_{\mathcal{I}}}(f) := \mathcal{E}^+_f \cap M_F$
end for

The output of Algorithm 6.5 is a family of fault equation sets where each $M_{F_{\mathcal{I}}}(f)$ contains the fault equations corresponding to the faults that are isolable from f. Therefore, the family of isolable sets $\mathbb{F}^1_{\mathcal{I}} = \{\dots, F_{\mathcal{I}}(f_i), \dots\}$ can be built as

$$F_{\mathcal{I}}(f_i) = \{f \in F \mid e_f \in M_{F_{\mathcal{I}}}(f_i)\} \tag{6.21}$$

6.5 Sensor Placement based on Causal Relations

Given a causal structural model, the causal detectability and isolability of the model can be computed by using Algorithms 6.4 and 6.5, respectively. In this section, a method to solve the sensor placement when causal computations are considered will be introduced.

Sensors are chosen for installation by adding their corresponding sensor equation to the system equation set. Remark that now a causal relation exists between a sensor equation e_s and the unknown variable that becomes measured,

$$(e_s, \mathrm{var}_{\mathbf{X}}(e_s)) \in A_{\times} \tag{6.22}$$

6.5.1 Maximum Causal Detectability and Isolability Specifications

According to Section 3.4, maximum detectability specification is ensured when all candidate sensors are installed in the system. Hence, causally detectable faults and causally isolable faults can be determined beforehand. However, there may be some useless sensors that do not improve diagnosis capabilities when causality is taken into account. Let $\mathcal{E}_S$ be the causally computable model of a structural system model $\mathbf{M}$ with all possible sensors $\mathbf{S}$ installed in the system, (i.e. $\mathcal{E}_S = \mathrm{ComputableModel}(\mathbf{M} \cup M_{\mathbf{S}})$). Then the set of useful sensors $S_0 \subseteq \mathbf{S}$ is determined according to

$$S_0 = \{s \in \mathbf{S} \mid e_s \in \mathcal{E}_S^+\} \tag{6.23}$$

Therefore, there is no need to consider further sensors from $\mathbf{S} \setminus S_0$ in the sensor placement analysis when causality is considered.

Given a structural model $\mathbf{M}$ of the system and the set of faults $\mathbf{F}$ (involving both, process faults $\mathbf{F_P}$ and sensor faults $\mathbf{F_S}$, i.e. $\mathbf{F} = \mathbf{F_P} \cup \mathbf{F_S}$), maximum causal detectability is computed by Algorithm 6.4 when all useful sensors S_0 are chosen for installation,

$$M_{F_{\mathcal{D}_{max}}} = \mathrm{CausalDetectability}((\mathbf{M} \cup M_{S_0}), M_{\mathbf{F}}) \tag{6.24}$$

Then, maximum causally detectable faults are determined by

$$F_{\mathcal{D}_{max}} = \{f \in \mathbf{F} \mid e_f \in M_{F_{\mathcal{D}_{max}}}\} \tag{6.25}$$

Remark that all useful sensor faults are causally detectable since $\mathcal{E}_S^+ \subseteq M_{F_{\mathcal{D}_{max}}}$ holds.

Maximum causal isolability is also determined when all sensors S_0 are installed in the system. Thus, Algorithm 6.5 can be used to compute the maximum causal isolability among the maximum detectable faults,

$$M_{F_{\mathcal{I}_{max}}} = \text{CausalIsolability}((\mathbf{M} \cup M_{S_0}), M_{F_{\mathcal{D}_{max}}}) \tag{6.26}$$

Then, maximum causal isolability $\mathbb{F}^1_{\mathcal{I}_{max}} = \{\dots, F_{\mathcal{I}}(f_i), \dots\}$ is determined by

$$F_{\mathcal{I}}(f_i) = \{f \in \mathbf{F} \mid e_f \in M_{F_{\mathcal{I}_{max}}}(f_i)\} \tag{6.27}$$

In the following, maximum causal detectability, $F_{\mathcal{D}_{max}}$, and maximum causal isolability, $\mathbb{F}^1_{\mathcal{I}_{max}}$, will be the assumed diagnosis specifications for the sensor placement problem when causal computations are considered in the residual implementation. However, remark that lower specifications than the maximum ones could also be used in this approach. Furthermore, for notational convenience, it will be assumed that non-useful sensors have been removed before the sensor placement analysis (i.e. $\mathbf{S} \triangleq S_0$).

6.5.2 Sensor Placement Algorithm

The main idea to solve the sensor placement problem is computing causally detectable faults and the causally isolable faults for a given sensor configuration and then test whether these faults fulfil the diagnosis specifications. Let $S_k \subseteq \mathbf{S}$ be a sensor configuration. Then, the causally detectable faults are computed as

$$M_{F_{\mathcal{D}}} = \text{CausalDetectability}((\mathbf{M} \cup M_{S_k}), M_{F_{\mathcal{D}_{max}}}) \tag{6.28}$$

Recall that detectable faults of the non installed sensors should not be considered. Then, causal detectability specifications are fulfilled as long as it holds that

$$M_{F_{\mathcal{D}}} = M_{F_{\mathcal{D}_{max}}} \setminus M_{\bar{S}_k} \tag{6.29}$$

where $M_{\bar{S}_k}$ are the sensor equations of the not installed sensors, $\bar{S}_k = \mathbf{S} \setminus S_k$. Expression (6.29) is fulfilled as long as all process faults and all sensors S_k faults in $M_{F_{\mathcal{D}_{max}}}$ are detectable.

Causal isolability can be similarly reasoned. Let $M_{F_{\mathcal{I}}}$ be the causal isolability for sensor configuration S_k,

$$M_{F_{\mathcal{I}}} = \text{CausalIsolability}((\mathbf{M} \cup M_{S_k}), M_{F_{\mathcal{D}_{max}}}) \tag{6.30}$$

Then, the causal isolability specifications are fulfilled if

$$M_{F_{\mathcal{I}}}(f) = M_{F_{\mathcal{I}_{max}}}(f) \setminus M_{\bar{S}_k} \qquad \forall f : e_f \in M_{F_{\mathcal{D}}} \tag{6.31}$$

A procedure P similar to the one introduced in (3.58) is designed. Now, the diagnosis specifications to be fulfilled are maximum causal detectability and isolability of both process and installed sensor faults. Algorithm 6.6 summarises all this procedure and returns a logic value: 1 if the diagnosis specifications are fulfilled and 0 otherwise.

Algorithm 6.6 isFeasible$(S_k, \mathbf{S}, \mathbf{M}, M_{F_{\mathcal{D}_{max}}}, M_{F_{\mathcal{I}_{max}}})$

$\bar{S}_k := \mathbf{S} \setminus S_k$
$M_{F_{\mathcal{D}}} = \text{CausalDetectability}((\mathbf{M} \cup M_{S_k}), M_{F_{\mathcal{D}_{max}}})$
if $M_{F_{\mathcal{D}}} \neq M_{F_{\mathcal{D}_{max}}} \setminus M_{\bar{S}_k}$ **then**
  return 0 % *Causal detectability specifications are not fulfilled*
end if
$M_{F_{\mathcal{I}}} = \text{CausalIsolability}((\mathbf{M} \cup M_{S_k}), M_{F_{\mathcal{D}_{max}}})$
for all $e_f \in M_{F_{\mathcal{D}}}$ **do**
  if $M_{F_{\mathcal{I}}}(f) \neq M_{F_{\mathcal{I}_{max}}}(f) \setminus M_{\bar{S}_k}$ **then**
    return 0 % *Causal isolability specifications are not fulfilled*
  end if
end for
return 1 % *Causal detectability and isolability specifications are fulfilled*

The same search strategy adopted for the incremental approach in Algorithm 4.1 is here applied but now using Algorithm 6.6 to verify the diagnosis specifications. Algorithm 6.7 performs the optimal sensor search for maximum causal detectability and isolability specifications.

Algorithm 6.7 $S^* = \text{CausalSensorPlacement}(\mathbf{M}, \mathbf{S}, C, M_{F_{\mathcal{D}_{max}}}, M_{F_{\mathcal{I}_{max}}})$

$k := 1$
repeat
  $S_k := \arg\min_{S_k \subseteq \mathbf{S}} \{C(S_k) : S_k \in (2^{\mathbf{S}} \setminus \bigcup_{l=1}^{k-1} \{S_l\})\}$
  $k := k + 1$
until isFeasible$(S_k, \mathbf{S}, \mathbf{M}, M_{F_{\mathcal{D}_{max}}}, M_{F_{\mathcal{I}_{max}}}) = 1$
$S^* := S_k$

The algorithm first chooses the set of sensors with the fewest cost, S_k, not previously chosen and then tests whether the set of causally detectable and isolable faults, computed for the sensor configuration S_k, satisfy the maximum causal detectability and isolability specifications. If satisfied, then the solution is found. Otherwise the algorithm continues searching for the next set of sensors until the optimal solution is found.

It is worth noting that no MSO set is generated for this approach. Therefore the MSO set computation burden is avoided. Note also that the same algorithm can be applied when causality is not considered. This would imply modifying Algorithm 6.6 in order not to consider causality.

6.5.3 Redundant Sensor Extension

Extending the approach to handle redundant sensors can be straightforwardly done by increasing the initial set of sensors to also cover the redundant sensors, i.e. $\mathbf{S} \cup \mathbf{S}'$. However, as in the other approaches, considering redundant sensors increases the number of sensor combinations to test and therefore the computing time of the algorithm is, in general, exponentially increased. How redundant sensors affect the causal detectability and isolability properties of the system will be next investigated.

Let $\mathcal{E}_{S_0}$ be the computable part of the system model when a set of useful sensors S_0 is installed, and let $\mathcal{E}_{S_0 \cup \{e_{s'}\}}$ be the causally computable part of the same model when an extra redundant sensor s' is added. Since a redundant sensor can only be used to compute an already measured variable, no new variables are computed by installing redundant sensors. This implies that

$$\mathcal{E}_{S_0 \cup \{e_{s'}\}} = \mathcal{E}_{S_0} \cup \{e_{s'}\} \tag{6.32}$$

From this expression, it is easy to see that installing redundant sensors does not contribute to improve the causal detectability nor the causal isolability of process faults. Furthermore, according to (6.23), $M_{S_0} \subseteq \mathcal{E}_{S_0}$. Thus expression (6.32) can be rewritten as

$$\mathcal{E}_{S_0 \cup \{e_{s'}\}} = (\mathcal{E}_{S_0} \setminus M_{S_0}) \cup M_{S_0} \cup \{e_{s'}\} \tag{6.33}$$

which is equivalent to expression (3.44). Therefore, Theorem 3.4 also holds for the causal approach. Consequently, when a sensor and its corresponding redundant sensor are installed in the system, it can be concluded that both sensor faults are causally detectable, and moreover any causally detectable fault is causally isolable from any of these two sensor faults. This means that Algorithm 6.6 does not need to verify whether the diagnosis specifications concerning a sensor and its redundant sensor faults are fulfilled, since according to the above discussion, they become automatically detectable and isolable from any other fault.

6.6 Optimal Sensor Search Improvement

The optimal sensor search in Algorithm 6.7 is now improved by designing an ad-hoc algorithm. This algorithm uses two heuristics based on properties of sensor placement for diagnosis.

The first heuristic is related with the monotonicity of the sensor cost function. The second heuristic entails a restriction on the type of faults to be considered.

Heuristic 1: Given two sensor sets, S_1 and S_2, such that $S_1 \subset S_2$, then it holds that

$$C(S_1) < C(S_2) \tag{6.34}$$

Heuristic 2: As long as only process faults are considered, if S_1 is not a solution for the sensor placement then any sensor set S_2 such that $S_2 \subset S_1$ neither is a solution,

$$S_1 \text{ is not a solution} \rightarrow S_2 \text{ is not a solution} \qquad \forall S_2 \subset S_1 \tag{6.35}$$

According to Definition 3.7, Heuristic 1 directly holds. On the other hand, to intuitively motivate Heuristic 2, note that the more sensors are installed in the system, the larger becomes the number of unknown variables that can be causally computed. In fact, it holds that

$$\mathcal{E}_{S_2} \subseteq \mathcal{E}_{S_1} \tag{6.36}$$

where $\mathcal{E}_{S_1}$ is the computable part of the system model $\mathbf{M} \cup M_{S_1}$ (with S_1 sensors chosen for installation) and $\mathcal{E}_{S_2}$ is the computable part of the system model $\mathbf{M} \cup M_{S_2}$ (with S_2 sensors chosen for installation). Then, according to Lemma 3.3, it holds that

$$\mathcal{E}^+_{S_2} \subseteq \mathcal{E}^+_{S_1} \tag{6.37}$$

Recall that process faults are always related to system equations in $\mathbf{M}$. Therefore, if a process fault is not detectable (or isolable) with S_1 installed in the system, then, according to (6.37), it will not be detectable (or isolable) with S_2 installed in the system.

Heuristic 2 does not hold for sensor faults. It may happen that sensor faults fulfil the diagnosis specifications with S_2 installed, whereas a fault in sensor $s \in S_1 \setminus S_2$ is not detectable with S_1 installed. For this reason, the present algorithm will only concern process faults.

The search space containing all possible sensor configurations is represented by the nodes in a graph-tree. The root node represents the sensor

configuration with all candidate sensors. The tree is built from the root node by first removing sensors with a greater cost. Thereby, nodes with a smaller cost are first built.

Every node in the tree consists of two sensor sets:

- $node.S$, the sensor configuration that the node represents (i.e. a possible solution to test).
- $node.R$, the sensors that are allowed to be removed in its sub-nodes.

An upper bound, $\bar{B}$, and a lower bound, $\underline{B}$, are defined for every node. The upper bound is the cost of the sensor configuration to test,

$$\bar{B}(node) = C(node.S) \tag{6.38}$$

while the lower bound is the lowest reachable cost by exploring sub-nodes. This is computed as

$$\underline{B}(node) = C(node.S) - C(node.R) \tag{6.39}$$

Example 6.2. A small example with four sensors, $\mathbf{S} = \{s_1, s_2, s_3, s_4\}$, will be used to illustrate the proposed search algorithm. Assume the following sensor costs: $C(s_1) = 6$, $C(s_2) = 5$, $C(s_3) = 2$ and $C(s_4) = 1$. The cost of a sensor configuration follows from Definition 3.7. The corresponding graph-tree is depicted in Figure 6.5. The tree building sequence is indicated in the figure by a circled number near each node. For each node, the elements contained in $node.R$ are indicated in bold.

□

The search strategy is based on a depth-first search by choosing first the nodes with lowest costs and back-tracking to other not already explored nodes when a branch exploration is aborted. Throughout the search, the best solution is updated in S^*, whenever a feasible solution with lower cost than the current best one is found. A branch exploration is terminated at some node when any of the following two conditions is fulfilled:

Condition 1: The node is not a feasible solution for the sensor placement problem.

Condition 2: The lower bound $\underline{B}$ of the current node is not lower than the cost of the current best solution: $\underline{B}(node) \geq C(S^*)$.

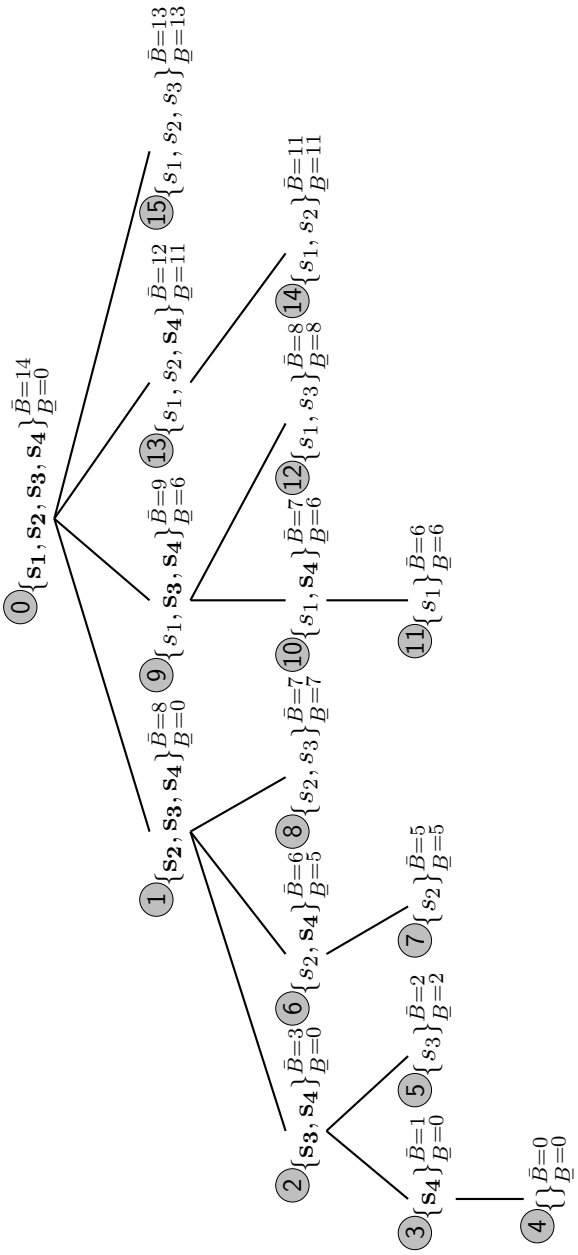


Figure 6.5: Search tree for $\{s_1, s_2, s_3, s_4\}$.

These two conditions are motivated by Heuristics 1 and 2. According to (6.35), if a node is not a feasible solution then its sub-nodes are neither a feasible solution, so the branch exploration is aborted.

On the other hand, concerning Condition 2, remark that the lower bound $\underline{B}$ of the node indicates the cost of the best sub-node in the branch. If Condition 2 is fulfilled, then there is no better sub-node in the branch than the current best solution S^*. Therefore, it does not worth going on exploring this branch, although there could exist a feasible solution (but not optimal).

Algorithm 6.8 recursively performs the optimal search for the sensor placement. Variable *node* is initialised with the set of all candidate sensor ($node.S := \mathbf{S}$, $node.R := \mathbf{S}$). The best sensor configuration is also initialised as $S^* := \mathbf{S}$, which ensures that the maximum diagnosis specifications are fulfilled. The algorithm generates sub-nodes by removing a sensor from $node.R$. The depth-first search is performed by first choosing the sub-node with the lowest cost. If the current sub-node does not fulfil Conditions 1 and 2, then deeper sub-nodes are recursively explored. Otherwise, the remaining sub-nodes are rejected (the branch exploration is aborted) and the next lowest cost sub-node is chosen. And so on until all the sub-nodes are tested or discarded.

Algorithm 6.8 $S^* = \text{searchOptimal}(node, S^*, M'_{F_{\mathcal{D}_{max}}}, M'_{F_{\mathcal{I}_{max}}})$

for all $s \in node.R$ ordered in decreasing cost **do**
  $childNode.S := node.S \setminus \{s\}$
  $node.R := node.R \setminus \{s\}$
  $childNode.R := node.R$
  if $\underline{B}(childNode) < C(S^*)$ **and**
  isFeasible$(childNode.S, \emptyset, \mathbf{M}, M'_{F_{\mathcal{D}_{max}}}, M'_{F_{\mathcal{I}_{max}}})$ **then**
    if $C(childNode.S) < C(S^*)$ **then**
      $S^* := childNode.S$ % *update the best solution*
    end if
    $S^* := searchOptimal(childNode, S^*, M'_{F_{\mathcal{D}_{max}}}, M'_{F_{\mathcal{I}_{max}}})$
  end if
end for
return S^*

Now, sensors faults are not considered in the maximum causal diagnosis specifications. Therefore, maximum causal detectability and isolability are redefined by means of its corresponding fault equations as

$$M'_{F_{\mathcal{D}_{max}}} = M_{F_{\mathcal{D}_{max}}} \setminus M_{\mathbf{S}} \tag{6.40}$$

iteration	current node	cost	cutting condition	best node	cost
0	0	14	—	0	14
1	1	8	—	1	8
2	2	3	(1)	1	8
3	6	6	(1)	1	8
4	8	7	—	8	7
5	9	9	—	8	7
6	10	7	—	8	7
7	11	6	—	11	6
8	12	8	(2)	11	6
9	13	12	(2)	11	6
10	15	13	(2)	11	6

Table 6.2: Optimal search example.

$$M'_{F_{\mathcal{I}_{max}}}(f) = M_{F_{\mathcal{I}_{max}}}(f) \setminus M_{\mathbf{S}} \tag{6.41}$$

Remark that Algorithm 6.6 is used to verify if the diagnosis specifications are fulfilled. However, any other algorithm could be used as long as property (6.35) is ensured. This means, for instance, that the same search algorithm can be used when causal computations are not considered.

The global optimal solution is guaranteed since all possible branches are considered and sub-nodes are only discarded (the search tree is pruned) when it is ensured that no better solution can be found. The following example shows a sample search of Algorithm 6.8.

Example 6.3. Following with Example 6.2, assume that the sensor configurations that fulfil the diagnosis specifications are:

$$\begin{array}{ccccc} \{s_1,s_2,s_3,s_4\} & \{s_2,s_3,s_4\} & \{s_1,s_3,s_4\} & \{s_1,s_2,s_4\} & \{s_1,s_2,s_3\} \\ \{s_2,s_3\} & \{s_1,s_4\} & \{s_1,s_3\} & \{s_1,s_2\} & \{s_1\} \end{array}$$

Table 6.2 shows the sequence that Algorithm 6.8 follows to find the optimal solution for this example.

After completing the sequence, the algorithm returns node 11 with $S^* = \{s_1\}$. Remark that the tree in Figure 6.5 is only used to illustrate the depth-first search procedure. The algorithm does not need to built the whole tree. □

The algorithm efficiency depends on where the optimal solution is located in the search tree. In other words, the nearer the solution is respect to the root node or the leave nodes, the faster it is found. The efficiency also depends on the cost of the sensor configuration. If all the sensors have similar costs then the number of nodes with the same cost becomes bigger, which hinders the task of rejecting nodes using Condition 2.

Given a set of k candidate sensors to be installed in the system, (i.e. $|\mathbf{S}| = k$) and assuming that the optimal solution involves $k/2$ sensors. The worst case is achieved when the algorithm needs to traverse all the $n/2$-combinations of n sensors to find the optimal solution.

6.7 Generating Causally Computable MSO Sets

After solving the sensor placement problem under causal relations, the set of MSO sets can be generated. However not all possible MSO sets will be useful to generate residuals using the computation sequence with non causal loops. An MSO set that is useful in the causal framework (i.e. a residual can be generated by means of the computation sequence, according to the discussion in Section 6.1), will be called *causally computable MSO set* (or causal MSO set).

The set of causal MSO sets is a subset of all possible MSO sets. Therefore, a possible method to obtain the causal MSO sets would be an exhaustive search among all the MSO sets in order to select those MSO sets that are causally computable. So, knowing whether a MSO set is causally computable could be done by means of Algorithm 6.3. Given an MSO set ω, the algorithm returns the causally computable sub-model. Then, the MSO set is causally computable if it holds that

$$\omega = \text{ComputableModel}(\omega) \tag{6.42}$$

However, generating all the MSO sets and then rejecting those that are not causally computable is not an efficient method. In this section, a modification of Algorithm 2.1 is presented in order to only compute the causally computable MSO sets.

The new part in Algorithm 6.9 comprises Steps 2-4. The *if*-condition in Step 3 is used to reject all over-determined sets that have no computable part. At the same time avoids finding repeated causal MSO sets.

Since Algorithm 2.1 finds all possible MSO sets, it can be stated that the present new algorithm finds all possible causally computable MSO sets. It is worth noting that the inputs of the algorithm are a causally computable

Algorithm 6.9 $\mathbf{\Omega_c} = \text{findCausalMSO}(M, R)$

```
1: Ω_c := ∅;
2: E = ComputableModel(M)    % Select the computable part of the model
3: if (M \ E^+) ∩ R = ∅ then
4:    M := E^+    % Update the model with the over-determined computable part
5:    if φ_s(M) = 1 then
6:       Ω_c := {M}
7:    else
8:       while R ⊉ M do
9:          Select an e ∈ M \ R
10:         E := M \ (M \ {e})^+
11:         if E ∩ R = ∅ then
12:            R := R ∪ E
13:            Ω_c := Ω_c ∪ findCausalMSO(M \ E, R)
14:         else
15:            R := R ∪ E
16:         end if
17:      end while
18:   end if
19: end if
20: return Ω_c
```

structural model M and the empty vector of already removed equations, $R = \emptyset$.

6.8 Conclusions

In this chapter, the computation of unknown variables when deriving residuals have been addressed. The novelty of this approach is that causal and linear relations between variables and equations are taken into account in order to guarantee the computability of the set of residuals later needed in the practical implementation of the diagnosis system. The approach leads to a restricted class of residuals. But, in compensation, an easy implementation is ensured where only forward value propagation is required. From a theoretical point of view, this can be viewed as a lack of completeness since, compared to the non-causal approach, some diagnosis capabilities might be lost. However, the presented approach is more focused on practical applications, where a simpler implementation prevails. Specially, for large-scale systems with non-linear equations. On the other hand, the degree of re-

strictiveness is chosen by the designer who decides how to assign causality according to the available tools for residual evaluation.

Since the approach might be too restrictive, the sensor placement problem is addressed. Sensors are placed in order to guarantee that a set of causal MSO sets are generated to detect and isolate faults. Our experience applying this approach to large scale systems shows that usually it suffices to install a few number of extra sensors to be able to obtain satisfactory results.

First, Algorithm 6.7 has been introduced to solve the sensor placement. This algorithm is just an adaptation of Algorithm 4.1 for the incremental MSO set generation: sensors are optimally added until the specifications are fulfilled. For this reason, Algorithm 6.7 presents similar computational drawbacks as Algorithm 4.1. Next, Algorithm 6.8 has been developed to improve the efficiency of the optimal search. This new algorithm is in general faster than the former, since it uses efficient heuristics to discard some sensor configurations. The main drawback is that sensors faults and redundant sensors are not considered. However, we believe that this methodology could be easily extended to cope with these issues.

It is worth noting that no residual or MSO set is computed when solving the sensor placement problem. This makes the presented approach more efficient compared to methods that require MSO set computation.

Furthermore, an adaptation of the MSO sets generation algorithm to only compute the causal MSO sets has been proposed. Although the algorithm is still time demanding for systems with certain redundancy degree, it significantly reduces the computation time that would require generating all MSO sets and later rejecting those that are not causal.

In the next chapter, a case study based on this approach is investigated. The results obtained will be compared with the ones obtained when applying alternative approaches described in the previous Chapters. Thus, advantages and drawbacks will be highlighted.

Chapter 7

Fuel Cell Stack System Application

Nowadays, fuel cell systems are in increasing interest in the scientific community as an alternative to conventional production fuel-fossil based energy. The energy produced by fuel cells is included in the so-called renewable energies and has serious possibilities to be the future energy. For this reason, many works on fuel cells technology have been recently published. However, there are only few works on diagnosis applied to fuel cell systems.

The aim of this chapter is double. Firstly, the sensor placement analyisis and residual generator design are performed on the Fuel Cell Stack (FCS) system. Secondly, a systematic methodology to implement residual generators for large scale systems is presented.

7.1 Fuel Cell Stack Description

A brief introduction of the basic operation principles of a fuel cell stack is given in this section. For further details on fuel cells, the reader is refereed to (Barbir, 2005), (EG&G Technical Services and Corporation, 2002) and (Larminie and Dicks, 2003).

A fuel cell is an electrochemical energy converter that converts the chemical energy of fuel into electrical current. A fuel cell is in some sense similar to a battery. It has an electrolyte, a negative electrode and a positive electrode, and it generates direct electrical current through an electrochemical reaction. The difference is that a fuel cell produces energy as long as reactants are supplied. Typical reactants for fuel cells are hydrogen as fuel and oxygen as oxidant. However, the ambient air is commonly used as an

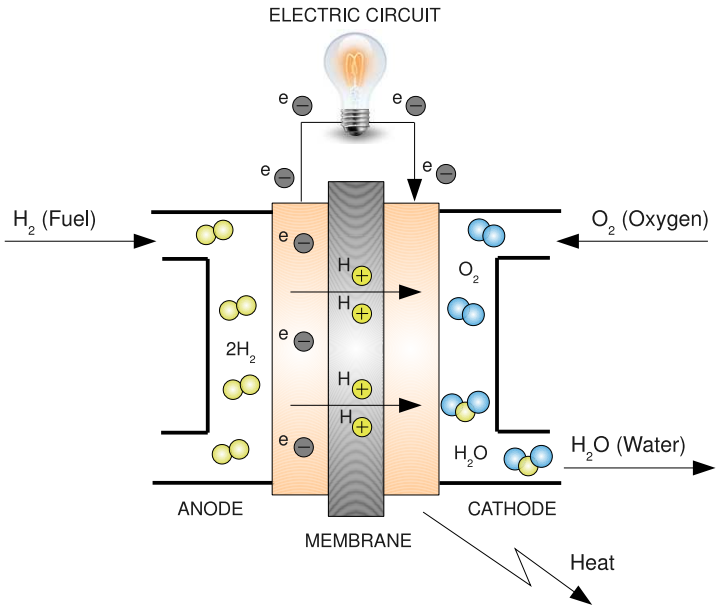


Figure 7.1: Basic principle of a PEM fuel cell.

oxidant since it contains enough oxygen for the reaction. Once the reaction takes place, the fuel cell generates waste heat and water, being this the main attraction of fuel cell systems as a clean power source.

The basic physical structure of a fuel cell consists of an electrolyte layer in contact with a porous anode and cathode electrode plates. A schematic representation of a fuel cell sandwich structure, and its basic operation principle is shown in Figure 7.1. There are different kinds of electrolyte layers. Here a PEM (Polymer Electrolyte Membrane or Proton Exchange Membrane) fuel cell is used. The PEM has a special property: it conducts protons but is impermeable to gas (the electrons are blocked through the membrane). Hydrogen gas is continuously fed to the anode electrode and, with the help of a catalyst, it splits into electrons and hydrogen protons,

$$2H_2 \rightarrow 4H^+ + 4e^-$$

Protons travel through the membrane, whereas electrons travel through an external electrical circuit, producing electrical energy, and come back to the

other side of the membrane, the cathode electrode. In the catalyst on the cathode side, the electrons meet the protons and the oxygen that is fed to the cathode electrode. Then, the hydrogen protons together with the electrons combine with the oxygen, producing water,

$$O_2 + 4H^+ + 4e^- \rightarrow 2H_2O$$

In fact, the overall reaction that starts at the anode side and ends at the cathode side of the fuel cell stack is a combustion reaction since the hydrogen is combusted or "burnt" in a simple reaction:

$$2H_2 + O_2 \rightarrow 2H_2O$$

Typically, the voltage produced by one cell is from 0 to 1 volt, depending on the operating point and the size of the load connected to the fuel cell. To obtain a useful voltage many cells have to be connected in series. Such a collection of fuel cells in series is known as a *stack*. The total stack voltage is therefore the number of cells multiplied by the average cell voltage.

Auxiliary devices are required to ensure the proper operation of the fuel cell stack. Typical devices are a gas compressor, filters, valves, pumps, a refrigeration subsystem, humidifiers, flow controllers, pressure and temperature sensors, etc. Thus, the term *FCS system* will be used to refer to the whole system comprising the fuel cell stack and its auxiliary devices.

Fuel cell stack systems present high efficiency and do not create pollutants such as hydrocarbon or nitrogen oxide. Due to their properties, fuel cells may be attractive for several applications such as vehicle propulsion or remote power sources.

7.2 Review of Works on Diagnosis of FCS Systems

In the literature, the works devoted to FCS system diagnosis can be roughly divided in two groups. The first group involves works coming from the fuel cell community. In this group, some cumbersome techniques such as, those based on *gas chromatography*, *neutron imaging* or *magnetic resonance imaging* can be found. For further details on these techniques the reader is refereed to (Wu et al., 2008b).

Furthermore, in the fuel cell community, there also exist other ad-hoc techniques that use a specialised knowledge to perform diagnosis on FCS systems. Some examples of such techniques are the *polarisation curve*, the *current interruption* or the *electrochemical impedance spectroscopy*. The

polarisation curve method is a simple technique to investigate the actual behaviour of the fuel cell. It consists in plotting the fuel cell voltage against the current density under some operation conditions and then using the expert knowledge to determine how the fuel cell is behaving. The *current interruption* method is used to determine the voltage losses of the fuel cell by interrupting the current of the electric circuit. The *electrochemical impedance spectroscopy* deals with measuring the internal resistance of the fuel cell in a frequency sweep. In (Wu et al., 2008a) and (Barbir, 2005) all these techniques are described in detail.

The second group of works on FCS system diagnosis are developed from the automatic control perspective. The proposed methods are based on models and thus more general. A reduced number of papers devoted to model-based diagnosis for FCS system has been found. Next, the most outstanding works are presented.

In (Riascos et al., 2008), four faults related to several subsystems of the FCS system are diagnosed by using Bayesian networks. The faults concern the air-reactor blower, the refrigeration system, a fuel loss in the membrane (also known as fuel crossover) and the hydrogen pressure. In (Escobet et al., 2009), a set of relative residuals are designed to diagnose a set of fault scenarios. Residual design techniques are also used in (Ingimundarson et al., 2008), where two test quantities are developed to detect hydrogen leaks in the anode side. Finally, in (Yang et al., 2009), a set of structured residuals is obtained from a bond-graph model of a FCS system.

7.3 FCS System Benchmark

Next, a benchmark based on a FCS system is presented for diagnosis purposes. First, the analytical equations involving the fuel cell stack and its auxiliary equipment are detailed. Then, the model is built, the set of faults to be diagnosed are introduced, and finally causality is defined in order to obtain a causal structural model.

The model used here was first developed and implemented in Simulink by Pukrushpan in his thesis (Pukrushpan, 2003) and later further detailed in (Pukrushpan et al., 2004). The model is widely accepted in the control community as a good representation of the behaviour of a FCS system. The system model scheme is depicted in Figure 7.2. The cathode side consists of an air compressor to feed atmospheric air to the cathode, a supply manifold that connects the compressor output with the air cooler input, and an air cooler and a static humidifier, that respectively refrigerates and humid-

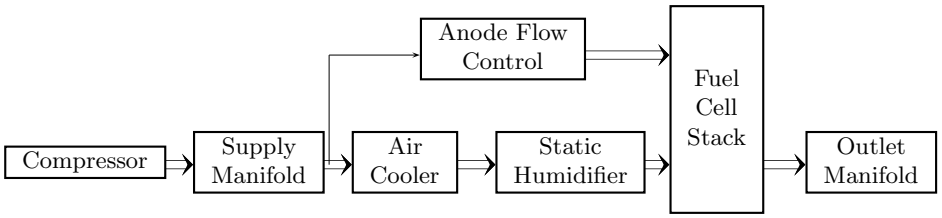


Figure 7.2: Fuel Cell Stack system.

ifies the air before entering the stack. The model guarantees the required stoichiometry by regulating the hydrogen, supplied from a pressurised or liquid hydrogen tank, by means of a controlled valve. The electrochemical principles of the fuel cell stack are also modelled in order to accurately evaluate the electricity production and the stack outputs. This implies specific model equations for the anode, the cathode, the membrane and the stack voltage. Finally, the cathode outlet manifold of the fuel cell is considered in the model as an external component.

The model only describes the normal operation mode. Hence purges in the anode side are not considered. This means that all the hydrogen in the anode side is consumed. It is also assumed that the temperature of the fuel cell is known and constant since its dynamic behaviour is much more slower than that of the rest of the model. These two assumptions reduce the complexity of the model by not considering a discrete behaviour nor thermodynamic equations for the stack model.

The model was originally developed for control purposes. So, it is necessary to first pinpoint which equations belong to each component. In order to do so, every component is modelled apart. This means that internal and external variables are considered apart for each component, and then extra equations will be defined to interconnect the different components. Following this procedure, the component behaviour can be easily modelled, as well as system faults defined. Note that, by doing this, the number of variables and equations involving the complete model is increased. However, the redundancy degree is preserved, meaning that no extra computing effort is expected. In fact, all the structural properties needed for diagnosis will remain unaltered.

The resulting FCS system model is a complex and large-scale model in-

volving 96 equations and 96 unknown variables. Next, the analytical equations of each subsystem are detailed and later in Section 7.3.4 the causal structure of the model will be obtained.

7.3.1 FCS System Model

Variable Nomenclature

For a better understanding, the meaning of the system variables is shown in table 7.1.

Variable	Description	Units
v	Voltage	V
ω_c	Angular speed	rad/s
τ	Torque	$N \cdot m$
i	Electrical current	A
ϕ	Relative humidity	
W	Mass flow	kg/s
T	Temperature	K
p	Pressure	Pa, atm, Bar
η	Efficiency	
y	Mole fraction	
m	Mass	kg
$\dot{\lambda}$	Water content	

Table 7.1: Variable nomenclature.

Variable subscripts will be used to indicate the component a variable is related to. Table 7.2 shows the subscripts denoting the system components, as well as some other useful atributes. It is also important to point out that given a variable x, in a differential equation, its derivative with respect to time will be denoted by $\dot{x}$.

Basic Principles

First, some equations concerning basic physic principles are detailed. From the ideal gas law, the pressure and the mass of gas are related according to

$$pV = mRT \tag{7.1}$$

Subscript	Meaning
cp	Compressor
sm	Supply manifold
ac	Air cooler
sh	Static humidifier
afc	Anode flow control
st	Fuel cell stack
an	Stack anode
ca	Stack cathode
m	Stack membrane
om	Outlet manifold
H_2	Hydrogen
O_2	Oxygen
v	Vapour
atm	Atmospheric

Table 7.2: Variable subscription.

where p is the gas pressure, V is the volume, m is the mass of the gas, T is the gas temperature and R is the gas constant.

The equation used to calculate the saturation pressure, at a given temperature from 0°C to 100 °C, is

$$log_{10}(p_{sat}) = a_4T^4 + a_3T^3 + a_2T^2 + a_1T + a_0 \tag{7.2}$$

where a_i, for $i = \{0, 1, 2, 3, 4\}$, are constant coefficients. Since this equation is frequently used in the model description, $p_{sat}(T)$ will be used instead of the equation in (7.2), indicating that saturation pressure only depends on temperature T.

The thermodynamic properties of a gas mixture are used to represent gases of different species. Specifically, when the mixture involves water vapour, the relative humidity is given by

$$\phi_2 = \frac{p_2 p_{sat}(T_1)\phi_1}{p_1 p_{sat}(T_2)} \tag{7.3}$$

Air Compressor

The air compressor is decomposed into two main parts. One part concerns the electric motor, whereas the other part concerns the compressor box.

The compressor motor is modelled using a direct current electric motor model. The first equation describes the electric part of the model, where the voltage applied to the compressor v_{cp} is transformed into an angular speed, ω_{cp}, through a ratio parameter k_v, and electrical losses through the resistance parameter R_{cp} and the compressor current i_{cp}:

$$v_{cp} = k_v \omega_{cp} + R_{cp} i_{cp} \tag{7.4}$$

The real motor torque $\tau_{m_{cp}}$ is calculated by subtracting the friction torque from the theoretical motor torque. The friction torque is calculated by means of a friction coefficient B, whereas the theoretical torque is calculated by means of the torque coefficient k_t:

$$\tau_{m_{cp}} = k_t \cdot i_{cp} - B \cdot \omega_{cp} \tag{7.5}$$

The variation of the angular speed is computed by the difference between the motor and the compressor torques,

$$J_{cp} \dot{\omega}_{cp} = \tau_{m_{cp}} - \tau_{com_{cp}} \tag{7.6}$$

where J_{cp} is the inertia coefficient of both compressor and motor.

A compressor flow map is used to determine the air flow rate, W_{cp}, supplied by the compressor. This is done by means of a nonlinear curve fitting method named Jensen & Kristensen (Moraal and Kolmanovsky, 1999). First, the input values (temperature, angular speed and pressure) are scaled:

$$T_{cp,cor} = T_{atm}/288 \tag{7.7}$$

$$\omega_{cp,cor} = \frac{\omega_{cp}}{\sqrt{T_{cp,cor}}} \tag{7.8}$$

$$p_{ratio} = \begin{cases} 0 \text{ if } \frac{10^{-5} p_{cp,out}}{1{,}01325 p_{atm}} \leq 0 \\ \frac{10^{-5} p_{cp,out}}{1{,}01325 p_{atm}} \text{ if } 0 < \frac{10^{-5} p_{cp,out}}{1{,}01325 p_{atm}} < 1000 \\ 1000 \text{ if } \frac{10^{-5} p_{cp,out}}{1{,}01325 p_{atm}} \geq 1000 \end{cases} \tag{7.9}$$

Then, the normalized compressor flow rate, Φ, is calculated as

$$\Phi = \Phi_{max}(M)(1 - e^{\beta(M)(\frac{\Psi}{\Psi_{max}(M)} - 1)}) \tag{7.10}$$

where the parameters $\Phi_{max}(M)$, $\beta(M)$ and $\Psi_{max}(M)$ are polynomial functions of the Match number M, defined by

$$M = \frac{U_c}{\sqrt{\gamma R_a T_{atm}}} \tag{7.11}$$

R_a is the gas constant, γ is the specific heat ratio of the air and U_c is the compressor blade tip speed calculated by

$$U_c = \frac{\pi}{60} d_c \omega_{cp,cor} \tag{7.12}$$

with d_c the compressor diameter outlet. Parameter Ψ in (7.10) is calculated using the following expression,

$$\Psi = \frac{C_p T_{atm} (p_{ratio}^{\frac{\gamma-1}{\gamma}} - 1)}{\frac{1}{2} U_c^2} \tag{7.13}$$

The corrected air flow rate, $W_{cor,cp}$ is calculated from the normalized compressor flow rate as

$$W_{cor,cp} = \Phi \rho_a \frac{\pi}{4} d_c^2 U_c \tag{7.14}$$

where ρ_a is the air density. Finally, the output air flow rate can be computed as

$$W_{cp,out} = W_{cp,cor} \frac{p_{atm}}{\sqrt{T_{cp,cor}}} \tag{7.15}$$

The compressor efficiency, η_{cp}, is determined by means of a look-up table from the corrected air flow rate and the corrected air pressure. This relation is denoted as

$$\eta_{cp} = \text{LookupTable}(W_{cp,cor}, p_{ratio}) \tag{7.16}$$

The air temperature at the compressor exit is calculated as

$$T_{cp,out} = T_{atm} - \frac{T_{atm}}{\eta_{cp}} (p_{ratio}^{\frac{\gamma-1}{\gamma}} - 1) \tag{7.17}$$

and the applied torque to drive the compressor is determined using the following thermodynamic equation

$$\tau_{com_{cp}} = C_p \frac{T_{atm}}{\eta_{cp}} (p_{ratio}^{\frac{\gamma-1}{\gamma}} - 1) W_{cp,out} \tag{7.18}$$

where C_p is the specific heat capacity of the air.

Equation (7.3) is used to calculate the relative humidity of the compressor output gas:

$$\phi_{cp} = \frac{p_{cp,out} p_{sat}(T_{atm}) \phi_{atm}}{p_{atm} p_{sat}(T_{cp,out})} \tag{7.19}$$

Supply Manifold

Manifolds are modelled as a lumped volume in pipes or connections between different devices. The inlet supply manifold model describes the behavior of the pipe connecting the compressor output with the air cooler input. To calculate the outlet air flow, a linearised model of a nozzle is used,

$$W_{sm,out} = k_{sm,out}(p_{sm,out} - p_{sm,ds}) \tag{7.20}$$

where $k_{sm,out}$ is the supply manifold outlet flow constant. The mass conservation principle is used in a flow balance, therefore it must hold that

$$\dot{m}_{sm} = W_{sm,in} - W_{sm,out} \tag{7.21}$$

The air temperature is expected to decrease in the supply manifold. Therefore, the following pressure dynamic equation is used

$$\dot{p}_{sm,out} = \frac{\gamma R_a}{V_{sm}}(W_{sm,in}T_{sm,in} - W_{sm,out}T_{sm,out}) \tag{7.22}$$

where V is the supply manifold volume and $T_{sm,out}$ is the output temperature of the air flow through the manifold, which is calculated using the ideal gas law

$$T_{sm,out} = \frac{V_{sm}p_{sm,out}}{R_a m_{sm}} \tag{7.23}$$

As before, equation (7.3) is used to calculate the relative humidity, $\phi_{sm,out}$, of the supply manifold output gas:

$$\phi_{sm,out} = \frac{p_{sm,out}p_{sat}(T_{sm,in})\phi_{sm,in}}{p_{sm,in}p_{sat}(T_{sm,out})} \tag{7.24}$$

Air Cooler

To prevent any damage in the stack membrane, the cathode inlet air has to be cooled down before entering the stack. As it was mentioned, this model does not describe the heat transfer effect. Hence an ideal static air cooler model is proposed. The outlet air temperature of the air cooler, $T_{ac,out}$ is set to a desired value,

$$T_{ac,out} = T_{des} \tag{7.25}$$

and only the effect of the humidity change is taken into account, according to (7.3):

$$\phi_{ac,out} = \frac{p_{ac,out}p_{sat}(T_{ac,in})\phi_{ac,in}}{p_{ac,in}p_{sat}(T_{ac,out})} \tag{7.26}$$

Both, flow and pressure, do not change inside the cooler, so:

$$W_{ac,out} = W_{ac,in} \tag{7.27}$$

$$p_{ac,out} = p_{ac,in} \tag{7.28}$$

Static Humidifier

To avoid hydration problems in the membrane, the air also needs to be properly humidified. This task is done by a static humidifier which sets a desired level of humidity in the air. A static model to describe the humidifier behavior is used. Here, it is assumed that the air temperature does not change inside the humidifier:

$$T_{sh,out} = T_{sh,in} \tag{7.29}$$

However, since an extra amount of vapour has to be added to the air, the air flow and pressure change. First, the vapour pressure of the inlet gas, $p_{v,sh,in}$, is calculated as

$$p_{v,sh,in} = \phi_{sh,in} p_{sat}(T_{sh,in}) \tag{7.30}$$

where $p_{sat}(T_{sh,in})$ is the saturation pressure of inlet gas calculated according to (7.2). Using the gas mixture the properties, dry air pressure, $p_{a,sh,in}$, is calculated by

$$p_{a,sh,in} = p_{sh,in} - p_{v,sh,in} \tag{7.31}$$

and the humidity ratio is determined as

$$w_{sh,in} = \frac{M_v}{M_a} \frac{p_{v,sh,in}}{p_{a,sh,in}} \tag{7.32}$$

where M_v is the water molar mass and M_a is the atmospheric air molar mass. Now, the mass flow rates $W_{a,sh,in}$ and $W_{v,sh,in}$, both from dry air and vapour, can be calculated:

$$W_{a,sh,in} = \frac{1}{1 + w_{sh,in}} W_{sh,in} \tag{7.33}$$

$$W_{v,sh,in} = W_{sh,in} - W_{a,sh,in} \tag{7.34}$$

Note that there is no change in the dry flow rate in the humidifier, so

$$W_{a,sh,out} = W_{a,sh,in} \tag{7.35}$$

Furthermore, the outlet vapour pressure of the humidifier can be calculated from the outlet air humidity, $\phi_{sh,out}$, together with the saturation pressure,

$$p_{v,sh,out} = \phi_{sh,out} p_{sat}(T_{sh,out}) \tag{7.36}$$

Remark that $p_{sat}(T_{sh,out})$ is the output gas saturation pressure calculated from (7.2), using $T_{sh,out}$. The humidity of the outlet air flow is the desired humidity set by the humidifier,

$$\phi_{sh,out} = \phi_{des} \tag{7.37}$$

and the dry air pressure does not change through the humidifier

$$p_{a,sh,out} = p_{a,sh,in} \tag{7.38}$$

So the outlet vapour flow, $W_{v,sh,out}$, is computed as

$$W_{v,sh,out} = \frac{p_{v,sh,out}}{p_{a,sh,out}} \frac{M_v}{M_a} W_{a,sh,out} \tag{7.39}$$

The amount of water flow needed to be injected in the air to achieve the desired level of humidity can be determined from the inlet and outlet vapour flows,

$$W_{v,inj} = W_{v,sh,out} - W_{v,sh,in} \tag{7.40}$$

Finally, the total outlet air flow rate $W_{sh,out}$ and the outlet air pressure $p_{sh,out}$ are determined by

$$W_{sh,out} = W_{sh,in} + W_{v,inj} \tag{7.41}$$

$$p_{sh,out} = p_{a,sh,out} + p_{v,sh,out} \tag{7.42}$$

Anode Flow Control

The hydrogen supplied to the anode is regulated by a proportional controller. The controller takes the differential pressure between anode and cathode to compute the regulated hydrogen flow, $W_{afc,out}$,

$$W_{afc,out} = K_1(K_2 \cdot p_{afc,+} - p_{afc,-}) \tag{7.43}$$

It is assumed that this control law is instantaneous, therefore no dynamic effect is considered. This is a feasible assumption since the hydrogen comes from a high pressure tank and the control valve has a fast response.

Outlet Manifold

A similar equation introduced for the supply manifold is now used to describe the outlet manifold. Here, the calculated pressure will be used in the cathode to determine the flow rate. Contrary to the supply manifold, the pressure drop is large since the output is the atmospheric pressure whereas the change of air temperature is negligible. So

$$\dot{p}_{om,out} = T_{om,in}\frac{R_a}{V_{om}}(W_{om,in} - W_{om,out}) \tag{7.44}$$

where V_{om} is the outlet manifold volume. The main difference with respect to the supply manifold is an equation that describes the behavior of the nozzle (throttle). Here a non-linear equation is used instead,

$$W_{om,out} = \text{NonlinearNozzle}(p_{om,out}, p_{om,ds}, T_{om,in}) \tag{7.45}$$

The air temperature and humidity do not change throughout the manifold, so:

$$T_{om,out} = T_{om,in} \tag{7.46}$$

$$\phi_{om,out} = \phi_{om,in} \tag{7.47}$$

Fuel Cell Stack

Cell anode:

Note first that, since purge operation is not considered in the model, the anode exit flow rate, $W_{an,out}$, is assumed null:

$$W_{an,out} = 0 \tag{7.48}$$

On the other hand, the hydrogen flow rate inside the anode is balanced regarding the hydrogen input flow,$W_{H_2,in}$, the hydrogen flow output, $W_{H_2,out}$, and the reacted hydrogen, $W_{H_2,reacted}$,

$$\dot{m}_{H_2} = W_{H_2,in} - W_{H_2,out} - W_{H_2,reacted} \tag{7.49}$$

The consumed hydrogen rate depends on the stack current,

$$W_{H_2,reacted} = M_{H_2}\frac{n \cdot i_{st}}{2F} \tag{7.50}$$

where M_{H_2} is the hydrogen molar mass, n is the number of cells and F is the Faraday's constant.

The hydrogen and vapour mass in the anode can be related with the hydrogen partial and the vapour partial pressure respectively by means of the ideal gas law (7.1):

$$m_i = \frac{p_i V_{an}}{R_i T_{st}} \quad i \in \{H_2, (v, an)\} \tag{7.51}$$

where V_{an} is the anode volume, R_{H_2} is the hydrogen gas constant and $R_{v,an}$ is the vapour gas constant. Note that the vapour mass in the hydrogen is bounded by the maximum amount of water that the gas can contain, $m_{v,max,an}$, calculated as

$$m_{v,max,an} = \frac{p_{sat}(T_{st}) V_{an}}{R_{v,an} T_{st}} \tag{7.52}$$

Therefore, if $m_{v,an} > m_{v,max,an}$ then $m_{v,an} = m_{v,max,an}$.

The inlet and outlet hydrogen flow, $W_{H_2,in}$ and $W_{H_2,out}$, are calculated from the inlet and outlet gas pressures and humidities, respectively. First, the gas pressures are calculated as

$$p_{v,an,i} = \phi_{an,i} p_{sat}(T_{st}) \quad i \in \{in, out\} \tag{7.53}$$

$$p_{H_2,i} = p_{an,i} - p_{v,an,i} \quad i \in \{in, out\} \tag{7.54}$$

and the humidity ratio as

$$w_{an,i} = \frac{M_v}{M_{H_2}} \frac{p_{v,an,i}}{p_{H_2,i}} \quad i \in \{in, out\} \tag{7.55}$$

Finally the hydrogen flow ratio can be calculated by

$$W_{H_2,i} = \frac{1}{1 + w_{an,i}} W_{an,i} \quad i \in \{in, out\} \tag{7.56}$$

A similar flow rate balancing is done for the anode humidity. It is assumed that there is no liquid water inside the anode volume. Hence, there is no water flow leaving the anode. This implies that $m_{v,an} \in [0, m_{v,max,an}]$.

$$\dot{m}_{v,an} = W_{v,an,in} - W_{v,an,out} - W_{v,membr} \tag{7.57}$$

Water flow across the membrane will be determined in the membrane hydration model. Both, the vapour input and output flows, are calculated by subtracting the hydrogen flow from the total gas flow,

$$W_{v,an,i} = W_{an,i} - W_{H_2,i} \quad i \in \{in, out\} \tag{7.58}$$

Once all the partial pressures of each gas in the anode have been determined, the output anode pressure can be calculated as

$$p_{an,out} = p_{H_2} + p_{v,an} \tag{7.59}$$

The relative humidity of the anode outlet gas is the ratio between vapour pressure and saturation pressure, as long as the vapour mass is under the maximum. Otherwise, the outlet gas is fully humidified and the relative humidity is set to 1:

$$\phi_{an,out} = \begin{cases} \frac{p_{v,an}}{P_{sat}(T_{st})} & \text{if } m_{v,an} \leq m_{v,max,an} \\ 1 & \text{if } m_{v,an} > m_{v,max,an} \end{cases} \tag{7.60}$$

Cell membrane:

The membrane model describes the water flow across the membrane, $W_{v,membr}$, and the membrane water content, $\dot{\lambda}$. The water content is a function of the average water activity between the anode and cathode water activities. It turns out that water activity is equivalent, for gases, to relative humidity, ϕ. Next, a generic equation for membrane water content is used to represent such relation. For a detailed information and a parameter description the reader is refereed to Pukrushpan (2003).

$$\dot{\lambda} = \text{WaterContentFunction}(\phi_{an,out}, \phi_{ca,out}) \tag{7.61}$$

The total stack water flow rate across the membrane, $W_{v,membr}$ can be computed as

$$W_{v,membr} = N_{v,membr} \cdot M_v \cdot A_{fc} \cdot n \tag{7.62}$$

where $N_{v,membr}$ is the water molar flow rate per unit area and per cell, and A_{fc} is the fuel cell active area.

The water molar flow rate is obtained through the following flow balance:

$$N_{v,membr} = N_{v,osmotic} - N_{v,diff} \tag{7.63}$$

The net anode to cathode water flow caused by electro-osmotic drag is calculated as

$$N_{v,osmotic} = n_d(\dot{\lambda}) \frac{i_{st}}{A_{fc}F} \tag{7.64}$$

where $n_d(\dot{\lambda})$ is the electro-osmotic drag coefficient which depends on the membrane water content. The net cathode to anode water flow caused by back diffusion is calculated as

$$N_{v,diff} = D_w(\dot{\lambda}, T_{st}) \frac{c_{v,ca}(\phi_{ca,out}) - c_{v,an}(\phi_{an,out})}{t_m} \tag{7.65}$$

where t_m is the membrane thickness and $D_w(\lambda, T_{st})$ is the diffusion coefficient calculated from the membrane water content and the stack temperature, T_{st}. Water concentrations at the membrane surface on the anode and cathode sides, $c_{v,ca}(\phi_{ca,out})$ and $c_{v,an}(\phi_{an,out})$, depend on the membrane water content in the cathode and the anode, respectively. Thus, they depend on the corresponding cathode or anode humidity.

Cell cathode:

The cathode flow model is similar to the anode flow model. Here, the mixed nature of atmospheric air is taken into account. Thus, mass flow balancing involving oxygen, nitrogen and vapour will be performed. First, cathode vapour mass flow balance is considered in the following equation,

$$\dot{m}_{v,ca} = W_{v,ca,in} - W_{v,ca,out} + W_{v,membr} + W_{v,ca,gen} \tag{7.66}$$

with $m_{v,ca} \in [0, m_{v,max,ca}]$.

The vapour mass flow rate generated in the fuel cell is obtained from the stack current

$$W_{v,ca,gen} = M_v \frac{n \cdot i_{st}}{2F} \tag{7.67}$$

The cathode inlet vapour mass flow can be determined from the total cathode inlet flow by

$$W_{v,ca,in} = W_{ca,in}(1 - \frac{1}{1 + w_{ca,in}}) \tag{7.68}$$

Cathode humidity ratios are calculated as for the anode side case:

$$w_{ca,i} = \frac{M_v}{M_{a,i}} \frac{\phi_{ca,i} p_{sat}(T_{st})}{p_{ca,i} - \phi_{ca,i} p_{sat}(T_{st})} \quad i \in \{in, out\} \tag{7.69}$$

where $M_{a,in}$ is the inlet air molar mass regarding the standard mixture of 0.21 of oxygen and $(1 - 0.21)$ of nitrogen and $M_{a,out}$ is the outlet air molar mass. Note that, since oxygen is consumed in the reaction, the inlet air molar mass is constant whereas the outlet air molar mass depends on the oxygen mole fraction, $y_{O_2,ca}$, $M_{a,out}(y_{O_2,ca})$.

The oxygen mole fraction, $y_{O_2,ca}$ is also needed to calculate the outlet flows,

$$y_{O_2,ca} = \frac{p_{O_2}}{p_{O_2} + p_{N_2}} \tag{7.70}$$

Similarly, the cathode inlet oxygen and nitrogen mass flows are determined from the cathode inlet humidity ratio as follows:

$$W_{O_2,in} = x_{O_2,in} W_{ca,in} \frac{1}{1 + w_{ca,in}} \tag{7.71}$$

$$W_{N_2,in} = (1 - x_{O_2,in})W_{ca,in}\frac{1}{1 + w_{ca,in}} \tag{7.72}$$

where $x_{O_2,in}$ is a constant parameter that depends on the input oxygen mass fraction. The mass of each gas in the cathode side is related to the corresponding partial gas pressure by the ideal gas law:

$$m_i = \frac{p_i V_{ca}}{R_i T_{st}} \quad i \in \{O_2, N_2, (v, ca)\} \tag{7.73}$$

where V_{ca} is the cathode volume, R_{O_2} is the oxygen gas constant, R_{N_2} is the nitrogen gas constant and $R_{v,ca}$ is the vapour gas constant. The cathode vapour mass is upper bounded by the maximum value,

$$m_{v,max,ca} = \frac{p_{sat}(T_{st})V_{ca}}{R_{v,ca}T_{st}} \tag{7.74}$$

Therefore, if $m_{v,ca} > m_{v,max,ca}$ then $m_{v,ca} = m_{v,max,ca}$. Now, the oxygen flow rate balancing is performed:

$$\dot{m}_{O_2} = W_{O_2,in} - W_{O_2,out} - W_{O_2,reacted} \tag{7.75}$$

where the oxygen mass flow depends on the stack current

$$W_{O_2,reacted} = M_{O_2}\frac{n \cdot i_{st}}{4F} \tag{7.76}$$

being M_{O_2} the oxygen molar mass.

The last gas flow balancing, corresponding to the nitrogen mass flow rate, is calculated as

$$\dot{m}_{N_2} = W_{N_2,in} - W_{N_2,out} \tag{7.77}$$

The vapour, oxygen and nitrogen outlet mass flows are respectively calculated in the following three equations

$$W_{v,ca,out} = W_{ca,out}(1 - \frac{1}{1 + w_{ca,out}}) \tag{7.78}$$

$$W_{O_2,out} = x_{O_2,out}(y_{O_2,ca})W_{ca,out}\frac{1}{1 + w_{ca,out}} \tag{7.79}$$

$$W_{N_2,out} = (1 - x_{O_2,out}(y_{O_2,ca}))W_{ca,out}\frac{1}{1 + w_{ca,out}} \tag{7.80}$$

where $x_{O_2,out}$ is a parameter which depends on the outlet molar fraction, $y_{O_2,ca}$.

The total cathode outlet flow rate is calculated by means of a simplified orifice model:

$$W_{ca,out} = k_{ca,out}(p_{ca,out} - p_{st,ds}) \tag{7.81}$$

where $k_{ca,out}$ is the orifice constant and $p_{st,ds}$ is the stack downstream pressure. The total cathode pressure is obtained using the mixed gas properties,

$$p_{ca,out} = p_{O_2} + p_{N_2} + p_{v,ca} \tag{7.82}$$

Finally, the cathode humidity is calculated from the vapour partial pressure and the saturation pressure

$$\phi_{ca,out} = \begin{cases} \frac{p_{v,ca}}{P_{sat}(T_{st})} & \text{if } m_{v,ca} \leq m_{v,max,ca} \\ 1 & \text{if } m_{v,ca} > m_{v,max,ca} \end{cases} \tag{7.83}$$

Stack voltage:

The stack voltage, v_{st}, is obtained by multiplying a single cell voltage by the number of cells, n. To calculate the cell voltage, the cell open circuit voltage, E, is first considered and then the voltage losses are subtracted from it. The losses considered in this model are the activation loss ,v_{act}, the ohmic loss, v_{ohm}, and the concentration loss, v_{conc}. Therefore, the main equation to compute the stack voltage is

$$v_{st} = n(E - v_{act} - v_{ohm} - v_{conc}) \tag{7.84}$$

It is assumed that the open circuit voltage depends on the stack temperature and both hydrogen and oxygen pressures,

$$E = \text{OpenCircuitVoltage}(T_{st}, p_{N_2}, p_{O_2}) \tag{7.85}$$

The activation loss voltage is determined by the following equation,

$$v_{act} = v_o + v_a(1 - e^{-c_1 \frac{i_{st}}{A_{fc}}}) \tag{7.86}$$

where c_i is a constant, v_o depends on the stack temperature, the saturation pressure and the cathode pressure, and v_a depends on the stack temperature, the saturation pressure and the oxygen partial pressure. These dependencies are determined by means of a nonlinear regression. Thus, they are generically represented in the following two equations

$$v_o = \text{NonlinearRegression1}(T_{st}, p_{sat}(T_{st}), p_{ca,out}) \tag{7.87}$$

$$v_a = \text{NonlinearRegression2}(T_{st}, p_{sat}(T_{st}), p_{O_2}) \tag{7.88}$$

The dropped ohmic voltage due to the membrane resistance is calculated by

$$v_{ohm} = \frac{i_{st}}{A_{fc}} R_{ohm}(\dot{\lambda}, T_{st}) \tag{7.89}$$

where $R_{ohm}(\dot{\lambda}, T_{st})$ is the internal electrical resistance, which strongly depends on the membrane water content, $\dot{\lambda}$, and the cell temperature, T_{st}. The concentration losses can be obtained from

$$v_{conc} = i_{st}(c_2(T_{st}, p_{O_2}, p_{sat}(T_{st}))\frac{i_{st}}{i_{max}})^{c_3} \tag{7.90}$$

where $c_2(T_{st}, p_{O_2}, p_{sat}(T_{st}))$ is a parameter that depends on the stack temperature, the oxygen pressure and the saturation pressure, while the c_3 and i_{max} are constant parameters.

7.3.2 Model Equations

In the previous subsection, the equations describing the basic physical principles have been described. In this subsection, the complete model involving up to 96 equations is presented. Three different kinds of equations are distinguished: *component equations*, *known variable equations* and *component interconnection equations*.

Component equations refer to the equations that model the FCS system components. These are obtained applying a model reduction procedure to the equations introduced in the previous subsection. *Known variables equations* are introduced in the model to indicate that some model variables are assumed known. *Component interconnection equations* describe the interconnections among components.

Component Equations

According to (Krysander and Nyberg, 2002), given an overdetermined structural model M, if there exists a subset $X' \subseteq X$ of unknown variables such that

$$1 + |X'| = |\text{equ}_M(X')| \tag{7.91}$$

then the set of equations $\text{equ}_M(X')$ can be combined into one equation, and the variables X' can be removed from the original model. By doing this, the structural model is reduced while no diagnosis structural property is missed since it is proved that all the equations in $\text{equ}_M(X')$ will appear

together in an MSO set. Furthermore, in (Krysander et al., 2008), it is shown that such set of equations can be found by means of the equivalent classes introduced in (2.33). In fact, every equivalent class that involves more than one equation fulfils expression (7.91). Therefore, it can be lumped into one single equation.

However, it is important to note that if some variables in X' can be measured (i.e. they involve candidate sensors), then, since X' are removed during the model reduction, the initial candidate sensor set might also be reduced, leading to a loss of performance in the sensor placement problem. To prevent this, it is required that all candidate measurable variables (see Section 7.4 for candidate sensors in the FCS system) will be omitted from the structural model when model reduction is performed. Hence, no measurable variable will be removed and the initial candidate sensor set is preserved.

The model reduction procedure is separately performed for each component in the FCS system model. For each lumped equation set, it is checked whether the removed variables can be causally computed (see Chapter 6). If the removed variables are not causally computable then the corresponding set of equations is not lumped. The result is depicted in Table 7.3 where an equivalence between the new and the original equations is given. Due to the model reduction, some original variables are missing and some equations are lumped, for this reason the variable nomenclature is preserved, whereas all equations are renamed.

Known Variable Equations

Some model variables are assumed known. This is the case of atmospheric variables such as humidity (ϕ_{atm}), temperature (T_{atm}) and pressure (p_{atm}). The outlet manifold releases the gas to the atmosphere. Therefore, the downstream pressure ($p_{ds,om}$) is also known. A similar assumption is done for the inlet anode hydrogen humidity ($\phi_{an,in}$) since the hydrogen comes from a pressurised tank where humidity is known. The desired air temperature (T_{des}) and the desired air humidity (ϕ_{des}) are setpoints and therefore regarded as known variables. Furthermore, the original model has an external controller which controls the compressor voltage (v_{cp}) by means of the stack current (i_{st}), consequently these two variables are also known.

The following equations state that these variables are known, so they have been added to the model:

$$
\begin{array}{lll}
e_{62} : \phi_{atm} = 0.5 & e_{65} : p_{om,ds} = 1atm & e_{68} : \phi_{des} = 0.2 \\
e_{63} : T_{atm} = 298K & e_{66} : \phi_{an,in} = 0.4 & e_{95} : i_{st} = i_{st}^{measured} \\
e_{64} : p_{atm} = 1atm & e_{67} : T_{des} = 353K & e_{96} : v_{cp} = v_{cp}^{controlled}
\end{array}
$$

New equ.	Original equations
e_1	(7.4)
e_2	(7.5)
e_3	(7.6)(7.18)
e_4	(7.7)
e_5	(7.9)
e_6	(7.8)(7.10-7.14)
e_7	(7.15)
e_8	(7.16)
e_9	(7.17)
e_{10}	(7.19)
e_{11}	(7.20)
e_{12}	(7.21)
e_{13}	(7.22)
e_{14}	(7.23)
e_{15}	(7.24)
e_{16}	(7.25)
e_{17}	(7.26)
e_{18}	(7.27)
e_{19}	(7.28)
e_{20}	(7.29)

New equ.	Original equations
e_{21}	(7.30)
e_{22}	(7.31)
e_{23}	(7.36)
e_{24}	(7.37)
e_{25}	(7.38)
e_{26}	(7.32-7.35)(7.39-7.40)
e_{27}	(7.41)
e_{28}	(7.42)
e_{29}	(7.43)
e_{30}	(7.44)
e_{31}	(7.45)
e_{31}	(7.46)
e_{33}	(7.47)
e_{34}	(7.2) for $T = T_{st}$
e_{35}	(7.49-7.50)
e_{36}	(7.51)(7.59) for $i = H_2$
e_{37}	(7.51) for $i = v, an$
e_{38}	(7.52)
e_{39}	(7.53-7.56) for $i = in$
e_{40}	(7.53-7.56) for $i = out$

New equ.	Original equations
e_{41}	(7.57)(7.58) for $i = in$
e_{42}	(7.58) for $i = out$
e_{43}	(7.60)
e_{44}	(7.48)
e_{45}	(7.61)
e_{46}	(7.62-7.63)
e_{47}	(7.66-7.68)
e_{48}	(7.69) for $i = in$
e_{49}	(7.69) for $i = out$
e_{50}	(7.73) for $i = O_2$
e_{51}	(7.71)(7.75)(7.76)(7.79)
e_{52}	(7.73) for $i = N_2$
e_{53}	(7.72)(7.77)(7.80)
e_{54}	(7.73) for $i = v, ca$
e_{55}	(7.74)
e_{56}	(7.70)
e_{57}	(7.78)
e_{58}	(7.81)
e_{59}	(7.82)
e_{60}	(7.83)
e_{61}	(7.84-7.90)

Table 7.3: Reduced equation of the FCS model.

It is important to note that by considering the above mentioned variables as known, the resulting model is a well-determined set of equations where all the remaining unknown variables can be computed. This is reasonable since the model can be simulated (i.e. all the internal variables can be computed by the simulation engine).

Component Interconnection Equations

Once every component is modelled, those components that have common external variables should be connected. This is accomplished by imposing that an external variable of a component equals to another external variable of a different component. Therefore, extra equations need to be added to the model.

For instance, consider the compressor output flow, $W_{cp,out}$ which, in the original model, is also the supply manifold input flow since all the compressed air flows to the supply manifold. An extra variable, $W_{sm,in}$, for the supply manifold input flow has been defined, and the following equation has been added to indicate that the compressor is connected with the supply

manifold:

$$e_{69} : W_{sm,in} = W_{cp,out} \tag{7.92}$$

The following equations are included to the model to interconnect system components:

$e_{70} : T_{sm,in} = T_{cp,out}$ $e_{78} : p_{ca,out} = p_{sm,ds}$ $e_{86} : W_{om,in} = W_{ca,out}$
$e_{71} : p_{sm,in} = p_{cp,out}$ $e_{79} : p_{afc,+} = p_{sm,out}$ $e_{87} : T_{om,in} = T_{st}$
$e_{72} : \phi_{sm,in} = \phi_{cp}$ $e_{80} : p_{afc,-} = p_{an,out}$ $e_{88} : \phi_{om,in} = \phi_{ca,out}$
$e_{73} : p_{sm,out} = p_{cp,out}$ $e_{81} : W_{an,in} = W_{afc,out}$ $e_{89} : p_{an,in,in} = p_{afc,-}$
$e_{74} : W_{ac,in} = W_{sm,out}$ $e_{82} : W_{sh,in} = W_{ac,out}$ $e_{90} : W_{ca,in} = W_{sh,out}$
$e_{75} : T_{ac,in} = T_{sm,out}$ $e_{83} : T_{sh,in} = T_{ac,out}$ $e_{91} : T_{st} = T_{sh,out}$
$e_{76} : p_{ac,in} = p_{sm,out}$ $e_{84} : p_{sh,in} = p_{ac,out}$ $e_{92} : p_{ca,in} = p_{sh,out}$
$e_{77} : \phi_{ac,in} = \phi_{sm,out}$ $e_{85} : \phi_{sh,in} = \phi_{ac,out}$ $e_{93} : \phi_{ca,in} = \phi_{sh,out}$
$e_{94} : p_{st,ds} = p_{om,ds}$

7.3.3 FCS Process Faults

Seven process faults have been defined in the FCS system. The faults are defined such that either a parameter or a computed value from an unknown variable is modified in the simulation model. As a result, the fault only affects one equation, i.e. the fault equation.

There are two compressor faults, f_{cp1} and f_{cp2}. Fault f_{cp1} represents an electric fault where the electrical resistance varies (e.g. due to an overheating). Specifically, it affects parameter R_{cp} in (7.4). Fault f_{cp2} represents a malfunction of the compressor box. It is simulated by modifying $W_{cor,cp}$, computed from the Jensen & Kristensen compressor map in (7.14). The supply manifold is affected by fault f_{sm} which represents, for example, a leak. It is simulated by a variation of the output air flow calculated in (7.20). Air cooler and static humidifier faults are represented, respectively, by f_{ac} and f_{sh}. These two faults are simulated by a change in the setpoints values, T_{des} and ϕ_{des}, meaning that the device is not working properly. Next fault, f_{st}, affects the fuel cell stack. It represents a malfunction in the outlet cathode (e.g. the outlet is partially stuck). Specifically, it is simulated by means of a variation of the outlet air flow $W_{ca,out}$ in (7.81). Last fault f_{om} affects the outlet manifold by modifying $W_{om,out}$ in the non-linear nozzle expression (7.45). It could represent either a leak or an outlet obstruction. Table 7.4 shows the correspondence between the faults and their fault equations of the model (see Table 7.3).

Faults f_{cp1}, f_{cp2}, f_{sm}, f_{st} and f_{om} are simulated as multiplicative faults where the corresponding parameters or computed variables are proportionally altered, e.g. $R_{cp}^{faulty} = 0.9R_{cp}$. On the other hand, faults f_{ac} and f_{sh} are

Fault	Fault equation	Fault description
f_{cp1}	e_1	compressor motor fault
f_{cp2}	e_6	compressor box fault
f_{sm}	e_{11}	supply manifold fault
f_{ac}	e_{16}	air cooler fault
f_{sh}	e_{24}	static humidifier fault
f_{om}	e_{31}	outlet manifold fault
f_{st}	e_{58}	stack cathode fault

Table 7.4: Correspondence between faults and equations.

simulated as additive faults, where an offset is added to their corresponding setpoints, e.g. $T_{des}^{faulty} = T_{des} + 0.4K$.

7.3.4 Causal Structural Model

The causality assignment is defined on the set of edges of the structural model. The resulting structural models are depicted in Appendix section 7.A.

Due to the non-linear and dynamic nature of the FCS model, the causally computable approach, introduced in Chapter 6, is used for this particular application. It is worth recalling, that other diagnosis approaches could also be implemented for this application, although this would imply to deal with non-linear equations using numerical solvers, for instance.

Next, the causal characterisation of the structural model is introduced. Causality has been applied by studying whether each equation-variable pair forms a causal, linear or non-causal relation, according to Definitions 6.1 and 6.2. Some of the equations of the air compressor model are used to exemplify different kinds of causality. For instance all the variables in equations e_1 and e_2 (corresponding to (7.4) and (7.5)) are assumed linear since they are linearly related (i.e. both equations can be rewritten according to Definition 6.2):

$$\begin{pmatrix} k_v & R_{cp} & -1 & 0 \\ B & -k_t & 0 & 1 \end{pmatrix} \begin{pmatrix} \omega_{cp} \\ i_{cp} \\ v_{cp} \\ \tau_{m_{cp}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{7.93}$$

As an example involving a non-invertible relation, consider equation e_5 (corresponding to (7.9)). This equation represents a typical saturation func-

tion. Hence, p_{ratio} is assigned as a causal variable and p_{atm} and $p_{cp,out}$ are non-causal variables. Another example of a non-invertible relation is found in equation e_8 (corresponding to (7.16)) involving a look-up table which is difficult or impossible to invert. Therefore, η_{cp} is a causal variable whereas $W_{cp,cor}$ and p_{ratio} are non-causal variables.

Next, a discussion follows on how causality is applied to differential equations for this application. Here, for the sake of a better residual implementation, integral causality is assumed. It should be noted that mixed (integral and derivative) causality could be used. However, since measurements are noisy and system dynamics are in general fast, derivative causality becomes unfeasible for a practical residual implementation. Therefore, derivative causality is avoided and no derivative variable is computed. The author is aware that assigning causality to differentiated variables is an issue under investigation and can affect the diagnosis properties of the system. Depending on whether integral, derivative or mixed causality is applied, fault detectability and isolability properties can vary (Frisk et al., 2010). This issue will not be studied here and causality on differentiated variables will be regarded as the standard causality defined in Chapter 6.

There are different approaches to represent differentiated variables in a structural model (Krysander et al., 2008). In this thesis, a variables and its time derivative are considered the same variables. This would be equivalent to consider a differential relation of the form

$$e_d : \dot{x} = \frac{d}{dt}x$$

and two different variables, x and $\dot{x}$. This is basically due to the fact that integral causality is assumed and each differentiated variable only appears in one equation. Consider, for example, equation e_3 where the differentiated variable, $\dot{\omega}_{cp}$, can also be represented by adding e_d in the model,

$$e_3 : \dot{\omega}_{cp} = \frac{1}{J_{cp}}\left(\tau_{m_{cp}} - C_p\frac{T_{atm}}{\eta_{cp}}(p_{ratio}^{\frac{\gamma-1}{\gamma}} - 1)W_{cp}\right) \tag{7.94}$$

$$e_d : \dot{\omega}_{cp} = \frac{d}{dt}\omega_{cp} \tag{7.95}$$

which corresponds, under integral causality, to the following causal structural model

	$\dot{\omega}_{cp}$	ω_{cp}	$\tau_{m_{cp}}$	T_{atm}	η_{cp}	p_{ratio}	W_{cp}
e_3	×		×	×	×	×	×
e_d	Δ	×					

However, as it was mentioned, equations e_3 and e_d could be merged by replacing $\dot{\omega}_{cp}$ to obtain

$$\omega_{cp} = \frac{1}{J_{cp}} \int_0^t \left[\tau_{m_{cp}} - C_p \frac{T_{atm}}{\eta_{cp}} (p_{ratio}^{\frac{\gamma-1}{\gamma}} - 1) W_{cp} \right] dt \tag{7.96}$$

which implies that ω_{cp} is assigned as a causal variable in e_3, whereas $\tau_{m_{cp}}$, T_{atm}, η_{cp}, p_{ratio} and W_{cp} are regarded as non-causal variables in e_3,

	ω_{cp}	$\tau_{m_{cp}}$	T_{atm}	η_{cp}	p_{ratio}	W_{cp}
e_3	$\times$	Δ	Δ	Δ	Δ	Δ

Therefore, in the structural model there is no distinction between, for example, ω_{cp} and $\dot{\omega}_{cp}$, and both are denoted by ω_{cp}.

The use of integral causality as in (7.96) implies to know the initial condition for the integration in order to obtain the real value of ω_{cp}. For this particular case and for the sake of simplicity, the system is always assumed to start in the same operating point, which is known. Nevertheless, if this was not the case, estimation techniques such as state observers could be considered to converge the model state to the initial operating point.

7.4 Sensor Placement for FCS Systems

The aim of this section is to solve the sensor placement problem for the FCS system by using the causal framework presented in Chapter 6. Especially, Algorithm 6.8 will be used to search for the optimal sensor configuration. The complete causal structural model used to solve the sensor placement problem is represented in Table 7.17 (in Appendix section 7.A).

Installing sensors for measuring any variable is not always possible or it may be difficult. For instance, measuring some internal variables in the fuel cell stack would require inserting probes into the stack which is physically impossible. Other variables like a partial mass in the gas mixture is considered not measurable because a complex measuring equipment is needed and therefore installing such device would not be realistic for practical applications. In all, 30 variables will be assumed to be measurable. Thus, the set of candidate sensors is

$$\begin{aligned}\mathbf{S} = \{&\omega_{cp}, \tau_{m_{cp}}, i_{cp}, W_{cp,out}, T_{cp,out}, \phi_{cp}, W_{sm,out}, T_{sm,out}, p_{sm,out}, \phi_{sm,out}, \\ &W_{ac,out}, T_{ac,out}, \phi_{ac,out}, W_{sh,out}, T_{sh,out}, p_{sh,out}, \phi_{sh,out}, W_{v,inj}, W_{om,out}, \\ &p_{om,out}, \phi_{om,out}, W_{afc,out}, p_{an,in}, W_{an,out}, p_{an,out}, \phi_{an,out}, W_{ca,out}, p_{ca,out},\end{aligned}$$

measurable variable	cost	measurable variable	cost	measurable variable	cost
ω_{cp}	10	$W_{ac,out}$	40	$\phi_{om,out}$	150
$\tau_{m_{cp}}$	25	$T_{ac,out}$	2	$W_{afc,out}$	40
i_{cp}	1	$\phi_{ac,out}$	150	$p_{an,in}$	5
$W_{cp,out}$	40	$W_{sh,out}$	40	$W_{an,out}$	40
$T_{cp,out}$	2	$T_{sh,out}$	2	$p_{an,out}$	5
ϕ_{cp}	150	$p_{sh,out}$	5	$\phi_{an,out}$	150
$W_{sm,out}$	40	$\phi_{sh,out}$	150	$W_{ca,out}$	40
$T_{sm,out}$	2	$W_{v,inj}$	100	$p_{ca,out}$	5
$p_{sm,out}$	5	$W_{om,out}$	40	$W_{v,an,out}$	100
$\phi_{sm,out}$	150	$p_{om,out}$	5	$W_{v,ca,out}$	100

Table 7.5: Measurable variables and costs.

$W_{v,an,out}, W_{v,ca,out}\}$

and their corresponding cost is depicted in Table 7.5.

Different dimensionless costs have been assigned to each measurable variable according to the ease of installation and the price of its corresponding sensor. For example, note that measuring humidity or vapour in gases has a large cost since the sensors are expensive and difficult to install in the system. On the other hand, installing sensors to measure air temperatures or pressures are easy and the price is low. Moreover, the measurements obtained from them are rather reliable. Therefore, this kind of sensors have an smaller costs. Air flow, angular speed and motor torque are assumed to be measurable at an intermediate cost.

The required specifications for this particular case are maximum causal detectability and maximum causal isolability for the set of faults in Table 7.4. No sensor fault is considered as well as no redundant sensor either.

First, the set of useful sensors, S_0, is determined according to (6.23). It is verified that all the proposed sensors are useful, i.e. $S_0 = \mathbf{S}$. Then, the maximum causal detectability and maximum causal isolability is computed according to (6.24) and (6.26), respectively. The specifications obtained are that all the faults are detectable and fully isolable,

$$F_{\mathcal{D}_{max}} = \{f_{cp1}, f_{cp2}, f_{sm}, f_{ac}, f_{sh}, f_{st}, f_{om}\}$$

and

$$
\begin{aligned}
F_{\mathcal{I}}(f_{cp1}) &= \{f_{cp2}, f_{sm}, f_{ac}, f_{sh}, f_{st}, f_{om}\} \\
F_{\mathcal{I}}(f_{cp2}) &= \{f_{cp1}, f_{sm}, f_{ac}, f_{sh}, f_{st}, f_{om}\} \\
F_{\mathcal{I}}(f_{sm}) &= \{f_{cp1}, f_{cp2}, f_{ac}, f_{sh}, f_{st}, f_{om}\} \\
F_{\mathcal{I}}(f_{ac}) &= \{f_{cp1}, f_{cp2}, f_{sm}, f_{sh}, f_{st}, f_{om}\} \\
F_{\mathcal{I}}(f_{sh}) &= \{f_{cp1}, f_{cp2}, f_{sm}, f_{ac}, f_{st}, f_{om}\} \\
F_{\mathcal{I}}(f_{st}) &= \{f_{cp1}, f_{cp2}, f_{sm}, f_{ac}, f_{sh}, f_{om}\} \\
F_{\mathcal{I}}(f_{om}) &= \{f_{cp1}, f_{cp2}, f_{sm}, f_{ac}, f_{sh}, f_{st}\}
\end{aligned}
$$

for $\mathbb{F}^1_{\mathcal{I}} = \{F_{\mathcal{I}}(f_{cp1}), F_{\mathcal{I}}(f_{cp2}), F_{\mathcal{I}}(f_{sm}), F_{\mathcal{I}}(f_{ac}), F_{\mathcal{I}}(f_{sh}), F_{\mathcal{I}}(f_{st}), F_{\mathcal{I}}(f_{om})\}$.

Algorithm 6.8 has been implemented in Matlab and then used to compute the optimal solution. The optimal set of sensors to be installed in the system is

$$S^* = \{\omega_{cp}, W_{cp,out}, T_{sm,out}, p_{sm,out}, T_{sh,out}, p_{sh,out}, W_{om,out}, p_{om,out}, p_{ca,out}\} \tag{7.97}$$

with a cost $C(S^*) = 114$. The solution is found in approximately 6 minutes.

7.5 Implementation of Causal Residuals

In this section, the family of causal MSO sets are first computed from the optimal solution (7.97). Then, an optimal set of those MSO sets is selected to later implement the corresponding residuals by using the computation sequence. The main objective of this section is to show how residuals are implemented for a diagnosis system within the causal framework.

7.5.1 Causal MSO Set Optimisation

Algorithm 6.9 is used to compute the causal MSO sets. First of all, the sensor equations for the optimal solution (7.97) must be added to the causal structural model. Running Algorithm 6.9, 63 causal MSO sets are found. These causal MSO sets are shown in Appendix section 7.B.

Not all MSO sets are required to diagnose the faults, but a reduced subset will suffice. To find the optimal reduced subset of causal MSO sets, the BILP formulation presented in Section 5.5.6 is used. Given a causal MSO set, ω, its corresponding cost $C(\omega)$ is defined by the number of fault equations appearing in ω. According to Table 7.4, this criterion can be stated as:

$$C(\omega) = |\omega \cap \{e_1, e_6, e_{11}, e_{24}, e_{31}, e_{58}\}| \tag{7.98}$$

	f_{cp1}	f_{cp2}	f_{sm}	f_{ac}	f_{sh}	f_{om}	f_{st}
$\omega_1 \rightarrow r_1$						×	
$\omega_2 \rightarrow r_2$				×			
$\omega_6 \rightarrow r_3$			×				
$\omega_7 \rightarrow r_4$					×		
$\omega_{41} \rightarrow r_5$		×					
$\omega_{57} \rightarrow r_6$	×						
$\omega_{60} \rightarrow r_7$							×

Table 7.6: Fault Signature Matrix for the FCS system.

For instance, ω_1 in Appendix section 7.B only contains fault equation e_{31}, so $C(\omega_1) = 1$, whereas $C(\omega_3) = 2$ since there are two fault equations (e_{16} and e_{31}) in ω_3. Recall from Section 5.5.5 that this criterion was proposed to maximize the isolability properties of the chosen MSO sets, Indeed, by doing this, the number of faults affecting each residual is minimal, which facilitates the task of fault isolation. The set of selected causal MSO sets is the following:

$$\omega_1, \omega_2, \omega_6, \omega_7, \omega_{41}, \omega_{57}, \omega_{60} \tag{7.99}$$

The corresponding Fault Signature Matrix is depicted in Table 7.6. Remark that all faults are causally detectable and causally isolable. Note also that thanks to forcing a minimal number of faults for each causal MSO set, it is possible, for this particular case, to obtain a one-to-one relationship between a fault and a residual, which trivialises the fault isolation task.

7.5.2 Residual Implementation and Simulation

Seven residuals, $r_1, \ldots, r_7$, are derived from the selected causal MSO set (7.99) (see Table 7.6). The computation sequence, as introduced in Chapter 6, is used to implement the corresponding residual generators[1]. Such computation sequences are depicted in Appendix section 7.C. Residuals are computed by forward value propagation without non-linear loops. Residuals relate measurements in the optimal set of sensors S^* (7.97) and the *a priori* known variables (atmospheric variables, setpoints and control variables). It is also important to point out that linear loops are not present in

[1]An observer has been implemented in residual generator r_6 in order to compensate the error produced by uncertainties in the dynamic equation.

any implemented residual, though linear relations are allowed in the causal framework.

For the sake of readability, relations among components and sensor equations are omitted in Appendix section 7.C, and only the main FCS model equations are depicted (i.e. equations from Table 7.3). These computation sequences are implemented in Simulink as residual generators. Then, they are tested against the FCS model presented in Section 7.3.

To make the approach more realistic, noise is added in the measurements. Four kind of sensors are considered: air flow sensors, angular speed sensors, pressure sensors and temperature sensors. The noise is modelled as a white noise with a Gaussian distribution. Table 7.7 shows nominal order of magnitude of the different kinds of sensor and their standard deviation.

	Nominal magnitude	Noise standard deviation
Air flow	$\backsim 0.1 kg/s$	$4.5 \cdot 10^{-4} kg/s$
Angular speed	$\backsim 10^4 rad/s$	$10 rad/s$
Air pressure	$\backsim 3 \cdot 10^5 Pa$	$300 Pa$
Temperature	$\backsim 350 K$	$0.07 K$

Table 7.7: Noise standard deviation.

Some modelling errors or uncertainties are also introduced in order to set up a more realistic simulation. This is done by modifying some values of the parameters involved in the equations of the computation sequences. Hence, discrepancies between the FCS system model and the equations in the residual generators are obtained, which is something often encountered in real applications. Table 7.8 shows both parameter values: one is used in the FCS system simulation model and the other in the residual generator implementation.

A series of simulations have been performed in order to test the residuals against different fault scenarios. In each simulation, the system operates in the same operating points. This is achieved by modifying the input variables of the model which obey to system load changes. In Figure 7.3, the compressor voltage and the stack current pattern used in all simulations is depicted.

Figures 7.4 and 7.5 show the normalised residuals for each simulation. Thresholds are roughly defined in order to set the intervals where residuals

Parameter description	Symbol	In the FCS system simulator	In the residual generator	Units
Gas constant	R_a	289.9	287	$J/(kg \cdot K)$
Vapour molar mass	M_v	$18.015 \cdot 10^{-3}$	$18.02 \cdot 10^{-3}$	kg/mol
Compressor diameter outlet	d_c	0.2286	0.2287	m
Compressor inertia	J	$5.01 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	kg/m^2
Speed-to-voltage coefficient	k_v	$15.3012 \cdot 10^{-3}$	$14.5 \cdot 10^{-3}$	$V/(rad/s)$
Current-to-torque coefficient	k_t	$22.496 \cdot 10^{-3}$	$24 \cdot 10^{-3}$	$N \cdot m/Amp$
Friction coefficient	B	$1.91 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$N \cdot m/(rad/s)$
Electric resistance	R	1.16	1.2	Ω
Supply manifold volume	V_{sm}	0.01838	0.02	m^2
Outlet manifold volume	V_{om}	$5.313 \cdot 10^{-3}$	$5 \cdot 10^{-3}$	m^2

Table 7.8: Modelling errors.

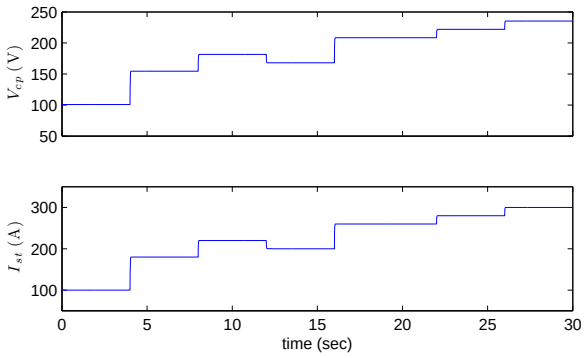


Figure 7.3: Input signal pattern.

are not activated. Specifically, in Figure 7.4(a) the evolution of the residuals for the fault free case is shown. Observe that, residual r_6 is sensitive to operating point changes. However, thanks to the observer the residual is

confined inside the interval defined by the thresholds. The residual responses corresponding to fault scenarios are represented in the remaining subplots (Figure 7.4(b)- 7.5(h)). Note that, despite the noise and modelling errors, it is possible to determine the occurrence of a fault by checking whether the residual crosses the threshold. Furthermore, by taking into consideration the fault signature matrix in Table 7.6, it is straightforward to determine which of the seven possible faults has occurred.

In these simulations, fault magnitudes concerning multiplicative faults are quantified between 5% and 10% of the nominal parameter or unknown variable, except for the compressor motor fault f_{cp1} which is 30%. Concerning additive faults, the air cooler fault f_{ac} is simulated as an increment of $0.4K$ in the desired air temperature, whereas the static humidifier fault f_{sh} is simulated as a 1% increment in the desired relative humidity of the air. All faults occur after 10 seconds of the beginning of the corresponding simulation scenario.

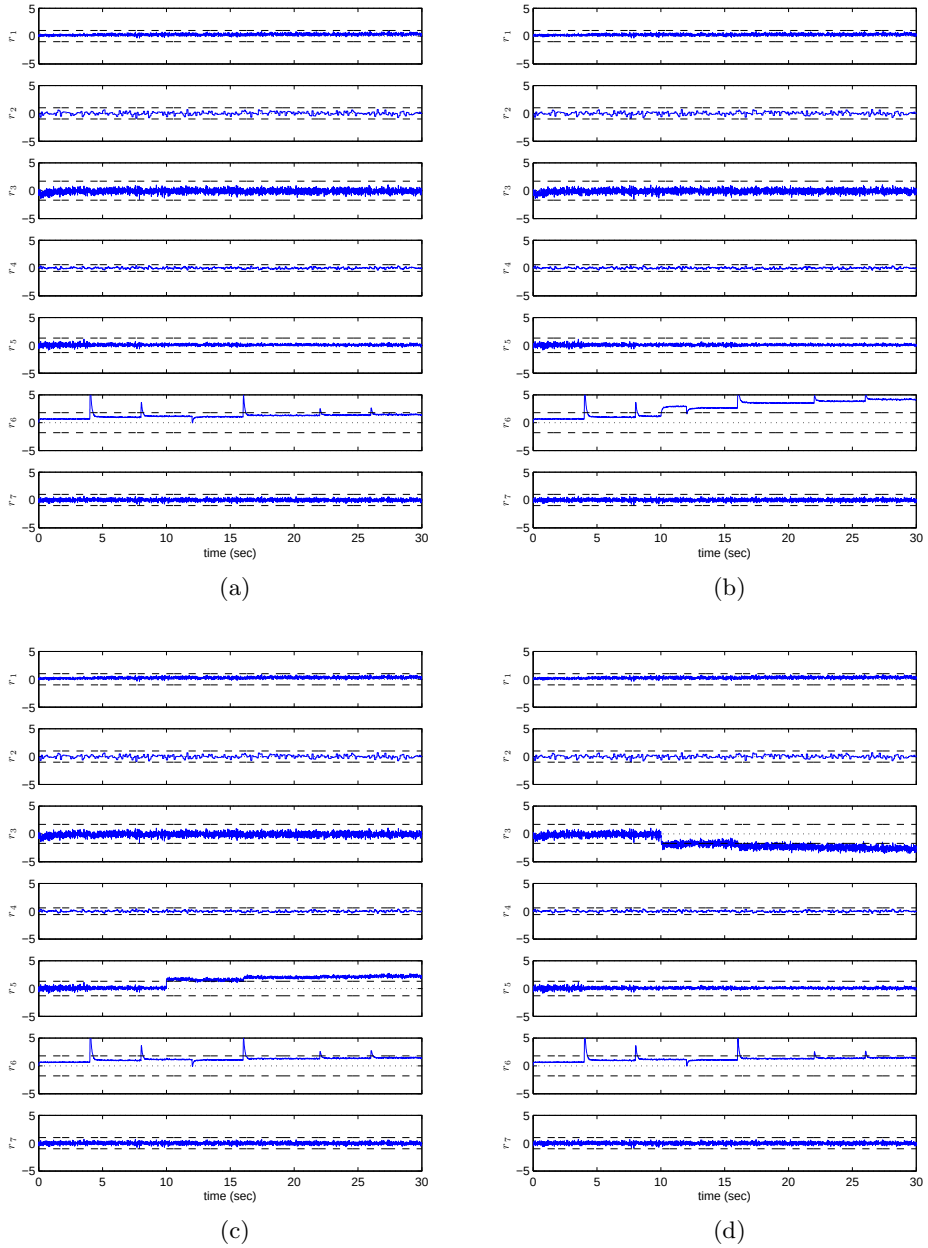
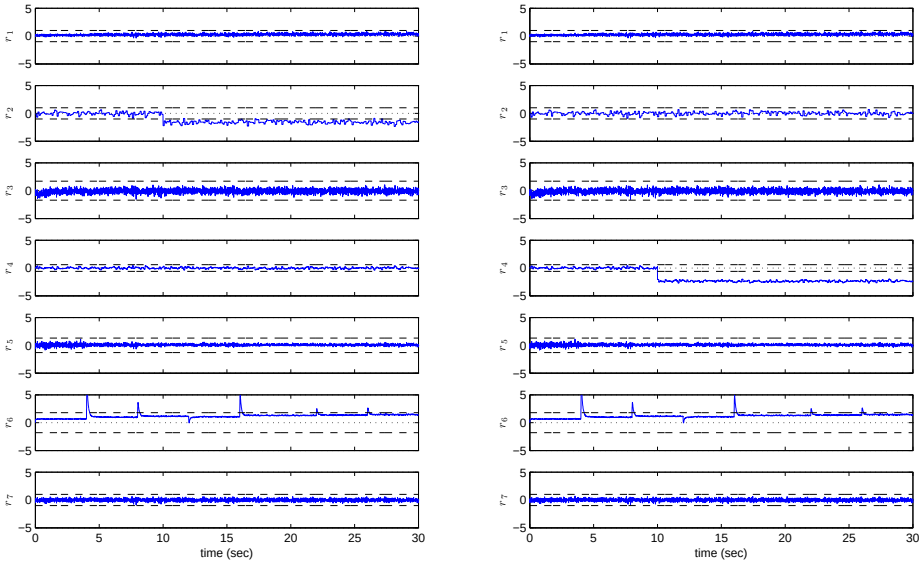
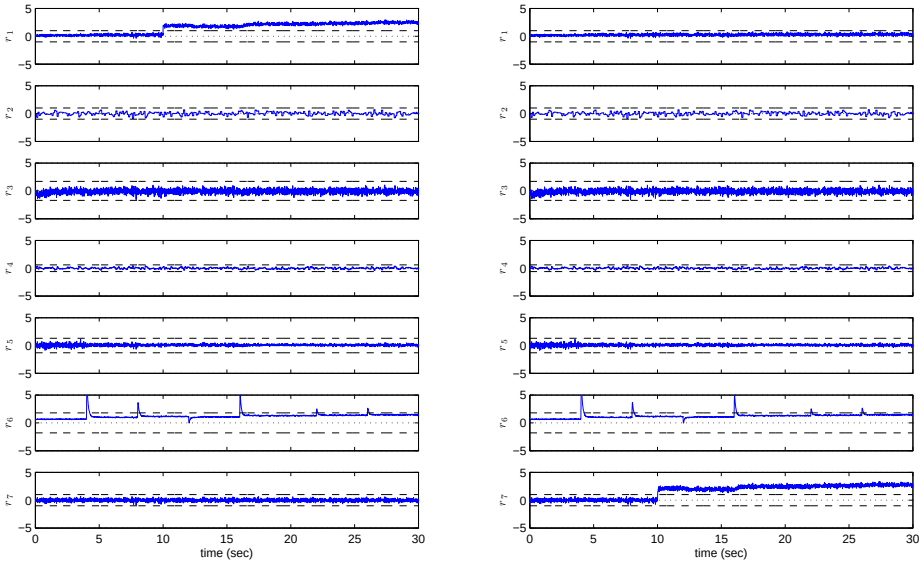


Figure 7.4: Simulation scenarios: (a) fault free scenario; (b) fault f_{cp1} scenario; (c) fault f_{cp2} scenario; (d) fault f_{sm} scenario.



(e)

(f)



(g)

(h)

Figure 7.5: Simulation scenarios: (e) fault f_{ac} scenario; (f) fault f_{sh} scenario; (g) fault f_{om} scenario; (h) fault f_{st} scenario.

7.6 Comparison with Alternative Approaches

The choice of Algorithm 6.8 to solve the sensor placement problem for the FCS system is motivated in this section. A comparison to the other approaches presented in this thesis is given next.

All the approaches proposed in this thesis could be used to solve the sensor placement problem for the FCS system within the causal framework. The incremental MSO set generation approach presented in Chapter 4 (Algorithm 4.1) could be adapted to only generate causal MSO sets, whereas the binary programing-based approach in Chapter 5 would require computing beforehand the whole set of causal MSO set with all candidate sensors installed in the system. However, as argued below, Algorithm 6.8, presented in Chapter 6, is the most suitable.

Bear in mind that computing the causal MSO sets with the optimal set of sensors S^* (7.97) installed in the system requires approximately 20 minutes (the redundancy degree of this model φ_s is 9). However, the complexity of computing causal MSO sets is, theoretically, exponential with respect to the structural redundancy degree. Thus, the time required to compute the MSO sets with all the candidate sensors installed in the system (with a redundancy degree $\varphi_s = 30$) is extremely large. Indeed, it has been tried to compute such sets by running a computer for several days but the algorithm did not end up. This motivates avoiding binary programming-based approaches to solve the sensor placement problem for the FCS system.

Next, the performance of the optimal search in the incremental MSO set generation approach is analysed. First of all, note that the set of candidate sensors involves 30 possible measurable variables. This implies that the search space consists of $2^{30} \approx 10^9$ possible configurations. Recall that, in this approach, the search strategy consists in starting with no sensor and then iteratively adding sensors according to their cost until a sensor configuration that fulfils the diagnosis specifications is found. The optimal solution S^* (7.97) has a cost of 114. Therefore, there are 180000 sensor configurations with a lower cost than 114. This implies that the incremental algorithm would traverse these sensor configurations in order to find the solution. Moreover, this also implies that 180000 sets of MSO sets ($[\omega]_{S_i}$ for $i \in \{1, \ldots, 180000\}$) need to be efficiently stored and manipulated by the algorithm, which poses a serious inconvenience to the applicability of this approach to the FCS system.

On the other hand, the sensor search strategy proposed in Algorithm 6.8 is more efficient than these other approaches. Actually, it has been found that Algorithm 6.8 only traverses 1854 sensor configurations. The diagnosis

specifications are tested just for 308 of those traversed configurations (i.e. Algorithm 6.6 receives 308 calls). The number of feasible solutions encountered during the search is 152, which means that the depth-first search is aborted in all the remaining configurations (i.e. Algorithm 6.6 aborts the depth-first search 1702 times). That explains why the solution is found in 6 minutes whereas in the other approaches it would take days.

7.7 Conclusions

Implementing diagnosis systems for large models involving non-linear equations is not a trivial issue. In this chapter, a systematic methodology to design such diagnosis systems has been proposed by using one of the algorithms previously presented in this thesis. Furthermore, a real case application based on a fuel cell stack system has been used to motivate and exemplify the diagnosis system design.

Given a set of non-linear equations describing the non-faulty behaviour of the system, the diagnosis system design methodology can be summarised in the following four main steps:

1. Extract the causal structural model of the system. The system is decomposed into components. Each component is modelled apart, and later extra interconnections are added to the model. To obtain the causal structural model, causal, non-causal and linear relations have to be identified.

2. Solve the optimal sensor placement problem in the causal framework. This requires assigning a cost to each possible sensor and computing the maximum achievable specifications.

3. Obtain the causal MSO sets needed for the diagnosis system. This implies computing first the causal MSO sets when the optimal sensor set is installed in the system, and secondly choosing the optimal set of causal MSO sets.

4. Given the optimal causal MSO sets, implement the residual generators by using the computation sequence. The diagnosis specifications can be then verified by simulation.

In Step 1, the model equations must be described as detailed as possible in order to avoid unintentional simplifications. Step 2 and its implementation is the main topic of this thesis. Once the optimal solution is found,

Step 3 can be carried out. Here, Algorithm 6.9 has been used to directly compute the causal MSO sets when the sensors S^* are installed in the system. The algorithm needs 20 minutes to compute the 63 causal MSO sets. It is important to point out that the complete set of MSO sets obtained with this solution when no causality is regarded is 2619201. Therefore, another possible approach would be first compute the whole set of MSO sets and later reject those that are not causal, to finally obtain the true 63 causal MSO sets. However, this would not be as efficient as directly computing the causal MSO sets. After computing the causal MSO sets, an optimal subset is sought by means of a binary optimisation.

Step 4 is devoted to residual implementation by first implementing the computational sequence which is ensured due to the existence of a causal matching without non-linear loops in every causal MSO set, and then testing its performance in simulation and/or the real plant whenever possible. At this step and looking at the results shown in Figures 7.4 and 7.5, it is worth to mention that worse results could be expected. This is because, since the approach is based on a structural model, it is only ensured that residuals are not affected by certain faults, whereas the reverse is not ensured. Thus, some residuals could be not sensitive to its corresponding fault according to Table 7.6.

Finally, the same problem could be solved without considering causality. This has been done with Algorithm 6.8 by using a causal structural model without causality assignments. The solution is found in 4 seconds. The new optimal sensor configuration is $\{i_{cp}, p_{om,out}\}$ with cost 6 (*a priori*, this is a better configuration than S^* (7.97)). From this sensor configuration, 26 non-causal MSO sets could be generated. However, each MSO set would contain, on average, 90 equations. This would hamper the implementation of the residual generators since non-linear equations should have to be solved in a non-causal way.

Appendix

7.A Causal Structural Models of the FCS System

Causal structural models for the FCS system are here described by means of bi-adjacency matrices. There is one structural model for each component in the FCS system. The corresponding bi-adjacency matrices are represented by means of the following tables. At the end of this section, in Table 7.17, all these models are gathered into the complete causal structural model of the FCS system.

	v_{cp}	ω_{cp}	$\tau_{m_{cp}}$	i_{cp}	ϕ_{atm}	$W_{cp,out}$	$T_{cp,out}$	$p_{cp,out}$	ϕ_{cp}	p_{ratio}	$W_{cor,cp}$	η_{cp}	T_{atm}	p_{atm}	$T_{cp,cor}$
e_1	L	L		L											
e_2		L	L	L											
e_3		×	Δ			Δ				Δ		Δ	Δ		
e_4													L		L
e_5								Δ		×				Δ	
e_6		Δ								Δ	×		Δ		Δ
e_7						×					×			×	Δ
e_8										Δ	Δ	×			
e_9							×			Δ		×	×		
e_{10}					×		Δ	×	×				Δ	×	

Table 7.9: Air compressor model.

	$W_{sm,in}$	$T_{sm,in}$	$p_{sm,in}$	$\phi_{sm,in}$	$p_{sm,ds}$	$W_{sm,out}$	$T_{sm,out}$	$p_{sm,out}$	$\phi_{sm,out}$	m_{sm}
e_{11}			L		L	L				
e_{12}	Δ					Δ				×
e_{13}	Δ	Δ				Δ	Δ	×		
e_{14}							×	×		×
e_{15}		Δ	×	×			Δ	×	×	

Table 7.10: Supply manifold model.

	$W_{ac,in}$	$T_{ac,in}$	$p_{ac,in}$	$\phi_{ac,in}$	T_{des}	$W_{ac,out}$	$T_{ac,out}$	$p_{ac,out}$	$\phi_{ac,out}$
e_{16}					L		L		
e_{17}		Δ	×	×			Δ	×	×
e_{18}	L					L			
e_{19}			L					L	

Table 7.11: Air cooler model.

	$W_{sh,in}$	$T_{sh,in}$	$p_{sh,in}$	$\phi_{sh,in}$	ϕ_{des}	$W_{sh,out}$	$T_{sh,out}$	$p_{sh,out}$	$\phi_{sh,out}$	$W_{v,inj}$	$p_{v,sh,in}$	$p_{a,sh,in}$	$p_{v,sh,out}$	$p_{a,sh,out}$
e_{20}		L					L							
e_{21}		Δ		×							×			
e_{22}			L								L	L		
e_{23}							Δ		×				×	
e_{24}					L				L					
e_{25}												L		L
e_{26}	×									×	×	×	×	×
e_{27}	L					L				L				
e_{28}								L					L	L

Table 7.12: Static humidifier model.

Table 7.16: Structural relations for component interconnections and known variables.

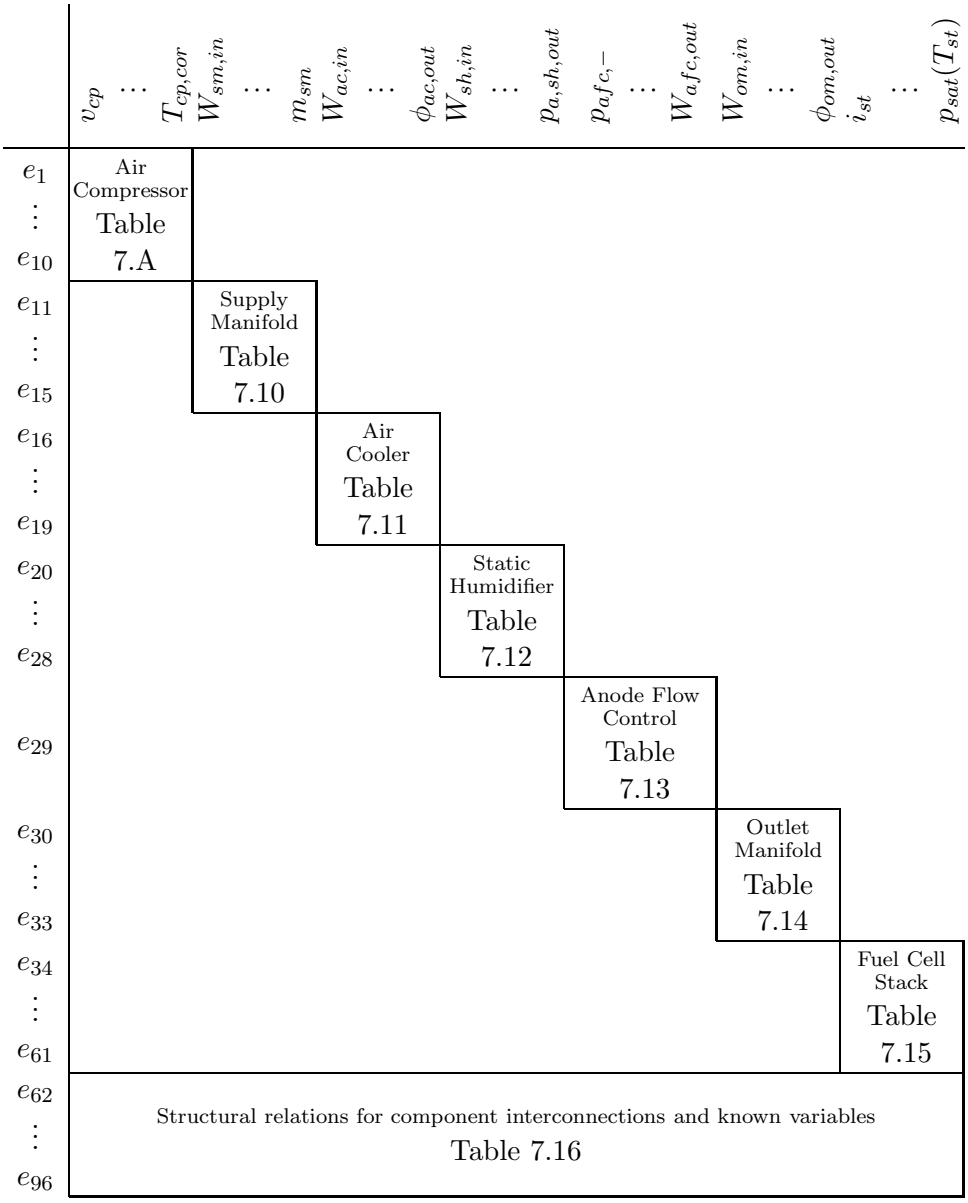


Table 7.17: Causal structural Fuel Cell System.

7.B Causal MSO Set of FCS Model Equations

Next, the set of causal MSO sets for the FCS system application are detailed. These MSO sets are computed for the optimal set of sensor S^* (7.97). Note that the sensor equations corresponding to S^* are $e_{\omega_{cp}}$, $e_{W_{cp,out}}$, $e_{T_{sm,out}}$, $e_{p_{sm,out}}$, $e_{T_{sh,out}}$, $e_{p_{sh,out}}$, $e_{W_{om,out}}$, $e_{p_{om,out}}$ and $e_{p_{ca,out}}$.

$$\omega_1 = \{e_{31}, e_{65}, e_{87}, e_{91}, e_{W_{om,out}}, e_{p_{om,out}}, e_{T_{sh,out}}\}$$

$$\omega_2 = \{e_{16}, e_{20}, e_{67}, e_{83}, e_{T_{sh,out}}\}$$

$$\omega_3 = \{e_{16}, e_{20}, e_{31}, e_{65}, e_{67}, e_{83}, e_{87}, e_{91}, e_{W_{om,out}}, e_{p_{om,out}}\}$$

$$\omega_4 = \{e_{11}, e_{12}, e_{14}, e_{69}, e_{71}, e_{73}, e_{78}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}\}$$

$$\omega_5 = \{e_4, e_5, e_7, e_8, e_9, e_{11}, e_{13}, e_{63}, e_{64}, e_{69}, e_{70}, e_{71}, e_{73}, e_{78}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}\}$$

$$\omega_6 = \{e_4, e_5, e_7, e_8, e_9, e_{11}, e_{12}, e_{13}, e_{14}, e_{63}, e_{64}, e_{69}, e_{70}, e_{71}, e_{73}, e_{78}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{ca,out}}\}$$

$$\omega_7 = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{15}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{68}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_8 = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{15}, e_{16}, e_{17}, e_{19}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_9 = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{10} = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{T_{sm,out}}\}$$

$$\omega_{11} = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11}, e_{13}, e_{15}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{12} = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11}, e_{13}, e_{15}, e_{16}, e_{17}, e_{19}, e_{21}, e_{22}, e_{23}, e_{24}e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{13} = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11}, e_{13}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{14} = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11}, e_{13}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}\}$$

$$\omega_{15} = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{16} = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{68}, e_{69}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sh,out}}\}$$

$$\omega_{17} = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{18} = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sh,out}}\}$$

$$\omega_{19} = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{20} = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}\}$$

$$\omega_{21} = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sh,out}}\}$$

$$\omega_{22} = \{e_4, e_5, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}\}$$

$$\omega_{23} = \{e_4, e_5, e_6, e_8, e_9, e_{11}, e_{13}, e_{63}, e_{64}, e_{69}, e_{70}, e_{71}, e_{73}, e_{78}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}\}$$

$$\omega_{24} = \{e_4, e_5, e_6, e_8, e_9, e_{11}, e_{12}, e_{13}, e_{14}, e_{63}, e_{64}, e_{69}, e_{70}, e_{71}, e_{73}, e_{78}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{ca,out}}\}$$

$$\omega_{25} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{15}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{68}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{83}, e_{84}, e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{26} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{15}, e_{16}, e_{17}, e_{19}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{83}, e_{84}, e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{27} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{24}, e_{23}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{84}, e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{28} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{83}, e_{84}, e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{T_{sm,out}}\}$$

$$\omega_{29} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{11}, e_{13}, e_{15}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{30} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{11}, e_{13}, e_{15}, e_{16}, e_{17}, e_{19}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{31} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{11}, e_{13}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{32} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{11}, e_{13}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}\}$$

$$\omega_{33} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{34} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{68}, e_{69}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sh,out}}\}$$

$$\omega_{35} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{36} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sh,out}}\}$$

$$\omega_{37} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{38} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}\}$$

$$\omega_{39} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sh,out}}\}$$

$$\omega_{40} = \{e_4, e_5, e_6, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}\}$$

$$\omega_{41} = \{e_4, e_5, e_6, e_7, e_{63}, e_{64}, e_{73}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}\}$$

$$\omega_{42} = \{e_4, e_5, e_6, e_7, e_{11}, e_{12}, e_{14}, e_{63}, e_{64}, e_{69}, e_{71}, e_{73}, e_{78}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}\}$$

$$\omega_{43} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{11}, e_{13}, e_{63}, e_{64}, e_{69}, e_{70}, e_{71}, e_{73}, e_{78}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}\}$$

$$\omega_{44} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{11}, e_{12}, e_{13}, e_{14}, e_{63}, e_{64}, e_{69}, e_{70}, e_{71}, e_{73}, e_{78}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{ca,out}}\}$$

$$\omega_{45} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{13}, e_{15}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{46} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{13}, e_{15}, e_{16}, e_{17}, e_{19}, e_{21}, e_{22}, e_{23}, e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{47} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{13}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24},$$

$$e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{84}, e_{85},$$
$$e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{48} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{13}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24},$$
$$e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84},$$
$$e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}\}$$

$$\omega_{49} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24},$$
$$e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84}, e_{85},$$
$$e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{50} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}, e_{24},$$
$$e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{68}, e_{69}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84},$$
$$e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sh,out}}\}$$

$$\omega_{51} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{21}, e_{22}, e_{23}, e_{24},$$
$$e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83}, e_{84},$$
$$e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{52} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{21}, e_{22}, e_{23}, e_{24},$$
$$e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83},$$
$$e_{84}, e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sh,out}}\}$$

$$\omega_{53} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23},$$
$$e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{84},$$
$$e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}, e_{T_{sh,out}}\}$$

$$\omega_{54} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23},$$
$$e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78}, e_{83},$$
$$e_{84}, e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sm,out}}\}$$

$$\omega_{55} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23},$$
$$e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78},$$
$$e_{84}, e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}, e_{T_{sh,out}}\}$$

$$\omega_{56} = \{e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{14}, e_{15}, e_{16}, e_{17}, e_{19}, e_{20}, e_{21}, e_{22}, e_{23},$$
$$e_{24}, e_{25}, e_{28}, e_{62}, e_{63}, e_{64}, e_{67}, e_{68}, e_{69}, e_{70}, e_{71}, e_{72}, e_{73}, e_{75}, e_{76}, e_{77}, e_{78},$$
$$e_{83}, e_{84}, e_{85}, e_{\omega_{cp}}, e_{p_{sm,out}}, e_{p_{sh,out}}, e_{p_{ca,out}}\}$$

$$\omega_{57} = \{e_1, e_2, e_3, e_4, e_5, e_7, e_8, e_{63}, e_{64}, e_{73}, e_{96}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}\}$$

$$\omega_{58} = \{e_1, e_2, e_3, e_4, e_5, e_6, e_8, e_{63}, e_{64}, e_{73}, e_{96}, e_{W_{cp,out}}, e_{\omega_{cp}}, e_{p_{sm,out}}\}$$

$$\omega_{59} = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_{63}, e_{64}, e_{73}, e_{96}, e_{\omega_{cp}}, e_{p_{sm,out}}\}$$

$$\omega_{60} = \{e_{58}, e_{30}, e_{86}, e_{87}, e_{91}, e_{94}, e_{W_{om,out}}, e_{p_{om,out}}, e_{p_{ca,out}}, e_{T_{sh,out}}\}$$
$$\omega_{61} = \{e_{58}, e_{30}, e_{31}, e_{65}, e_{86}, e_{87}, e_{91}, e_{94}, e_{p_{om,out}}, e_{p_{ca,out}}, e_{T_{sh,out}}\}$$
$$\omega_{62} = \{e_{58}, e_{16}, e_{20}, e_{30}, e_{67}, e_{83}, e_{86}, e_{87}, e_{91}, e_{94}, e_{W_{om,out}}, e_{p_{om,out}}, e_{p_{ca,out}}\}$$
$$\omega_{63} = \{e_{58}, e_{16}, e_{20}, e_{30}, e_{31}, e_{65}, e_{67}, e_{83}, e_{86}, e_{87}, e_{91}, e_{94}, e_{p_{om,out}}, e_{p_{ca,out}}\} \quad (7.100)$$

7.C Computation Sequences

Next, the computation sequences corresponding to the optimised causal MSO sets, ω_1, ω_2, ω_6, ω_7, ω_{41}, ω_{57}, ω_{60} are detailed. Hence, deriving residual generators is straightforward since the causal residual computation is always ensured.

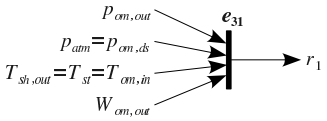


Figure 7.6: Computation sequence for r_1.

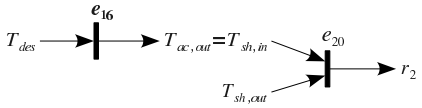


Figure 7.7: Computation sequence for r_2.

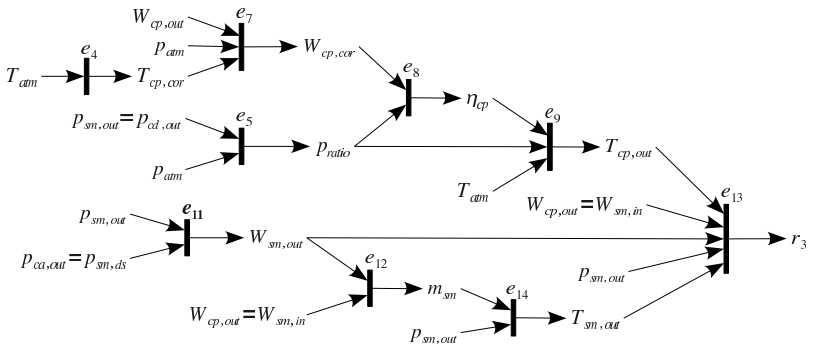


Figure 7.8: Computation sequence for r_3.

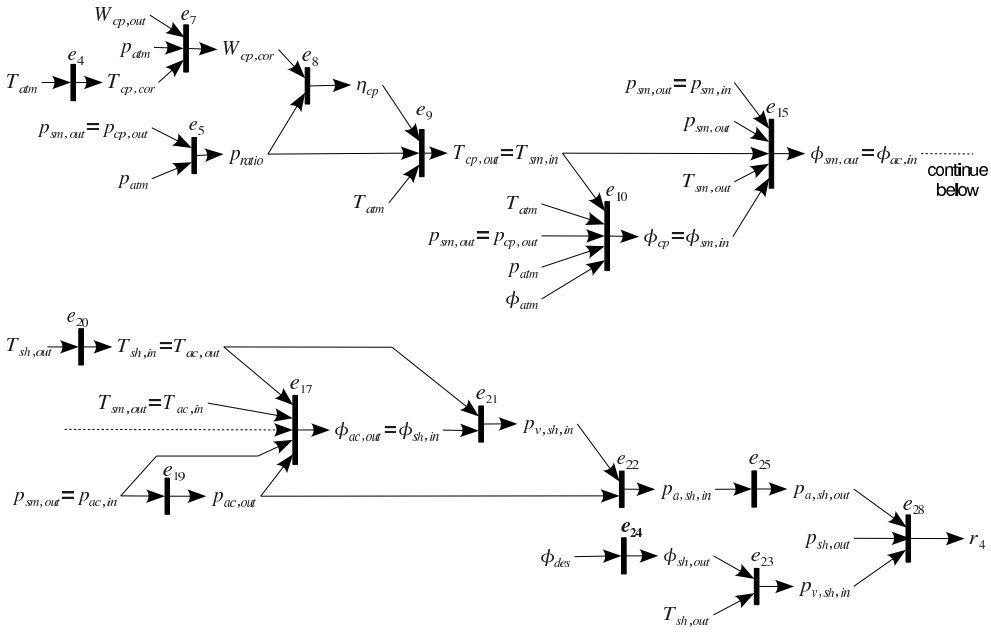


Figure 7.9: Computation sequence for r_4.

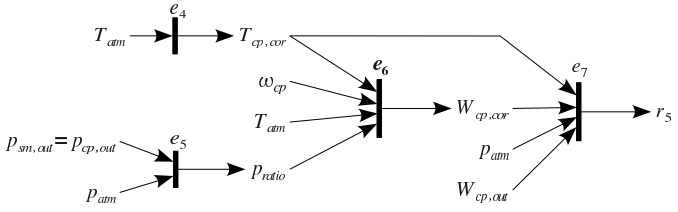


Figure 7.10: Computation sequence for r_5.

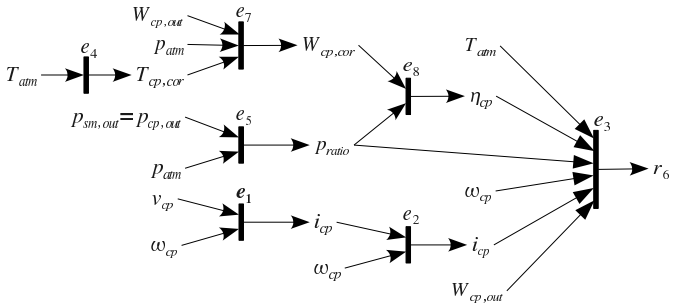


Figure 7.11: Computation sequence for r_6.

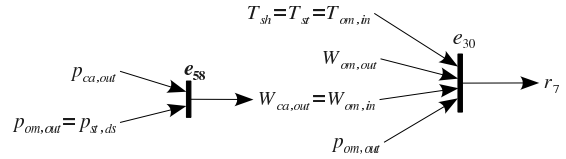


Figure 7.12: Computation sequence for r_7.

Chapter 8

Concluding Remarks

The conclusions and contributions drawn from this thesis are presented in this chapter. First, the main contributions are summarised. In the conclusions section, the importance of placing sensors for diagnosis is highlighted. Also, general conclusions derived from the sensor placement problem analysis are detailed. Next, a comparison of the basic features of the three approaches presented throughout this thesis is presented. Finally, some guidelines are given for future works and extensions.

8.1 Summary of Contributions

The sensor placement has traditionally been a marginal topic in the diagnosis field. In the last few years, this subject has gained the interest of several researches, and new contributions have been recently published. This thesis takes advantage of this, by using or improving these new works. On the other hand, new contributions and developments are achieved, where the existing works are not suitable. The main contributions are next summarised:

1. **Framework development for sensor placement analysis:** Chapter 3. The bases for the sensor placement analysis are established based on existing literature. In order to do so, definitions for fault detectability and isolability, based on either structural model or MSO sets properties, are introduced. This allows to handle the same diagnosis specifications throughout this thesis. Furthermore, a suitable representation of sensors is chosen to efficiently handle them in all proposed methodologies. Maximum diagnosis specifications are also determined.

2. **Redundant sensors in the sensor placement problem:** Theorem 3.4. A novel contribution in Chapter 3 is the study of how redundant sensors affect fault detectability and isolability properties for a given structural model. The results derived from Theorem 3.4 are later used to efficiently handle redundant sensors in all proposed methods. Similar results were achieved in Krysander and Frisk (2008) and Frisk et al. (2009). However, in these works, no explicit redundant sensor consideration is done and the results are only valid for models without under-determined part.

3. **Sensor placement by incremental MSO set generation:** (Rosich et al., 2007) and Chapter 4. A method to solve the sensor placement based on MSO set properties is proposed. The novelty is that not all the MSO sets are computed, being this approach more efficient than similar existing approaches. A key contribution for the applicability of this method is Theorem 4.1. Given a sensor configuration, MSO sets can be incrementally generated by only computing those MSO sets that contain all the corresponding sensor equations. It is ensured that the remaining MSO sets can be reused from previously tested sensor configurations.

4. **Binary non-linear programming for sensor placement:** (Sarrate et al., 2007) and, Sections 5.2 and 5.3. The sensor placement problem is addressed as a binary integer optimisation problem. This is mainly possible due to a MSO set selector constraint (5.7), which validates MSO sets for a given sensor configuration.

5. **Binary linear programming for sensor placement:** (Rosich et al., 2009c), (Nejjari et al., 2010) and Section 5.5. The binary non-linear programming formulation turns into a linear formulation by defining a linear MSO set selector constraint (5.37). This allows the use of standard BILP optimisation techniques. This formulation can be used not only to solve the sensor placement problem (Section 5.5.4) but also the MSO set selection. This can be done together with the sensor selection (Section 5.5.5) or independently (Section 5.5.6).

6. **Causal computation for residual generators:** (Rosich et al., 2009a) and Chapter 6. A framework to properly deal with the computation of unknown variables when residual generators are derived from MSO sets is implemented. Algorithms that ensure a feasible variable

computation are developed. Furthermore, fault detectability and isolability are redefined by taking into account such causal computations.

7. **Sensor placement for causal computation framework:** (Rosich et al., 2009a) and Chapter 6. The sensor placement problem is addressed when causal computations are considered. It is important to mention that no MSO set is generated to solve such problem.

8. **Efficient optimal sensor search strategy:** (Frisk et al., 2010) and Section 6.6. The new search strategy developed to find the optimal sensor configuration allows, in general, to solve the sensor placement problem for a large number of candidate sensors. Indeed, there are some practical cases (e.g. the FCS system) that can only be solved in a reasonable time, at the moment, by using such search strategy.

9. **Sensor placement and residual generators for a FCS system:** (Rosich et al., 2009b) and Chapter 7. Sensor placement and residual generator techniques are applied on a FCS system. This is a novel contribution to the fuel cell community regarding the diagnosis system design.

8.2 Conclusions

In this thesis, the sensor placement problem based on fault diagnosis has been addressed as the main topic. It has been shown that diagnosis capabilities of the system can be significantly improved by installing the appropriate set of sensors. Furthermore, the diagnosis system design can be simplified when the problem of placing suitable sensors is solved beforehand. For these reasons, the sensor placement problem should be considered in the design of a process to be diagnosed, or to improve the diagnosis capabilities of the diagnosis system to be implemented in an existing process.

Solving the sensor placement for diagnosis can be treated from many different points of view. Indeed, such a problem depends on the kind of system description, the required diagnosis specifications, as well as the technique used to implement the diagnosis system. Because of this, developing a sensor placement method, that works for all possible fault diagnosis systems, is unattainable. Here, fault diagnosis systems are based on consistency checking by means of structural models. The required diagnosis specifications to be fulfilled are fault detection and isolation for both, process and sensor faults. In author's opinion, such diagnosis requirements are the most

elementary that a diagnosis system must fulfil. However, some of the methods presented here could be easily extended to other possible diagnosis requirements.

Considering faults in sensors is a delicate issue since the more sensors are added to the system, the larger the number of faults to be diagnosed is. Consequently, full fault isolability concerning all faults is generally not possible, or the number of sensors required for installation is too large. A good solution to deal with this issue is to consider redundant sensors, as long as installing the same sensor more than once in the system is feasible. It has been proved that adding redundant sensors only improves sensor fault diagnosis. Therefore, the sensor placement problem for diagnosis when redundant sensor are accepted has been addressed in this thesis. Furthermore, efficient strategies to work with redundant sensors have been developed for the proposed approaches.

The nature of the sensor placement problem, as seen in this thesis, is basically combinatorial. This means that the search space exponentially grows with the number of sensors to be considered. To cope with this issue, the problem has been set up as an optimisation problem (the optimal sensor set is sought). Different optimisation techniques have been introduced based on the approach suitability. These optimisation techniques cover standard optimisation formulation (e.g. BILP formulation) and optimum search algorithms as well.

MSO set generation methods have also been addressed in this thesis as a consequence of the study of the sensor placement problem. Specifically, algorithms for incremental generation of MSO sets and causal MSO sets have been developed.

Another secondary topic addressed within the sensor placement problem is the computation of unknown variables in a residual. A framework that takes into account how unknown variables can be computed for a given structural model has been introduced. Finally, such approach has been tested in a process based on a fuel cell stack. The results have proved that the method is perfectly applicable and efficiently works for large-scale systems.

8.3 Proposed Approaches Discussion

In the following, the main features of the three approaches presented in this thesis are compared and discussed. Below, we reefer to the *incremental* approach, the *binary programming* approach and the *causal computation*

approach introduced in Chapters 4, 5 and 6, respectively.

An important feature is the number of MSO sets generated during the sensor search. As mentioned before, generating MSO sets has a high computational cost. Therefore, the fewer MSO sets are computed, the more efficient the approach is. The *binary programming* approach requires the whole set of all the MSO sets generated beforehand. The *incremental* approach efficiently computes only those MSO sets required at each iteration of the sensor search. Thus, in general, fewer MSO set are generated. On the other hand, the *causal computation* approach needs no MSO sets. Thus, it is the most efficient approach from the required MSO sets point of view.

Another feature to be taken into account is the sensor search efficiency. All the proposed approaches search for the optimal sensor configuration. How this search is performed varies from one approach to another. First, it is worth noting that, in the worst case, the computation time of all the adopted search strategies exponentially depends on the number of sensors. However, in general, there are approaches that are faster than others. The most time demanding approach is the *incremental* approach. This is due to the search strategy being rather simple, since it is based on testing sensor configurations in increasing cost until a solution is found. The *binary programming* approach, specially the BILP formulation, should be the most efficient since it uses standard and well-studied algorithms. However, its efficiency is penalised as the optimisation vector is increased by the MSO set selector variables. Thus, it can be stated that the *binary programming* approach has a medium efficiency. Finally, the *causal computation* approach using Algorithm 6.8 results to be the most efficient since sets of sensor configurations are cleverly rejected from the diagnosis specification test. Equivalent size problems have been tested in the three approaches, being the *causal computation* approach the fastest.

The last feature to be analysed is how easily the proposed approaches can be adapted to a diagnosis specification other than fault detectability or isolability. Defining diagnosis specifications as MSO set properties is in general easier than defining the same diagnosis specifications as structural model properties. The *causal computation* approach, as mentioned before, does not generate any MSO set. Thus, diagnosis specifications must be tested by means of structural model properties. In (Krysander and Frisk, 2008), fault detectability and isolability are characterised as structural model properties. Later, in Chapter 6 such properties are extended to also cover causal detectability and isolability. Furthermore, to use Algorithm 6.8, Heuristics 1 and 2 (Section 6.6) must be fulfilled by the diagnosis specifications. Indeed, fault detectability and isolability specifications (as presented in (Krysander

and Frisk, 2008)), as well as causal detectability and isolability, fulfil these two heuristics. For other diagnosis specifications, further studies should be done. This poses serious restrictions on the diagnosis specification flexibility for the *causal computation* approach. In the *binary programming* approach, the diagnosis specifications are based on MSO set properties. However, they must be formulated as inequality constraints (non-linear or linear, depending on the BIP or BILP formulation). This limits the degree of freedom at coping with new diagnosis specifications. Finally, the *incremental* approach tests specifications by means of an external procedure P' (4.1). Therefore, diagnosis specification treatment depends on the implementation of this procedure where the required MSO sets and sensors are provided. Hence, the *incremental* and *binary programming* approaches are easier to implement for a wide range of diagnosis specifications than the *causal computation* approach.

Ranking	Generated MSO sets requirements	Sensor search efficiency	Diagnosis specifications flexibility
1	*causal computation*	*causal computation*	*incremental*
2	*incremental*	*binary programming*	*binary programming*
3	*binary programming*	*incremental*	*causal computation*

Table 8.1: Approaches comparison.

Summarising this discussion, Table 8.1 qualitatively ranks the three approaches according to the main features explained above. It can be concluded that efficiency is lower when a more flexible algorithm, with respect to the diagnosis specification requirements, is used. On the other hand, ad-hoc algorithms for specific diagnosis specifications entail better efficiency. However, they usually require more design effort. As it could be expected, when developing an algorithm to solve the sensor placement problem, there is a trade-off between flexibility and efficiency.

8.4 Future Works

Many advances on sensor placement for fault diagnosis, specially based on a structural model, have been done recently. Nevertheless, there are some open issues that need to be solved. The first issue concerns the computational complexity of the problem. Methods that work for a certain num-

ber of candidate sensors become useless when the number is scarcely increased. Therefore, algorithms with non-exponential computational complexity should be explored.

An alternative to deal with a high computational complexity is to investigate distributed diagnosis techniques. Such techniques are based on the "divide and conquer" principle, transforming an unfeasible problem into several feasible sub-problems. One possibility is to design decentralised diagnosis systems. Another possibility is to develop distributed algorithms to perform the sensor placement analysis and the MSO sets computation.

Another future extension related with the complexity is the reduction and simplification of the inequality constrains of the BIP formulation. It has been noticed that there are some constraints that can be removed since they do not affect the solution. Moreover, constraints could also be simplified by applying boolean algebra theory.

Up to now, most of the works devoted to sensor placement for fault diagnosis are focused on fault detection and isolation specifications. However, sometimes these specifications are not sufficient to design good diagnosis systems, and other diagnosis specifications must be sought. For instance, in this thesis, the computation of the unknown variables in the residual generators has been addressed. However, future works could deal with other diagnosis specifications such as disturbance decoupling, noise filtering, fault identification, robust residual generators, etc. First, studies on determining other meaningful diagnosis requirement should be carried out. Then, how they can be integrated in the sensor placement problem, by means of structural model properties if feasible, should be investigated.

Furthermore, the sensor placement for diagnosis could be extended to also cover techniques other than the ones used here. For instance, sensor placement for fault diagnosis based on either analytical or statistical models could be addressed. This may include linear and non-linear observer techniques, parameter estimation techniques or likelihood ratio test.

Another issue that can be improved is the MSO set computation. For a given a model the set of MSO sets computed is usually too large and some MSO sets must be rejected based on some MSO set properties. This can be done by combining optimisation techniques (e.g. see Section 5.5.6) and adapting the algorithm devoted to the MSO set computation (e.g. Algorithm 6.9 for causal MSO set computation). However, algorithms that directly compute, without exponential computational complexity, the set of optimal MSO sets fulfilling certain properties should be investigated. This would solve the time demanding problem of computing MSO sets.

A possible future work is the implementation or integration of the pro-

posed methodologies in a diagnosis toolbox. Since such methodologies are based on models, it is possible to automatically extract the structural model from any simulator and then perform the sensor placement analysis and the MSO set computation. If causalities were assigned in the simulator, residual generators could also be derived and implemented by the toolbox.

Finally, the developed theory could be extended to fault tolerant control. The idea is to place sensors in the process in order to improve the control capabilities when faults are present in the system. Actuator placement analysis could be also performed using similar approaches since redundancy in the actuators is a key concept in fault tolerant control.

Bibliography

J. Armengol, A. Bregon, T. Escobet, E. G. M. Krysander, M. Nyberg, X. Olive, B. Pulido, and L. Travé-Massuyès. Minimal structurally overdetermined sets for residual generation: A comparison of alternative approaches. In *Proceedings of IFAC Safeprocess'09*, pages 1480–1485, Barcelona, Spain, 2009.

M. Bagajewicz. *Design and Upgrade of Process Plant Instrumentation*. Technomic Publishers, Lancaster, PA, 2000.

M. Bagajewicz, A. Fuxman, and A. Uribe. Instrumentation network design and upgrade for process monitoring and fault detection. *AIChE J.*, 50(8): 1870–1880, Aug. 2004.

F. Barbir. *PEM Fuel Cells: Theory and Practice*. Elsevier, 2005.

M. Basseville, A. Benveniste, G. Moustakides, and A. Rougee. Optimal sensor location for detecting changes in dynamical behavior. *IEEE Transactions on Automatic Control*, 32(12):1067 – 1075, December 1987.

M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer, 2nd edition, 2006.

J. Chen and R. Patton. *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Kluwer Academic Publishers, Boston, 1999.

C. Commault, J.-M. Dion, and S. Y. Agha. Structural analysis for the sensor location problem in fault detection and isolation. *Automatica*, 44(8):2074 – 2080, 2008.

M.-O. Cordier, P. Dague, F. Levy, J. Montmain, M. Staroswiecki, and L. Trave-Massuyes. Conflicts versus analytical redundancy relations: a comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 34(5):2163–2177, 2004.

V. de Flaugergues, V. Cocquempot, M. Bayart, and M. Pengov. Structural analysis for fdi: a modified, invertibility-based canonical decomposition. In *20th International Workshop on Principles of Diagnosis (DX-09)*, Stockholm, Sweden, June 2009.

J. De Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97 – 130, 1987.

D. Düştegör, V. Cocquempot, and M. Staroswiecki. Structural analysis for residual generation: Towards implementation. In *Proceedings of the 2004 IEEE International Conference on Control Applications*, 2004.

A. L. Dulmage and N. S. Mendelsohn. Covering of bi-partite graph. *Canada J. Math*, 10:527–534, 1958.

I. EG&G Technical Services and S. A. I. Corporation. Fuel cell handbook, 2002.

T. Escobet, D. Feroldi, S. de Lira, V. Puig, J. Quevedo, J. Riera, and M. Serra. Model-based fault diagnosis in pem fuel cell systems. *Journal of Power Sources*, 192(1):216 – 223, 2009.

A. Fijany and F. Vatan. A new efficient algorithm for analyzing and optimizing the system of sensors. In *Proceedings of the 2006 IEEE Aerospace Conference*, Big Sky, Montana, USA, Mar. 4–11, 2006.

E. Frisk and M. Krysander. Sensor placement for maximum fault isolability. In *18th International Workshop on Principles of Diagnosis (DX-07)*, pages 106–113, Nashville, USA, 2007.

E. Frisk, D. Düştegör, M. Krysander, and V. Cocquempot. Improving fault isolability properties by structural analysis of faulty behavior models: application to the DAMADICS benchmark problem. In *Proceedings of IFAC Safeprocess'03*, Washington, USA, 2003.

E. Frisk, M. Krysander, and J. Åslund. Sensor placement for fault isolation in linear differential-algebraic systems. *Automatica*, 45(2):364 – 371, 2009.

E. Frisk, A. Bregon, J. Åslund, M. Krysander, B. Pulido, and G. Biswas. Diagnosability analysis considering causa interpretations for differential constraints. In *21th International Workshop on Principles of Diagnosis (DX-10)*, Portland, USA, October 2010.

E. R. Gelso, S. M. Castillo, and J. Armengol. An algorithm based on structural analysis for model-based fault diagnosis. In *Artificial Intelligence Research and Development. Frontiers in Artificial Intelligence and Aplications*, 2008.

J. Gertler. *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker, Inc., New York, 1998.

A. Ingimundarson, A. G. Stefanopoulou, and D. A. McKay. Model-based detection of hydrogen leaks in a fuel cell stack. *Control Systems Technology, IEEE Transactions on*, 16(5):1004 –1012, September 2008.

R. Isermann. *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*. Springer, 2006.

M. Khemliche, B. O. Bouamama, and H. Haffaf. Sensor placement for component diagnosability using bond-graph. *Sensors and Actuators A: Physical*, 132(2):547 – 556, 2006.

M. Krysander. *Design and Analysis of Diagnosis Systems Using Structural Analysis*. PhD thesis, Linköping Univ., Linköping, Sweden, June 2006.

M. Krysander and E. Frisk. Sensor placement for fault diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics-Part A*, 38(6):1398–1410, 2008.

M. Krysander and M. Nyberg. Structural analysis utilizing MSS sets with application to a paper plant. In *Proc. of the Thirteenth International Workshop on Principles of Diagnosis*, Semmering, Austria, May 2002.

M. Krysander, J. Åslund, and M. Nyberg. An efficient algorithm for finding minimal over-constrained sub-systems for model-based diagnosis. *IEE Transactions on Systems, Man, and Cybernetics-Part A*, 38(1), 2008.

J. Larminie and A. Dicks. *Fuel Cell Systems Explained*. Wiley, 2003.

L. Lovász and M. Plummer. *Matching Theory*. North-Holland, 1986.

P. Moraal and I. Kolmanovsky. Turbochanger modeling for automotive control applications. *SAE Paper*, 1999.

K. Murota. *Matrices and Matroids for Systems Analysis*. Springer-Verlag, 2000.

F. Nejjari, R. Sarrate, and A. Rosich. Optimal sensor placement for fuel cell system diagnosis using bilp formulation. In *18th Mediterranean Conference on Control and Automation*, pages 1296–1301, Marrakech, Morocco, 2010.

M. Nyberg. A fault isolation algorithm for the case of multiple faults and multiple fault types. In *Proceedings of IFAC Safeprocess'06*, Beijing, China, 2006.

M. Nyberg and M. Krysander. Combining AI, FDI, and statistical hypothesis-testing in a framework for diagnosis. In *Proceedings of IFAC Safeprocess'03*, Washington, USA, 2003.

R. Patton, P. Frank, and e. R. Clark. *Fault Diagnosis in Dynamic Systems Theory and Application*. Prentice Hall, New York, 1989.

S. Ploix, M. Desinde, and S. Touaf. Automatic design of detection tests in complex dynamic systems. In *Proceedings 16th IFAC Word Congress*, Prague, Czech Republic, 2005.

S. Ploix, A. A. Yassine, and J. M. Flaus. An improved algorithm for the design of testable subsystems. In *Proc. of 17th IFAC World Congress*, Seoul, Korea, 2008.

J. T. Pukrushpan. *Modeling and Control of Fuel Cell Systems and Fuel Processors*. PhD thesis, Univ. of Michigan, Ann Arbor, Michigan, 2003.

J. T. Pukrushpan, A. Stefanopoulou, and H. Peng. *Control of Fuel Cell Power Systems Principles, Modeling, Analysis and Feedback Design*. Springer, 2004.

B. Pulido and C. A. Gonzalez. Possible conflicts: a compilation technique for consistency-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 34(5):2192–2206, Oct. 2004.

R. Raghuraj, M. Bhushan, and R. Rengaswamy. Locating sensors in complex chemical plants based on fault diagnostic observability criteria. *AIChE J.*, 45(2):310–322, Feb. 1999.

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

L. A. M. Riascos, M. G. Simoes, and P. E. Miyagi. On-line fault diagnostic system for proton exchange membrane fuel cells. *Journal of Power Sources*, 175(1):419 – 429, 2008.

A. Rosich, R. Sarrate, V. Puig, and T. Escobet. Efficient optimal sensor placement for model-based FDI using and incremental algorithm. In *Proc. 46th IEEE Conference on Decision and Control*, pages 2590–2595, New Orleans, USA, December 2007.

A. Rosich, E. Frisk, J. Åslund, R. Sarrate, and F. Nejjari. Sensor placement for fault diagnosis based on casual computations. In *Proceedings of IFAC Safeprocess'09*, Barcelona, Spain, July 2009a.

A. Rosich, R. Sarrate, and F. Nejjari. Fuel cell system diagnosis based on a causal structural model. In *Proceedings of IFAC Safeprocess'09*, Barcelona, Spain, July 2009b.

A. Rosich, R. Sarrate, and F. Nejjari. Optimal sensor placement for fdi using binary integer linear programming. In *20th International Workshop on Principles of Diagnosis (DX-09)*, Stockholm, Sweden, June 2009c.

R. Sarrate, V. Puig, T. Escobet, and A. Rosich. Optimal sensor placement for model-based fault detection and isolation. In *Proc. 46th IEEE Conference on Decision and Control*, pages 2584–2589, New Orleans, USA, December 2007.

S. Spanache, T. Escobet, and L. Travé-Massuyès. Sensor placement optimisation using genetic algorithms. In *15th International Workshop on Principles of Diagnosis (DX-04)*, Carcassonne, France, June 23–25, 2004.

C. Svärd and M. Nyberg. A mixed causality approach to residual generation utilizing equation system solvers and differential-algebraic equation theory. In *19th International Workshop on Principles of Diagnosis (DX-08)*, Blue Mountains, Australia, 2008.

L. Travé-Massuyès, T. Escobet, and X. Olive. Diagnosability analysis based on component supported analytical redundancy relations. *IEE Transactions on Systems, Man, and Cybernetics-Part A*, 36(6):1146–1160, 2006.

V. Vekatasubramanian, R. Rengaswamy, K. Yin, and S. Kavuri. A review of process fault detection and diagnosis part I: Quantitative model-based methods. *Computer and Chemical Engineering*, 27:293–311, 2003.

L. A. Wosley. *Integer Programming*. John Wiley & Sons, New York, USA, 1998.

J. Wu, X. Z. Yuan, H. Wang, M. Blanco, J. J. Martin, and J. Zhang. Diagnostic tools in pem fuel cell research: Part i electrochemical techniques. *International Journal of Hydrogen Energy*, 33(6):1735 – 1746, 2008a.

J. Wu, X. Z. Yuan, H. Wang, M. Blanco, J. J. Martin, and J. Zhang. Diagnostic tools in pem fuel cell research: Part ii: Physical/chemical methods. *International Journal of Hydrogen Energy*, 33(6):1747 – 1757, 2008b.

Q. Yang, A. Aitouche, and B. Ould-Bouamama. Structural diagnosability of fuel cell stack system based on bond graph tool. In *Proceedings of IFAC Safeprocess'09*, Barcelona, Spain, 2009.

A. A. Yassine, S. Ploix, and J. M. Flaus. A method for sensor placement taking into account diagnosability criteria. *Int. J. Appl. Math. Comput. Sci.*, 18(4):497–512, 2008.