A comparative study of two structural methods for fault isolability analysis

Master's thesis
performed in **Vehicular Systems**

by
Linda Rattfält

Reg nr: LiTH-ISY-EX-3462-2004

25th February 2004

A comparative study of two structural methods for fault isolability analysis

Master's thesis

performed in **Vehicular Systems**,
Dept. of Electrical Engineering
at **Linköpings universitet**

by **Linda Rattfält**

Reg nr: LiTH-ISY-EX-3462-2004

Supervisor: **Erik Frisk, Mattias Krysander**
Linköping University
Dilek Düstegör
Lille University of Science and Technology

Examiner: **Erik Frisk**
Linköpings Universitet

Linköping, 25th February 2004

Avdelning, Institution Division, Department Vehicular Systems, Dept. of Electrical Engineering 581 83 Linköping	**Datum** Date 25th February 2004

Språk Language ☐ Svenska/Swedish ☒ Engelska/English ☐ ______	**Rapporttyp** Report category ☐ Licentiatavhandling ☒ Examensarbete ☐ C-uppsats ☐ D-uppsats ☐ Övrig rapport ☐ ______	**ISBN** — **ISRN** LITH-ISY-EX-3462-2004 **Serietitel och serienummer** Title of series, numbering **ISSN** —

URL för elektronisk version

http://www.vehicular.isy.liu.se
http://www.ep.liu.se/exjobb/isy/2004/3462/

Titel	En jämförande studie av två strukturella metoder för felisoleringsanalys
Title	A comparative study of two structural methods for fault isolability analysis
Författare Author	Linda Rattfält

Sammanfattning
Abstract

Technical systems of today are often complex and integrated. If a fault occurs, the consequences can be disastrous both for the system itself and its surroundings. To maintain the operation and the security it is necessary to have a surveillance system which can detect a fault in an early stage.

In this thesis two structural methods for fault isolation analysis are discussed. The result from the studied algorithms shows what fault isolation properties a diagnostic model is expected to have. If the isolability is not good enough, it also gives information on where further modelling needs to be done.

To base a comparison of the two structural analysis algorithms on, four criteria are defined concerning for example realizability of residuals and time complexity. One interesting part of the methods is how dynamic models are handled. It is shown how differential constraints can end up in differential cycles which implies calculatory problems and what effects structural differentiation has on a system.

The algorithms have been tested on an application from the research training network DAMADICS. The result shows how different types of input models in this case give the same result.

Nyckelord
Keywords Structural analysis, Model based diagnosis, DAMADICS

Abstract

Technical systems of today are often complex and integrated. If a fault occurs, the consequences can be disastrous both for the system itself and its surroundings. To maintain the operation and the security it is necessary to have a surveillance system which can detect a fault in an early stage.

In this thesis two structural methods for fault isolation analysis are discussed. The result from the studied algorithms shows what fault isolation properties a diagnostic model is expected to have. If the isolability is not good enough, it also gives information on where further modelling needs to be done.

To base a comparison of the two structural analysis algorithms on, four criteria are defined concerning for example realizability of residuals and time complexity. One interesting part of the methods is how dynamic models are handled. It is shown how differential constraints can end up in differential cycles which implies calculatory problems and what effects structural differentiation has on a system.

The algorithms have been tested on an application from the research training network DAMADICS. The result shows how different types of input models in this case give the same result.

Keywords: Structural analysis, Model based diagnosis, DAMADICS

Acknowledgment

First of all I would like to express my gratitude to Erik Frisk, for not only making it possible for me to perform a part of my thesis in Lille, but most of all for his guidance and friendship. In Lille I was well taken care of by my supervisor Dilek and the rest of the staff. I would also like to pass a special thanks to all of my new friends from all over the world, who made sure that I seldom was bored in my spare time.

One memory I will keep from the staff at Vehicular System is how often coffee can be appreciated and to what extent. I have felt both welcome and encouraged by the good atmosphere at the office. Especially thanks to Mattias Krysander who has helped me with all my questions regarding his algorithm and structural analysis in general.

There are also people who needs to be mentioned for taking care of me during this thesis work. My family has supported me, Klara and Jonas have shown great hospitality and Tommy has been a good friend.

Last but not least I would like to thank my opponent for his comments regarding my work.

Contents

Chapter 1

Introduction

Technical systems of today are often complex and integrated. If a fault occurs, the consequences can be disastrous both for the system itself and its surroundings. To maintain the operation and the security it is necessary to have a surveillance system which can detect a fault at an early stage. Preferably, the surveillance system is connected with a diagnosis system which can interpret the systems behavior and more precisely isolate the faulty component [**?**].

A model based method of supervision and diagnosis is based on verifying the consistency between a model of the system and measurements from the process. If they are consistent, the system is probably in a functioning mode. If they are not consistent, a fault is detected. Depending on the accuracy of the model, and its extent, a diagnosis can be made.

One disadvantage with this method is that modelling is an expensive and time consuming process and even if a detailed model is used, its fault detectability/isolability, (FDI) can be low. It would be of great advantage if the FDI properties of a diagnosis system could be estimated at an early stage and if they are not satisfactory, receiving information on where to model further and/or place additional sensors.

Structural analysis is a method which can be used to determine properties of a model such as fault detectability/isolability. Its strength lies in that it is not dependent on analytical relations, but only the *existence* of a relation between a variable and a constraint. Consider for example the well known analytical equation describing the velocity below:

$$\text{constraint, } c: velocity, v = \frac{distance, d}{time, t}$$

Its structural correspondence is:

constraint c contains the variables velocity, distance and time

or shorter:

$$c : v, d, t.$$

With this way to represent a system it is possible to investigate what to expect of a diagnosis system. How the analysis is made is going to be thoroughly explained in this thesis.

The structural analysis can be used for other purposes than fault detection/isolation. Other applications can for example be sensor placement for fault detection/isolation [**?**] and reconfigurability analysis for fault tolerant control [**?**]. This thesis will focus on structural analysis for fault detection and isolation.

1.1 Objectives

Fault detection/isolation has been studied in for example [**?**], [**?**] and [**?**]. There is not yet a standardized way to perform a structural analysis and the methods vary when it comes to handling dynamic systems for example. In this thesis two structural analysis algorithms are studied. One is developed at Department of Vehicular systems at Linköping University, Sweden and the other at Laboratoire d'Automatique et d'Informatique Industrielle de Lille, France. The objectives of this report is threefold.

- Define criteria to base a comparison of the algorithms on.
- Point out the differences between the two structural analysis methods, and to discuss their advantages and disadvantages concerning the choice of models, implementations and performance.
- Use the two methods on a real application taken from a valve in a sugar factory in Poland.

The last point is a part of a European research training network called DAMADICS [1].

[1]Development and Application of Methods for Actuator Diagnosis in Industrial Control Systems, 1/7/2000 - 30/6/2003

1.2 Contributions

The main contribution of this thesis is a discussion about the differences in the two algorithms and the criteria which the discussion is partly based upon. Furthermore a graphical user interface was implemented to fit the Lille algorithm and functions performing decoupling of faults were developed.

1.3 Outline of the thesis

The first five chapters of this thesis can be considered as an introduction to understand what structural analysis is. In Chapter 2 the foundations for structural analysis are explained and in Chapter 3 it is highlighted which desired properties such an algorithm should posses. Chapter 4 describes shortly the algorithms based on what has been discussed in Chapter 2 and in Chapter 5 they are exemplified with a small tank example. In Chapter 6, a thorough discussion is made based upon the previous chapters. This is the main contribution of my work. In Chapter 7 the two algorithms are run on an example from a valve in a sugar factory. Chapter 8 shows my programming contributions and in Chapter 9 my conclusions and suggestions for further work are presented.

Chapter 2

Introduction to structural analysis for fault isolation

Structural analysis is based on only the existence of relations between variables and equations in a model. Normally when dealing with models the variables and the equations are connected via analytical functions and parameters. In structural analysis however only the information that a variable appears in an equation is considered. This way of regarding a system permits investigations concerning the structure without the complexity that follows with analytical functions. It is therefore an efficient tool in for example a model building process where it can be used to, only by regarding the structure, determine if some parts of the model need to be modelled further. It does not however exclude the model building itself, it only gives guidelines on what you can expect from the system.

The applications of structural analysis can for example be in the process industry where the systems are complex and it is not affordable to make detailed models of the whole processes. With structural analysis you can start with an existing analytical model, extract the structure of the model, perform a fault isolation analysis and get information if there are specific parts that need to be modelled further to obtain the desired fault isolability. Another application is reconfiguration of system. A good example of this is the control of a satellite, where it is impossible to repair any hardware faults. In this case a reconfigurability algorithm can find a new optimal way to control the satellite if for example a sensor fails.

The purpose of this chapter is to give an introduction to model based fault diagnosis and to describe the main steps performed in a structural analysis. Since the main purpose of this thesis is to investigate two algorithms for fault isolability analysis, it is also explained the main ideas how to perform such an analysis.

2.1 Model Based Diagnosis

When diagnosing a system, an expected value is compared with an observed. If they do not correspond, an alarm is raised. The expected values are derived from a model of the system, and by adding information of how faults affect the system, it is possible to increase the precision of the diagnostic statement.

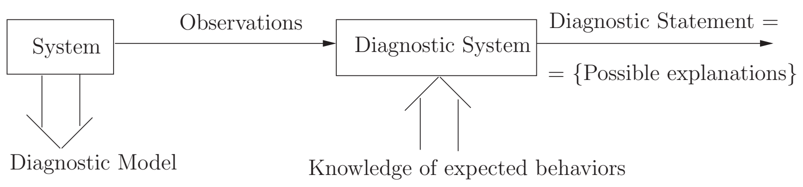


Figure 2.1: A diagnosis system and how it interacts with the real system.

In Figure 2.1, the system is described with a diagnostic model where the faults are modelled. However, when the expected behavior is calculated, it is based on a model where the faults are said to be zero. This to describe the system in a functioning mode. If the expected behavior does not match with the observations in the diagnostic system, some of the diagnostic tests in Figure 2.2 are going to raise an alarm. By using logics on which tests that have reacted and knowledge of which faults that may influence which tests, a diagnostic statement can be made.

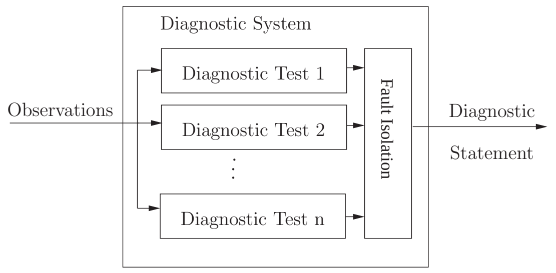


Figure 2.2: Architecture of a diagnosis system.

The choice of diagnostic tests are several, in this thesis each test consists of an Analytical Redundancy Relation, ARR (also called consistency relation or parity relation). An ARR is an equation based on only known signals such as actuator or sensor signals and it always equals zero when no faults are present and nonzero when a certain set of faults occur. The resulting signal from the ARR is called a residual. In Figure 2.3 a residual is shown. As can be seen it "equals" zero until t=5 which means that no fault is detected. At t=5 it reacts due to a fault.

By creating a number of such residuals, all sensitive to different sets of faults, a diagnostic statement can be made by using logics. An example of this is shown in the following example.

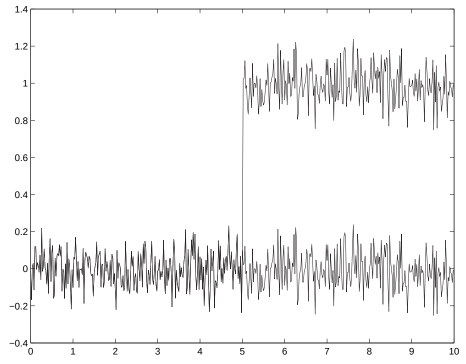
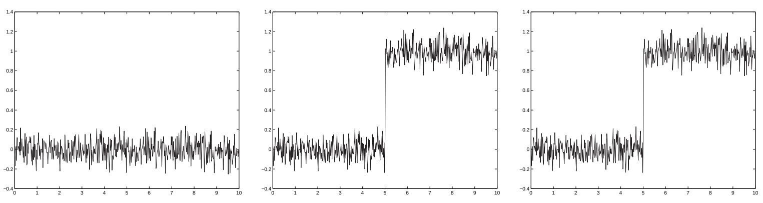


Figure 2.3: Example of two residuals, at t=5 one of the residuals respond to a fault.

Example 2.1
A diagnosis system is based on three residuals. Each residual is sensitive to the faults according to the table below:

	f_1	f_2	f_3
r_1	1	0	0
r_2	1	1	0
r_3	0	1	1

From the plots in Figure 2.4, the result can be drawn that it is fault f_2 that has occurred. The result is derived as follows: For residual r_2, f_3 cannot have caused the reaction since r_2 is not sensitive to f_3 according to the given fault sensitivity table. In the last plot f_1 cannot have caused the response in r_3. It leaves that the fault has to be f_2.



(a) Residual, r_1, sensitive to f_1. (b) Residual, r_2, sensitive to f_1 and f_2. (c) Residual, r_3, sensitive to f_2 and f_3.

Figure 2.4: Residuals.

ARR:s are discussed further in Section 2.7 and more information on model based diagnosis can be found in [**?**].

In the following part of this chapter, the general ideas and steps in a structural analysis algorithm are presented. The goal is here to analyze the fault isolability. First a structural model is needed, then the relevant data is extracted from that model in order to find a way to create ARR:s. When found, it is discussed how the ARR:s can be used together to make a fault isolation analysis. All these necessary steps are going to be explained in the remaining part of this chapter.

2.2 Structural models

In this section a structural model is going to be defined based upon an analytical model. Two ways of representing the structural system are also presented.

A system can mathematically be described by a set of equations, which in their turn consist of variables, parameters and analytical functions. The set of equations (also called constraints) are denoted $\mathcal{C}$ and the set of variables $\mathcal{Z}$. The variables can be divided in two parts, known and unknown. Examples of known variables are actuator and sensor signals. Unknown ones are internal states, noise and faults. Henceforth the known signals are denoted Y. The unknown variables are divided into state variables, X and faults F. In Example 2 a small system is used to show how the equations and the variables are parted in the different sets.

Example 2.2
Consider a small system with two states, x_1, x_2 , two sensor signals $y_1,\ y_2$, an actuator signal u and a fault f:

$$\begin{aligned} c_1 &: \ x_1 = x_2 + u \\ c_2 &: \ y_2 = x_2 + f \\ c_3 &: \ y_1 = x_1 \end{aligned}$$

With the introduced notation the different sets would contain the elements as follows:

$$\begin{aligned} \mathcal{C} &= \{c_1,\ c_2,\ c_3\} \\ \mathcal{Z} &= X \cup Y \cup F \\ X &= \{x_1,\ x_2\}, \quad Y \ = \ \{u,\ y_1,\ y_2\}, \quad F \ = \ \{f_{y_2}\} \end{aligned}$$

As said in the introduction, it is the relations between equations in $\mathcal{C}$ and variables in $\mathcal{Z}$ that are of interest. To represent these connections a matrix called the incidence matrix can be used. In the matrix the rows correspond to the equations and the columns to the variables. In Figure 2.5 the incidence matrix for the small system in the previous example is shown.

	x_1	x_2	u	y_1	y_2	f_{y_2}
c_1	1	1	1	0	0	0
c_2	0	1	0	0	1	1
c_3	1	0	0	1	0	0

Figure 2.5: Incidence matrix for the small example.

The advantage of this representation is that it is easy to base the algorithms (that are going to be presented later) on an input on this matrix form. However there are more ways to represent the structural system that are equivalent to the incidence matrix. One is by a bi-partite graph. (Further described in [**?**] and [**?**]) The definition of a bi-partite graph reads:
Definition 1 (Bi-partite graph). *A graph that consists of two disjoint sets of vertices such that every edge has one of its vertices in one set and its other in the other set, is called a bi-partite graph.*

The two disjoint vertices in our case are equations and variables. The edges between them represent the connections. In Figure 2.6 the bi-partite graph of the system in Example 2 is shown.

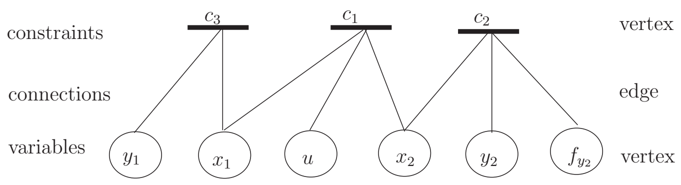


Figure 2.6: The bi-partite graph of the small example.

The advantage of using bi-partite graphs is that results from graph theory can be used. A structural model can be defined as follows using graph theory.

Definition 2 (Structural model). *The structural model of the system consisting of the set of equations $\mathcal{C}$ and the set of variables $\mathcal{Z}$ is a bi-partite graph $(\mathcal{C}, \mathcal{Z}, \mathcal{E})$ where $\mathcal{E} \subset \mathcal{C} \times \mathcal{Z}$ is the set of edges defined by:*

$$(c_i, z_j) \in \mathcal{E} \text{ if the variable } z_j \text{ appears in the constraint } c_i.$$

Note that the two representations are equivalent and the choice of which one to use depends on the application. Henceforth both of the representations are used to show different structural phenomena. In some cases both are used together to point out that they are equivalent. The structural model is what the following calculations are based upon.

2.3 Considering derivatives

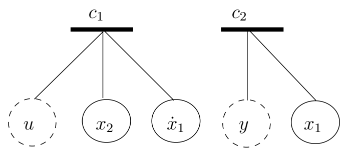
So far, the structural model has been based on a static system, but many real systems are dynamic. Therefore it is necessary to find a way to handle derivatives.

To visualize the need of knowledge of how to handle derivatives the following example is used.

$$c_1 \quad \dot{x}_1 = x_2 + u \tag{2.1}$$

$$c_2 \quad y = x_1 \tag{2.2}$$

Using the same straightforward technique as before its bi-partite graph is as follows. Here the known signals are represented with dashed lines to separate them from the unknown signals.



As can be seen in the figure, in the first part there are two unknown variables, x_2 and $\dot{x}_i$ and only one known, u. They cannot both be calculated by only one known variable. In the second part there is one known variable, y and one unknown, x_1 which can be calculated. By adding information that $\dot{x}$ is a derivative of x these parts can be combined and both of the unknown variables in the first part can be calculated. However, this can be done in different ways. The purpose of this section is to clarify that the choices of how to handle dynamic models are several and what the main differences are. Here three ways of handel dynamics are discussed:

1. The variable and its derivative are considered to be two different variables connected via differential constraint. The model is extended with differential equations like $\dot{x} = \frac{dx}{dt}$ for each differentiated variable.[**?**]
2. The variable x and its derivative $\dot{x}$ are considered to be two different variables and they are combined via structural differentiation. [**?**]
3. The variable x and its derivative $\dot{x}$ are considered to be structurally the same. [**?**]

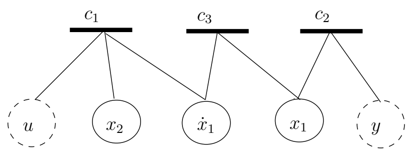
When introducing the algorithms to perform a structural analysis in Chapter 4, two different representations are for example used. Below, the three representations are discussed and it is shown how the dynamics are included in the analytical model and in the bi-partite graph.

2.3.1 Connection by differential constraint

This way to handle the derivatives always add extra differential constraints to the original model. The extended model in this case is:

$$\begin{aligned}\dot{x}_1 &= x_2 + u \\ y &= x_1 \\ \dot{x}_1 &= \frac{dx_1}{dt}\end{aligned}$$

And its corresponding bi-partite graph is:



Note that all variables can be calculated. By the second constraint x_1 can be calculated, which in its turn is used to calculate $\dot{x}_1$. The remaining unknown variable is x_2 which now can be solved by the first constraint.

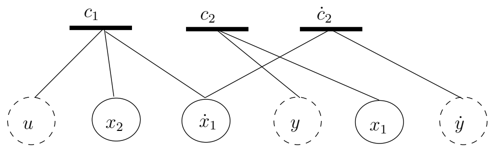
2.3.2 Structural differentiation

This approach depends on a method called structural differentiation. It connects a variable and its derivative by deriving an equation where the undifferentiated version of the variable appears in. Since structural differentiation can be made arbitrarily many times, a stop condition is introduced. This condition can be created in many ways and one of them is to limit the number of derivatives of the known signals. When the limit is reached, the structural differentiation algorithm ends.

In the example here the second equation can be derived if it is assumed that y is differentiable once. Then the system becomes:

$$\begin{aligned} c_1 \quad & \dot{x}_1 = x_2 + u \\ c_2 \quad & y = x_1 \\ \dot{c}_2 \quad & \dot{y} = \dot{x}_1 \end{aligned}$$

And the bipartite graph is:



Note that the structural differentiation added a constraint that connected $\dot{x}_1$ to a known variable, $\dot{y}$ and that x_1 is linear in c_2 and $\dot{x}_1$ in $\dot{c}_2$. To show the difference between linear and nonlinear appearances lets modify Equation (2.1) to:

$$\begin{aligned} c_1 \quad & \dot{x}_1 = x_2 + u \\ c_2 \quad & y = x_1 x_2 \end{aligned}$$

When deriving c_2, $\dot{c}_2$ will become:

$$\dot{c}_2 \quad \dot{y} = \dot{x}_1 x_2 + x_1 \dot{x}_2$$

Here, x_1 and x_2 are nonlinear in c_2 and in $\dot{c}_2$ the variables x_1, x_2, $\dot{x}_1$ and $\dot{x}_2$ are nonlinear. This can be concluded as:

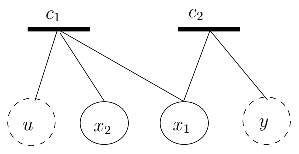
- If x is linearly contained in c then $\dot{x}$ is linearly contained in $\dot{c}$.
- If x is nonlinearly contained in c then both x and $\dot{x}$ are nonlinearly contained in $\dot{c}$.

2.3.3 Dynamic variables treated as static

The third way to model derivatives, is to handle x and $\dot{x}$ as structurally the same variable. An interpretation of this is that the values of the derivatives of x in the model can be calculated from x.

$$\begin{aligned} x_1 &= x_2 + u \\ y &= x_1 \end{aligned}$$

The bipartite graph is:



Note that both of the unknown variables can be calculated.

In the rest of this chapter, only static systems are used. For the purpose of describing the new concepts this is no limitation.

2.4 Redundancy and structure

In Section 2.1, it was said that ARR:s were used to generate residuals. The creation of ARR:s is dependent on a concept named *redundancy* which is going to be described in this section. It is also discussed how the structure of the incidence matrix is connected with this concept.

Redundancy can be described as extra information in a system and is used to verify previous results. It occurs when the number of equations are larger than the number of unknown variables, that is when the system is overdetermined. When equally many equations as the unknown variables has been used to calculate the unknown variables, it still remains equations to verify the results. To clarify this a small example is used.

Example 2.3
Consider again the small system:

$$\begin{aligned} c_1 &: \; x_1 = x_2 + u \\ c_2 &: \; y_2 = x_2 + f_{y_2} \\ c_3 &: \; y_1 = x_1 \end{aligned}$$

Since the ARR:s are used to predict the system's behavior when no fault is present, the term f_{y_2} is considered to be zero. The system is left with two unknown variables, x_1 and x_2 and three equations. Now, two of the equations are needed to calculate x_1 and x_2 and the redundancy is present with the third equation. For example, x_1 can be calculated by c_3 and x_2 by c_2. Inserted in c_1 the ARR becomes

$$y_1 - y_2 - u = 0$$

Note that only known signals are used and that it should equal zero if no fault is present.

In terms of the incidence matrix, redundancy exists only in an overdetermined system when only regarding the unknown variables and the equations. The already known variables do not need to be considered. As said before, the faults are not regarded since the use of the redundancy is creating ARR:s which in turn are used to predict the behavior of the fault free system.

2.5 Extracting the overdetermined part of a system

It was described in the previous section why redundancy is of importance in diagnosis. But, when having a structural model, is it obvious that there always exist redundancy for the whole system? The answer is no. In this section it is shown how to investigate if there is redundancy in the system and if so, how to extract that part from the rest of the system.

There are tools to extract the part of a system that contains redundancy. A method called canonical decomposition uses only row and column permutations on the incidence matrix to divide the system into three parts, where one of them contains redundancy. The three parts are:

- the underdetermined part, denoted S_-, contains more variables than equation and therefore an unambiguous do not exist. There are infinitely many solutions.
- the justdetermined part, S_0, contains as many variables as equations. All variables can be calculated unambiguously.
- the overdetermined part, S_+, contains more equations than variables. The variables can be calculated and verified.

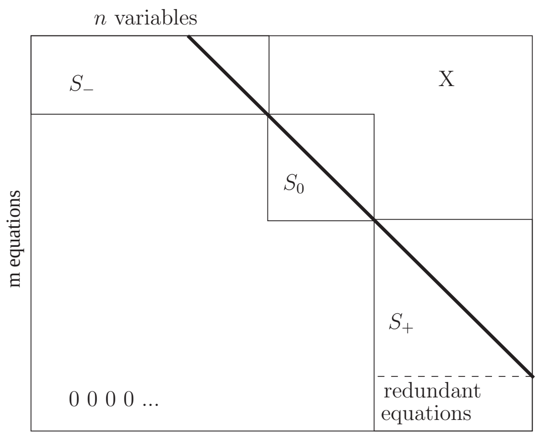


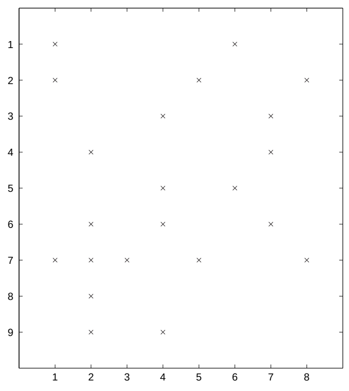
Figure 2.7: Canonical decomposition of an arbitrary system. It consists of three parts; the overdetermined S_+, the just determined S_0 and the under determined S_-

If there exists redundancy, this means that S_+ is nonzero. A schematic figure of a decomposed incidence matrix is shown in Figure 2.7. The three parts S_-, S_0 and S_+ are shown. The bold line represents that the variables can be calculated by the equations. The underdetermined part contains more variables then equations and therefore not all of the variables can be calculated unambiguously. There exist many solutions. The justdetermined part allows each variable to be calculated and so does also the overdetermined part but it has also additional equations to verify the results. Outside of the blocks there are two areas. The lower one consists only of zeros and the upper of either "0" or "1":s.

Practically the decomposition can be made in matlab by the command `dmperm`, named by the developers behind these permutations, Duhlmage and Mendelsson. Since the under- and justdetermined parts contain no redundancy and therefore are of no interest for our purpose, it is only the overdetermined part of the system that need to be considered henceforth. The goal to extract the overdetermined part is achieved! To show how a decomposition can be made, the following example is introduced.

Example 2.4
In this example a system is decomposed. In Figure 2.8(a) it is shown the initial structure of an incidence matrix. It consists of nine equations and eight unknown variables. It is hard from this representation to determine if there exists redundancy. After using the `dmperm` command

the matrix is decomposed and the new structure is shown in Figure 2.8(b). The overdetermined subsystem consists of three unknown variables and five equations, {3, 4, 6, 8, 9}. Note the resemblance with the structure in figure 2.7. It is shown that there is an overdetermined part that can be used for further analysis.


(a) Initial system.

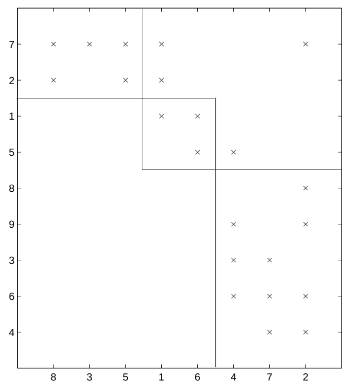

(b) Decomposed system.

Figure 2.8: Initial and decomposed system. The lower block is the overdetermined part which can be used for diagnosis.

2.6 Matchings, the first step in finding ARR:s

Now, when we possess a system with guaranteed redundancy, we can start the actual calculations to find ARR:s. The goal of this section is to find a way such that ARR:s can be created.

Lets return to the schematic figure of the overdetermined part of the decomposed system in Figure 2.7. The overdetermined part can itself be divided in two parts, a just determined part where the unknown variables can be calculated and a redundant part where the result can be tested in the redundant equations. Are the order of these equations always the same? Is it always the same equations that are used to calculate the variables and the same equations in the redundant part? The answer is no and this implies that the redundancy can be created in many different ways.

This is illustrated in Figure 2.9 where the overdetermined part of the system in Example 4 is used. In the middle it is shown two possibilities (there are altogether 17 different ways) to place the bold line which represents that the variables can be solved using the corresponding equation. This connection between all the variables and equations is called a matching. In both cases, it leaves two redundant equations. Note that the redundant equations are not the same and by using only row permutations the "original" look is obtained. The conclusion is that a system with redundancy can alter where the redundancy lies. For each one of these redundant equations, it can be shown that an ARR can be created. The number of possible ARR:s is therefore here the number of different matchings ("ways to place out the bold line") times the number of redundant equations.

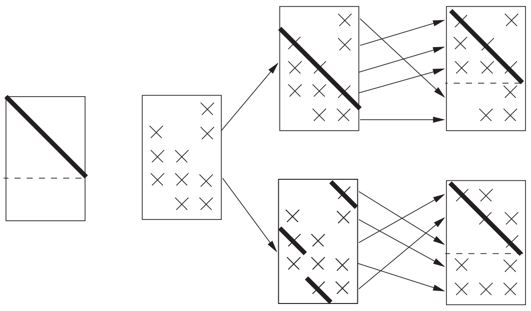


Figure 2.9: Overdetermined part from Example 4. The redundant equations can be chosen in several ways.

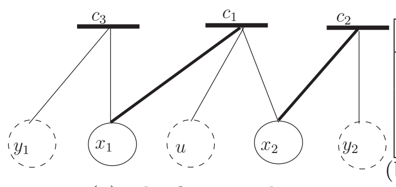
Lets return to the concept of a matching. What does it mean and what does it imply? In Section 2.4 it was said that in order to calculate the unknown variables, the number of equations has to equal the number of unknown variables. A matching is used to represent this calculability by assigning an equation to a variable. In other words, the assigned equation is used to solve the variable. It implies that in a matching the equation and the variable can only be matched once. In this thesis, complete matchings with regard of the variables are used, which means that all the variables have to be matched. A definition based on graph theory reads:

Definition 3 (Matching). *A matching in the bi-partite graph is a subset of edges such that they have no common node. If all of the vertices corresponding to the variables have edges in the matching it is said to be complete with regard to the variables.*

For the incidence matrix a matching can be characterized by circled entries. A complete matching correspond to that all variables *must* have one and only one circled element in each column. Regarding the equations, each row can have at most one circled entry. In the following example two different matchings are described with both representations.

Example 2.5

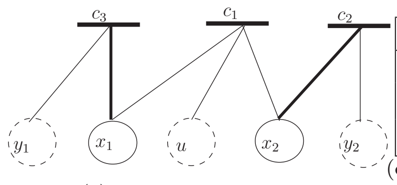
Consider again Example 2. One possible matching is shown in Figure 2.10(a). In the graph, the solid circles are the unknown variables and the dashed are known ones. In the incidence matrix the matching is represented by the bold elements. The redundancy in this case lies in c_3 where the calculated value of x_1 can be compared to the one given by the sensor signal y_1. Another possible matching is given by the graph in Figure 2.10(c). Note that the known variables do not need to be matched, their purpose here is only to make it easier to recognize the system's structure.



(a) The first matching.

	x_1	x_2	u	y_1	y_2	f_{y_2}
c_1	$\textcircled{1}$	1	1	0	0	0
c_2	0	$\textcircled{1}$	0	0	1	1
c_3	1	0	0	1	0	0

(b) Incidence matrix for the first matching.



(c) The second matching

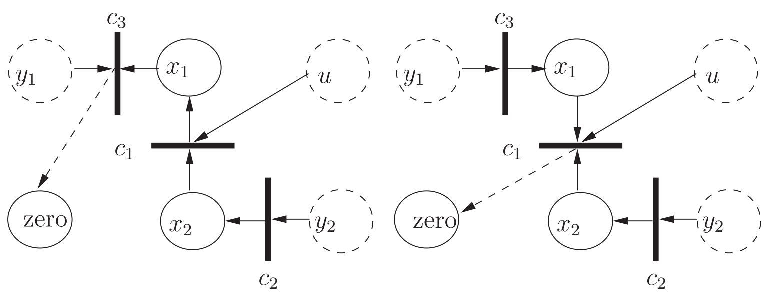
	x_1	x_2	u	y_1	y_2	f_{y_2}
c_1	1	1	1	0	0	0
c_2	0	$\textcircled{1}$	0	0	1	1
c_3	$\textcircled{1}$	0	0	1	0	0

(d) Incidence matrix for the second matching.

Figure 2.10: Two matching represented by bi-partite graphs and corresponding incidence matrices.

To represent the fact that in a matched equation all the variables except the matched one has to be known, directed edges are introduced. If an edge points at a variable it means that the variable is given by this constraint. If an edge points at a constraint it represents that the variable is needed in that constraint in order to calculate another variable. In other words, for a matching to be complete with respect to the variables, all unknown variables have one and only one edge pointing at them.

In Figure 2.11 the two previous graphs are represented as directed bi-partite graphs. The graphs have also been given another layout to make them easier to follow.



(a) Directed graph to the first matching. (b) Directed graph to the second matching.

Figure 2.11: Directed graphs.

With these directed edges, an order of calculation is introduced. It always starts with the known variables and works its way to the redundant equation. In Figure 2.11 (a) the calculations start with y_1, u and y_2 and following the directions of the edges the calculation stops at the redundant equation, here c_3, which is later used to create an ARR. The "zero-argument" represents that in the fault free case, the ARR based on that redundant equation should equal zero. In b) all the variables are known in constraint c_1 and therefore it is used when creating an ARR.

2.7 Analytical redundancy relations and MSS sets

So far in this chapter, ARR:s have been mentioned a numerous number of times. The previous steps have all been focused on the process of finding ARR:s. In this section, the focus will be on the ARR rather then the process of finding it. A new concept related to the ARR:s is also introduced. To start with, an ARR is defined as follows.
Definition 4 (Analytical Redundancy Relation). *If a system $\mathcal{C}$ contains the variables $\mathcal{X}$ and $\mathcal{Y}$, and is in a non fault state, a consistency relation, c, is a scalar equation such that* $c(y) = 0$

One might ask oneself *how* the ARR:s are used. An ARR is a relation describing how to combine known signals in a way such that they always equals zero in the fault free case and non zero when a certain set of faults occur. This is shown in the following example.

Example 2.6
Once again the system in Example 2 is used.

$$c_1 : \; x_1 = x_2 + u \tag{2.3}$$
$$c_2 : \; y_2 = x_2 + f_{y_2} \tag{2.4}$$
$$c_3 : \; y_1 = x_1 \tag{2.5}$$

Assume that x_1 and x_2 are matched with c_3 and c_2 respectively. This gives that the first equation is redundant and can be used to create an ARR. The directed graph is shown in Figure 2.10(c). The ARR is constructed by replacing x_1 and x_2 in c_1 with the sensor signals in c_2 and c_3.

$$\left.\begin{array}{ll} x_1 & = y_1 \;\; (c_3) \\ x_2 & = y_2 \;\; (c_2) \end{array}\right\} \Rightarrow$$

in equation c_1

$$y_1 = y_2 + u \Longleftrightarrow 0 = y_2 - y_1 + u$$

There exists a correspondence to ARR:s which only regards the set of equations that are needed to make an ARR. They are called Minimal Structurally Singular sets or shorter MSS sets. These sets state which equations that are needed in order to create an ARR but contain no information about *how* the equations should be used. From these sets it can also be derived which faults they are sensitive to. These definitions are taken from [**?**].

Definition 5 (Structurally Singular, SS). *A finite set of equations $\mathcal{C}$ is* structurally singular *with respect to the set of variables X if $|\mathcal{C}| > |var_{\mathcal{X}}(\mathcal{C})|$, where $|\mathcal{C}|$ is the number of elements in $\mathcal{C}$ and $|var_{\mathcal{X}}(\mathcal{C})|$ is the number of unknown variables in the set $\mathcal{C}$.*

Definition 6 (Minimal Structurally Singular, MSS). *A structurally singular set is a* minimal structurally singular *(MSS) set if none of its proper subsets are structurally singular.*

What Definition 5 says is that if a set is SS, the number of variables appearing in that set are fewer than the number of equations in that set. This implies that it is an overdetermined system. The MSS is minimal in in the sense that it is an overdetermined system with only one redundant equation. From a set that is SS it can be possible to find many different MSS:s.

For our purpose, which one of ARR:s or MSS:s that is used is of no importance. For fault isolation analysis both can be used. The choice of using ARR:s or MSS:s is a question of design of the diagnosis system and which applications that are wanted to be performed after the structural analysis. In Chapter 4 it is shown that ARR:s are a base for fault isolation for the algorithm developed in Lille, and MSS:es for the one developed in Linköping.

Example 2.7
In Figure 2.9 it was shown how matchings can be made in many ways in an overdetermined system. In this example, one of the matchings is going to be used to derive ARR;s which can be used in a diagnostic system such as the one introduced in 2.1, and to exemplify the two defined sets above. First, assume that the system in Figure 2.9 was derived from the following equations.

$$x_3 = y_3 + fy_3 \tag{2.6a}$$
$$x_1 = 2x_2 + u + f_u \tag{2.6b}$$
$$x_1 + x_2 = y_2 \tag{2.6c}$$
$$x_2 = x_3 - x_1 \tag{2.6d}$$
$$x_2 + x_3 = y_1 \tag{2.6e}$$

The system consists of three unknown variables, an actuator signal (with a possible fault) and three sensor signals, where y_3 can have an additive fault. The system is an SS system since the number of constraints (5) is larger then the number of unknown variables (3). It is not an MSS however, since it is going to be shown that there are subsystems that are structurally singular as well.

When creating the ARR:s the faults are excluded from the equations (said to be zero). However, it is kept in mind that there are two faults and in which equations they appear in, why is going to be explained later.

Consider the matching in Figure 2.9 that corresponds to the lower case. The matching is:

$$c_1 - x_3$$
$$c_3 - x_1$$
$$c_5 - x_2$$

Using these equations to express the unknown variables with only known ones, gives the equations below. On the right it is shown which equations that have been used in the calculations for each variable.

$$\begin{aligned} x_3 &= y_3 && \{1\} \\ x_2 &= y_1 - y_3 && \{1,5\} \\ x_1 &= y_2 - y_1 + y_3 && \{1,3,5\} \end{aligned}$$

The redundant equations, (2.6b) and (2.6d) can now be used to create ARR:s. Once again the equations that have been used in any form to create the relations are shown to the right.

Using Equation (2.6b), the following ARR is obtained:

$$\begin{aligned} y_2 - y_1 + y_3 &= 2y_1 - 2y_3 + u && \Longleftrightarrow \\ 0 &= 3y_1 - 3y_3 - y_2 + u && \{1,2,3,5\} \end{aligned}$$

Using Equation (2.6d), the following ARR is obtained:

$$\begin{aligned} y_1 - y_3 &= y_3 - y_2 - y_1 + y_3 && \Longleftrightarrow \\ 0 &= 2y_1 + y_2 - 3y_3 && \{1,3,4,5\} \end{aligned}$$

The ARR:s are found, but it still remains to say which fault sensitivity they have before they can be used in a diagnosis system. For a fault to be able to make an ARR to react, it has to be included in one or more of the equations that have been used to create that ARR. In this case there are two faults, in Equation (2.6a) and in Equation (2.6b), respectively. As can be seen in the first ARR, both of these equations have been used and therefore the ARR can react on both of the faults. The second ARR can only react on f_{y_3} since Equation (2.6b) is not included in the set.

Lets consider the sets of equations needed to create the ARR:s. In both cases the sets consist of four equations used to solve three unknown variables and to create an ARR. They are both SS, but also MSS sets since no subset of these sets are SS.

2.8 Fault isolation analysis

Structural analysis can be used in different applications. This report is focused on structural analysis used to design diagnosis systems to isolate faults. In this section it is going to be described how the ARR:s can be used to analyze fault isolability possibilities in a system.

When an ARR is constructed, it is assumed that the system is in a fault free state. But what happens if a fault occurs? Previous sections have shown that an ARR is constructed from a redundant equation, and equations used to solve the unknown variables. Each of these equations can be sensitive to a number of faults, according to the structure of the model. Together they make a set of faults that the ARR is sensitive to. When any of the faults in the set occur, the ARR *can*, but will not necessarily *have to* react. One reason why a fault does not make the ARR react can be that the fault is so small that it is covered by noise. On the other hand, it is for certain that the ARR cannot react on those faults not included in the fault sensitivity set.

To represent the fault sensitivity for each ARR, a sensitivity matrix is introduced. The rows represents the ARR:s and the columns all possible faults. A "0" represents that the ARR is unsensitive to that fault and a "1" that the ARR can detect that fault.

The isolation analysis is made possible when the knowledge of fault sensitivities are put together from many ARR:s. Since the ARR:s can be sensitive to different faults, combinations of ARR:s can give information on the isolability. This is done by analyzing which faults that can be explained by other faults in the fault sensitivity matrix. It is illustrated by an example.

	f_1	f_2	f_3
ARR 1	0	1	0
ARR 2	0	1	1
ARR 3	1	0	1
ARR 4	1	1	1

Table 2.1: Fault sensitivity matrix.

Consider the fault sensitivity matrix in Table 2.1. There are four ARR:s and three possible faults. In this example three cases are going to be studied, corresponding to each of the faults occurrences. Lets start with fault f_1, then ARR:s 3 and 4 can react. ARR:s 3 and 4 can also be caused by fault f_3. Therefore, if fault f_1 is present, it cannot be distinguished from f_3. Remember that a "1" in the matrix does not have to imply that the fault always will make the ARR react. This

	f_1	f_2	f_3
f_1	1	0	1
f_2	0	1	0
f_3	0	0	1

Table 2.2: Fault matrix.

is why the ”1” for ARR 2 and f_3 does not matter. Fault 2 on the other hand has a ”0” for ARR 3 and cannot explain the reaction of that ARR. The result is presented in a fault incidence matrix that is a square matrix with the same dimension as the number of possible faults. A row correspond to a present fault and the columns which other faults that possibly can explain the present fault.

If fault f_2 is present, ARR:s 1, 2 and 4 can react. Since ARR 1 only is sensitive to fault f_2 a possible reaction cannot be due to any of the other faults and therefore it is possible to isolate fault number two.

The same goes for fault f_3 which can be isolated since ARR 2 and ARR 3 cannot be explained by f_1 and f_2, respectively. The resulting fault incidence matrix is presented in Table 2.8.

The results of the algorithms are going to be presented in this form, as fault incidence matrices. Note that the same method goes for MSS:es. The only difference is that the fault sensitivities are based on the faults occurring in the equations in an MSS set.

Chapter 3

Desirable goals for a structural analysis algorithm

To make it easier to compare and discuss the two structural analysis algorithms presented in Chapter 4, four points which are desirable for such algorithms are going to be presented in this chapter. The points are modified from the general properties for diagnostic systems described in [**?**] to fit structural analysis in particular.

3.1 Realizable solutions

The first point concerns wether the found ARR:s/MSS:es are realizable or not. With *realizable* it is meant that given a set of equations needed to create an ARR or an MSS, the resulting analytical expression should be solvable. This is not evident since first of all the structural model is a simplification of the analytical model itself. Secondly, some problems occur due to the fact that numerical calculations disturbed by noise and integrations are hard to perform accurately.

Another example that can cause problems is differential constraint such as $\dot{x} = \frac{dx}{dt}$ where $\dot{x}$ can be calculated from x, but not the other way around. To perform the integration the initial value has to be known, and this is not always the case. The same type of problem also occurs with injective functions, which can be considered as "one-way-relations" as well. An example of an injective function is the square function:

$$\begin{aligned} x^2 &= u \\ y &= x \end{aligned}$$

To verify the sensor signal y, x has to be matched by the first equation which gives:

$$x = \pm\sqrt{u}$$

Since x can be either positive or negative it cannot be matched here. If the situation is the opposite; that the second equation should be used to match x, no problem occurs since $x = y$ gives an unambiguous answer. In order to realize the ARR:s the injectivity therefore should be handled.

3.2 Ability to handle different type of systems

Even though it is often assumed that systems are linear to simplify calculations, this is not always suitable. The method should be compatible for nonlinear systems in order to be applicable to a larger set of systems.

Two different types of systems are static and dynamic ones. Static systems are much simpler to handle since it does not contain any derivatives. However, in real applications systems are generally dynamic, in other word the equations describing the system contain differentiated variables. There are different ways to handle the dynamic aspects of the system, three ways to model it was described in Section 2.3 and their influence on the algorithms will be presented in Section 6.1.1. Generally it is important that the dynamics are handled in a correct manner.

3.3 Time complexity

This is a constraint which deals with the implementation of the algorithm. However this is not the most important criterium since this step is only made once in the beginning of the modelling and not online. Normally, the analysis is made in advance in order to determine the isolability. But nevertheless, if the models are extensive, the search operations through the large matrices take a considerably amount of time. Cut down on operations and try to eliminate unnecessary information is therefore of importance.

3.4 Completeness

This is one of the most important points. If many different ARR:s/MSS:s are found, it is more likely that they all together contain more information that can be used for the fault analysis. For the fault isolability application this mean that there exist more sets that are sensitive to different sets of faults. What decides how many MSS:s the algorithms find is how they handle the information in the system. The main goal should be to use the initial systems information efficiently and to find all possible ARR:s/MSS:s.

Chapter 4

Two structural analysis algorithms for fault isolation

In this chapter, two structural analysis methods are going to be described based on the theory presented in Chapter 2. The algorithms are developed at Department of Vehicular systems at Linköping University, Sweden and at Laboratoire d'Automatique et d'Informatique Industrielle de Lille, France. To get an overview of what steps they consist of, a block diagram is presented in Figure 4.1. As can be seen some of the steps are in common, while others differ.

Please note the similarity between the block diagram and what was presented in Chapter 2.

4.1 The Lille algorithm

In this section the different steps of the Lille algorithm are described. One of the purposes of the algorithm is of course to analyze the fault detection/isolation properties of a system by creating realizable ARR:s, but also reconfigurability properties have been studied. For a more detailed description of the algorithm see [**?**].

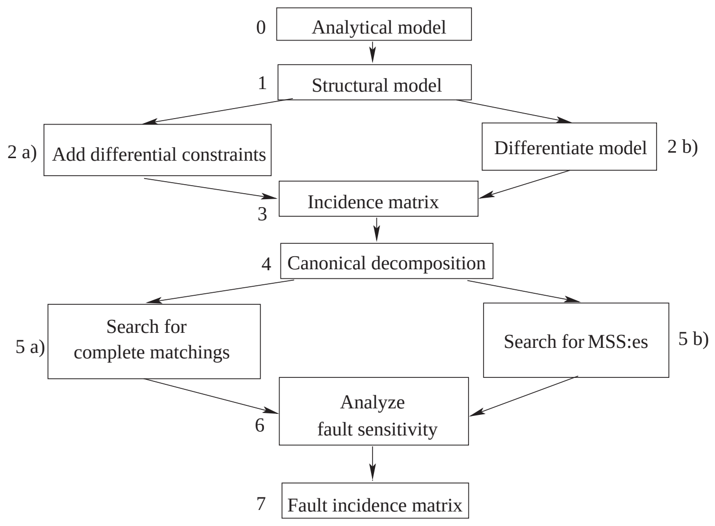


Figure 4.1: Schematic overview of the two methods. The path to the left corresponds to the Lille algorithm and the right to the one developed in Linköping.

4.1.1 Structural model

The structural model which is used here, is derived from an analytical model in the same way as described in Section 2.2. Using the incidence matrix approach, all of the equations get its own row where the corresponding variables presences are marked. A differentiated variable and its original variable are considered to be two separate variables.

4.1.2 Addition of differential constraints

The first visible difference between the methods is how they consider the relationship between a variable and its derivative. The Lille method consider that they are two separate variables connected via a differential constraint as described in Point 1 in Section 2.3. Therefore, an extra constraint for each derived variable is added to show that there is a connection between the variable and its derivative.

In this step, there is a difference between the theoretical algorithm and the implemented version which is going to be used in Chapter 5 on a small tank example. The difference lies in how the differential constraint is represented in the incidence matrix. Consider the two incidence matrices in Figure 4.2.

	x_1	$\dot{x}_1$	x_2	$\dot{x}_2$	$\dots$
c_1	$\dots$				
$\vdots$	$\vdots\ \ddots$				
c_n					
	1	1	0	0	
	0	0	1	1	

(a) Implemented version of incidence matrix.

	x_1	$\dot{x}_1$	x_2	$\dot{x}_2$	$\dots$
c_1	$\dots$				
$\vdots$	$\vdots\ \ddots$				
c_n					
	Δ	1	0	0	
	0	0	Δ	1	

(b) Theoretical version of the incidence matrix.

Figure 4.2: Two incidence matrices with differential constraints. The entry Δ represents that the variable cannot be matched by that constraint.

In the theoretical version, x is prevented of being calculated from $\dot{x}$, which is represented by the Δ entries. The motivation behind this is that if x is matched with $\dot{x}$, when calculating the ARR, an integration has to be made. Since the initial value is not known, the answer would be incomplete. It is therefore avoided in an early stage to make such assumptions.

4.1.3 Decomposition of the incidence matrix

As described in Section 2.5, Step 4 in the algorithm decomposes the system. The input is the incidence matrix from the extended system (with differential constraints) and the outcome is the overdetermined subsystem.

4.1.4 Search for complete matchings

The Lille method uses matchings as a base to derive ARR:s. A recursive matching algorithm is used in order to find all possible matchings. In the implemented version, all variables can be matched as described in Section 2.6. In the theoretical algorithm the matchings are limited because of the Δ entries in the incidence matrix.

The result is a matrix where each row is a matching and the columns are the different variables. The values in the matrix correspond to which equation each variable is matched with. Since the matchings are complete with respect to the variables, all of the entries are nonzero in the matrix.

4.1.5 Find possible ARR:s and faults affecting them

When all of the variables have been matched, the next step is to analyze how to use the redundant information in the system. Although the algorithm itself does not create ARR:s it gives information about how to create them, or more precisely which equations that are needed to do so.

For each matching, each redundant equation (those not included in the matching), can be used to create an ARR. It is made by, for each redundant equation, substituting the variables in that expression by an expression from where the variable have been matched. The result is a set of equations that are needed to create an ARR. To this matrix belongs a fault sensitivity matrix. It states which faults each ARR set of equations is sensitive to.

4.1.6 Create the fault incidence matrix

The last part of the algorithm is to calculate which faults that are detectable and isolable from each other. This is done as described in Section 2.8.

4.2 The Linköping Algorithm

This algorithm is developed in Linköping and its purpose is to analyze isolability via residuals. The main idea is to find all possible MSS:s and then find a small set of the MSS:s with the same isolability properties as all of them. Then the MSS:es are used to create residuals. The last step concerning the residual generation is not discussed further in this report. The details of the algorithm can be found in [**?**].

4.2.1 Structural model

The input to the algorithm is a structural model. In the model the variables are stored with what kind of appearance they have (linear or nonlinear) in the equations. It is also specified that if a variable is known, how many of its derivatives it is possible to derive from this variable. This knowledge is used in the structural differentiation step later. The result can be considered as an incidence matrix with extra information about the variables.

4.2.2 Structural differentiation

Structural differentiation is made on the system until all the variables and their derivatives have been "connected" as described in Section 4.2.2.

It is important to note, that the additional information which is needed for the structural differentiation does not *exclude* the possibility to use the algorithm in a manner that handles dynamics differently. The algorithm permits all three approaches described in Section 2.3. Structural differentiation is however the one that is unique for this algorithm.

4.2.3 Simplification

Altogether two simplifications can be made. First, the variables that are impossible to eliminate and find redundancy for are excluded via canonical decomposition. This is the case for variables that appears in the under and just determined part of the system.

The next simplification is to merge equations that have to be used together to eliminate a variable. In an MSS set these two equations will always appear together. The simplest form of merging appears when a unknown variable exists in exactly two equations. After the merge a search is performed once again, but this time with the set of equations as a unit. It is possible that another equation is merged into the first set. The importance of this step is to reduce the complexity when searching for MSS sets later. This simplification has not been implemented in the version of the implemented algorithm that has been used.

4.2.4 Search for MSS sets

The MSS algorithm searches through the incidence matrix. It selects an equation, and then tries to find a matching which makes the selected equation a base for an ARR. After each new variable have been matched, a condition clause verifies if the set of equations is a one-overdetermined system, MSS, due to the variables or not. The result is a list with MSS:es and a fault sensitivity matrix

4.2.5 Create the fault incidence matrix

The incidence matrix is created according to what was described in Section 2.8 and is based on the fault sensitivity matrix.

Chapter 5

The algorithms presented with a small tank example

In this chapter the two methods are demonstrated with a small tank example. First, the tank example will be described and then the algorithms are used to perform a fault isolability analysis upon it. The purpose of this chapter is to show practically how the algorithms work.

5.1 Description of the tank example

To be able to calculate by hand how the algorithms work, this example is small to keep the complexity down. The example originates from [**?**] with the modification that an extra sensor has been added. The system consists of a tank, two sensors, an actuator and a control system as shown in Figure 5.1

Before the analytical relations for this tank model are shown, lets reason about what the structure of this system looks like. In the tank, there are three interesting things that happen. First water flows into the tank, which implies that there has to be a relation between the inflow q_i and the actuator signal u. Secondly, the hight of the water level changes. The variables that can affect the hight of the water are the inflow and the outflow. Thirdly, there is an outflow of the tank that should be affected by the hight of the water level. According to the figure there are also two sensor signals connected to h and q_i respectively. The

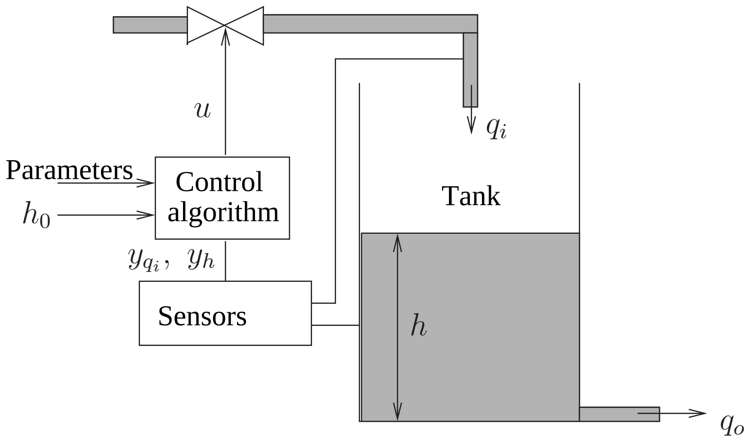


Figure 5.1: The small tank example.

incidence matrix from this reasoning is:

	$u(t)$	$y_h(t)$	$y_i(t)$	$h(t)$	$\dot{h}(t)$	$q_i(t)$	q_o
c_1	0	0	0	0	1	1	1
c_2	1	0	0	0	0	1	0
c_3	0	0	0	1	0	0	1
c_4	0	1	0	1	0	0	0
c_5	0	0	1	0	0	1	0

An incidence matrix has now been derived without involving any analytical expressions. To show that the result above are the same as those derived from the analytical model, the systems equations are shown in Equations 5.1.

$$c_1 : \ \dot{h}(t) = q_i(t) - q_o(t) \tag{5.1a}$$
$$c_2 : \ q_i(t) = \alpha u(t) \tag{5.1b}$$
$$c_3 : \ q_o(t) = k\sqrt{h(t)} \tag{5.1c}$$
$$c_4 : \ y_h(t) = h(t) \tag{5.1d}$$
$$c_5 : \ y_{q_i}(t) = q_i(t) \tag{5.1e}$$

The faults we would like to detect and isolate are an additive fault in the actuator, f_u, and two multiplicative faults in the level sensor and in the output pipe, f_{y_h} and f_{q_o} respectively. The faults are added to the model.

$$c_1: \; \dot{h}(t) = q_i(t) - q_o(t) f_{q_o}(t) \tag{5.2a}$$
$$c_2: \; q_i(t) = \alpha(u(t) + f_u) \tag{5.2b}$$
$$c_3: \; q_o(t) f_{q_o}(t) = k\sqrt{h(t)} \tag{5.2c}$$
$$c_4: \; y_h(t) = h(t) + f_{y_h}(t) \tag{5.2d}$$
$$c_5: \; y_{q_i}(t) = q_i(t) \tag{5.2e}$$

As said before, for structural analysis purpose the importance is not the analytical relations, but only structural information. The relations between the known and unknown variables and the constraints are concluded in the bi-partite graph in Figure 5.2. This representation is useful to visualize the structure.

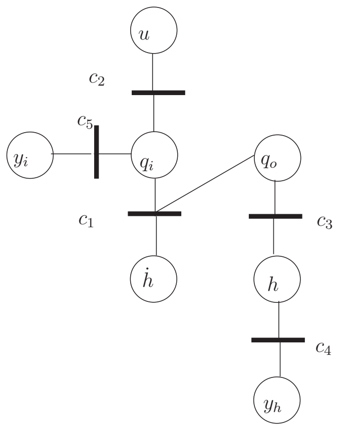


Figure 5.2: Bi-partite graph of the small tank system.

When implemented in an algorithm it is however more appropriate to use the incidence matrix representation. The graph and the incidence matrix are equivalent and how you like to represent the system is a question of application. The incidence matrix with the faults included is shown below. It is this matrix that serves as input to the algorithms.

5.2 The Lille algorithm

In this section the different steps of the Lille algorithm are described. Please compare the steps here with the description of the algorithm in Chapter 4 and the theory presented in Chapter 2.

	$u(t)$	$y_h(t)$	$y_i(t)$	$h(t)$	$\dot{h}(t)$	$q_i(t)$	q_o	f_u	f_y	f_{q_o}
c_1	0	0	0	0	1	1	1	0	0	1
c_2	1	0	0	0	0	1	0	1	0	0
c_3	0	0	0	1	0	0	1	0	0	1
c_4	0	1	0	1	0	0	0	0	1	0
c_5	0	0	1	0	0	1	0	0	0	0

Table 5.1: Incidence matrix over the known input and outputs, the unknown internal states and the faults. A '1' in a position means that the variable in that column appears in the corresponding constraint.

5.2.1 Structural model

The structural model of the tank for the Lille algorithm is made by adding a sixth constraint to the ones in Table 5.1 which represents the connection between h and $\dot{h}$. The expanded matrix is shown in Table 5.2.

	$u(t)$	$y_h(t)$	$y_i(t)$	$h(t)$	$\dot{h}(t)$	$q_i(t)$	q_o	f_u	f_y	f_{q_o}
c_1	0	0	0	0	1	1	1	0	0	1
c_2	1	0	0	0	0	1	0	1	0	0
c_3	0	0	0	1	0	0	1	0	0	1
c_4	0	1	0	1	0	0	0	0	1	0
c_5	0	0	1	0	0	1	0	0	0	0
c_6	0	0	0	1	1	0	0	0	0	0

Table 5.2: Incidence matrix over the tank example. Included is also the sixth differential constraint.

Theoretically, the sixth constraint should consist of a Δ value for h and therefore it would not be matchable in that constraint.

5.2.2 Decomposition of the tank example

As described in Section 2.5, the first step in the algorithm is to try to decompose the system further. In the following section the tank data (including the sixth constraint) is fed to a decomposition routine.

In our tank example the initial system is already overdetermined. This can be verified with Dulmage-Mendelson - permutations. In Matlab this is done by the `dmperm` command. The output is the matrix

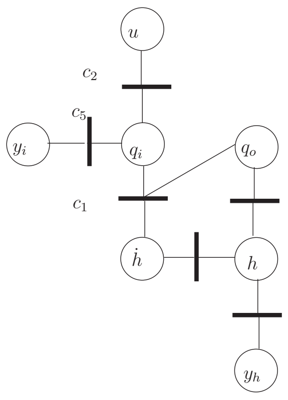


Figure 5.3: Bi-partite graph of the tank example including the differential constraint.

below. Note that the rows and columns in the matrix have been permutated.

	h	q_o	q_i	$\dot{h}$
c_3	1	1	0	0
c_4	1	0	0	0
c_1	0	1	1	1
c_2	0	0	1	0
c_5	0	0	1	0
c_6	1	0	0	1

Table 5.3: Permutated matrix of the internal states. The matrix is overdetermined because there are more equations than variables.

The permutated matrix in Table 5.3 serves as input to the matching algorithm.

5.2.3 Find all matchings

The Lille matching algorithm is a recursive algorithm which searches the incidence matrix systematically for all possible matchings. As said before, a matching is a 'connection' between all variables and some or all of the equations, such that the matched variable is the only unknown variable in that equation. Each variable and equation can only be matched once in every turn. Below the match algorithm is applied to the tank example.

- Consider the first internal state, h, in the permutated matrix in Table 5.3. The first equation it appears in, is in c_3. Lets assume that h is matched with c_3. Since h and c_3 now are matched, the corresponding column and row in the matrix are temporarily neglected in the rest of the matching process.
- Consider now the second state, q_o, the first equation it appears in (since c_3 has already been matched) is in c_1 and hence they are matched.
- Using the same technique, q_i and $\dot{h}$ are matched with c_2 and c_6 respectively. The match is shown in Figure 5.4. The match is represented with the bold edges.

	h	q_o	q_i	$\dot{h}$
c_3	$\textcircled{1}$	1	0	0
c_4	1	0	0	0
c_1	0	$\textcircled{1}$	1	1
c_2	0	0	$\textcircled{1}$	0
c_5	0	0	1	0
c_6	1	0	0	$\textcircled{1}$

Table 5.4: Permutated matrix of the internal states. The circled entries show the matching.

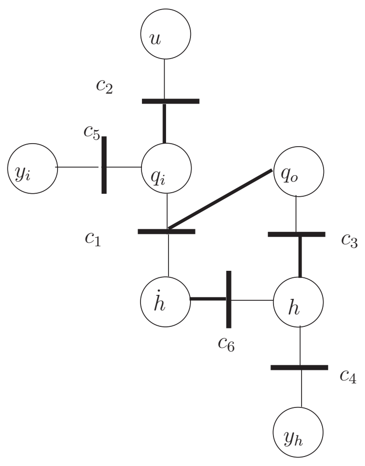


Figure 5.4: The first complete matching. The bold edges correspond to the matching.

The matching above is not the only one. For example h, q_o and $\dot{h}$ can be matched the same way, but q_i is matched with c_5 instead of c_2. In total there exist 11 different possible matchings in this example.

The output from the matching algorithm is a matrix where the rows correspond to the matchings, the columns correspond to the variables and the values in the matrix tell with which equation the variable has been matched.

5.2.4 Find possible ARR:s and faults affecting them

When all of the variables have been matched, the next step is to analyze how to use the redundant information in the system. In our tank example there are six equations and four variables. This permits that for every matching two ARR:s can be found based on the redundant equations.

Although the algorithm itself does not create ARR:s it gives information about how to create them, or more precisely which equations that are involved in creating it. The next example will try to clarify how the ARR sets are derived and how they are used together to isolate faults.

Consider the matching

$$\begin{aligned} &c_1 - q_o \\ &c_3 - h \\ &c_5 - q_i \\ &c_6 - \dot{h} \end{aligned}$$

The redundant equations are c_2 and c_4.

First, lets see which equations that are needed to create an ARR based on constraint c_2. The set of equations needed is denoted

$$\{c_2\}$$

In c_2 an unknown variable, q_i, is introduced. From this we get that we need to include at least the equation that matches q_i in the set as well. From the given information we see that q_i is matched by c_5. The set is now

$$\{c_2, c_5\}$$

On the other hand, c_5 does not introduce any other new variable and therefore we have found the first set of equations to create an ARR, $\{c_2, c_5\}$. The fault sensitivity to this set is $\{f_u\}$ since f_u appears in c_2.

For c_4 the corresponding set is derived as follows.

$$\{c_4\}$$

In c_4, the variable h occurs and has to be matched by c_3. The set is now:

$$\{c_4, c_3\}$$

In c_3, the variable q_o occurs and therefor c_1 has to be included as well.

$$\{c_4, c_3, c_1\}$$

In c_1, both q_i and $\dot{h}$ are new included variables. To be able to calculate these the equations c_5 and c_6 has to be included as well. Note that none of the constraints introduce any new variables, since h already has been matched. The resulting set of equations to create an ARR from c_4 is therefore:

$$\{c_4, c_3, c_1, c_5, c_6\}$$

The fault sensitivity of this set is $\{f_{q_o}, f_y\}$.

Even though many ARR:s can be found, some of them are made of the same set of equations. If so, the fault sensitivity are the same and it does not contribute to the fault isolability. Therefore, unique sets are wanted. In this case, there exists another unique set which can be used for an ARR. It is $\{c_1, c_2, c_3, c_4, c_6\}$, which origins for example from the matching

$$\begin{aligned} c_1 &- q_o \\ c_2 &- q_i \\ c_3 &- h \\ c_6 &- \dot{h} \end{aligned}$$

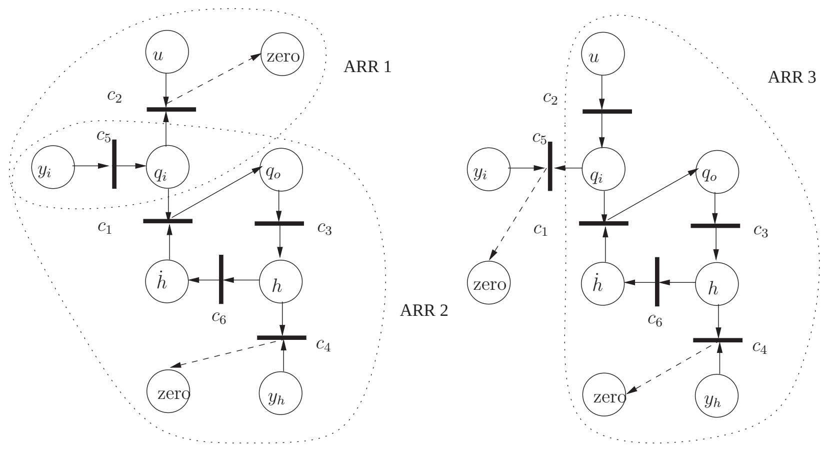
and its fault sensitivity is $\{f_{q_o}, f_u, f_y\}$

The ARR:s are shown in Figures 5.5(a) and 5.5(b) with the graph representation. When a matching has been made it implicates that an order of calculation is set as can be seen by the directed edges in the figure.

The fault sensitivity for the graph derived ARR:s are made in the same way as before, by seeing which faults that occur in the equations in each path.

5.2.5 Create the fault incidence matrix

The last part of the algorithm is to calculate which faults that are detectable and isolable from each other. This is done by comparing the fault sensitivities of the ARR:s.



(a) First directed graph. (b) Second directed graph.

Figure 5.5: Two directed graphs. Within the dotted lines one path each can be found which correspond to an ARR.

From above three unique sets have been obtained, $\{c_2, c_5\}$, $\{c_1, c_3, c_4, c_5, c_6\}$ and $\{c_1, c_2, c_3, c_4, c_6\}$. The first set is sensitive to f_u, the second to f_h and f_{q_o} and the third to f_u, f_h and f_{q_o}, as described in Table 5.5. By combining them the isolability possibilities can be investigated.

ARR	f_u	f_y	f_{q_o}
1	1	0	0
2	0	1	1
3	1	1	1

Table 5.5: Fault signature for the first four matchings in the tank example.

By using the technique described in Section 2.8, the fault incidence is calculated. It is shown in Table 5.6.

What the matrix says, is that if there is an actuator fault, it can be detected and isolated. If some of the other faults occur (sensor fault or clog in the output pipe), they can be detected, but not isolated from each other.

	f_u	f_y	f_{q_o}
f_u	1	0	0
f_y	0	1	1
f_{q_o}	0	1	1

Table 5.6: Fault incidence matrix for the tank system.

5.3 The Linköping algorithm

In this section the different steps of the Linköping algorithm are described. Once again, use the previous chapters to recognize the different steps and their importance in the analysis.

5.3.1 Structural model

The input is a structural model with extra knowledge about if the variables are linear or not. In the structural model $\dot{h}$ is modelled by h with the extra knowledge of that it is derived.

	h	$\dot{h}$	q_i	q_0
c_1	0	1	1	1
c_2	0	0	1	0
c_3	1	0	0	1
c_4	1	0	0	0
c_5	0	0	1	0

Table 5.7: Input to the Linköping algorithm for the tank algorithm. The matrix comes from Table 5.1 with the modification that the information of $\dot{h}$ is stored in h.

5.3.2 Differentiation

In order to create or expand the redundancy in the system it can be motivated to differentiate some equations. This can for example be the case if a derived variable appears in only one equation in the original model.

The result of this first differential step is an extended model consisting of the original model and the new derived equations and variables. When the known variables are permitted to be derived once each, the extended model gets four extra constraints, and if they are permitted to be derived twice, the extended model consists of 13 equations. Since it is too hard to verify the results by hand with this complexity, it is here instead demonstrated third approach discussed in Section 2.3. The incidence matrix with this method is:

	h	q_i	q_0
c_1	1	1	1
c_2	0	1	0
c_3	1	0	1
c_4	1	0	0
c_5	0	1	0

5.3.3 Simplification

The simplification implemented is to extract the overdetermined part of the system. In this case the system is already overdetermined. The other simplification step merges equations that need to be used together. Even though it is not implemented it here could be shown that c_1, c_3, c_4 would create such a group. This since c_1 is dependent on c_4 and c_3 and the other way around to solve h and q_i respectively.

5.3.4 Search for MSS sets

What is done now is very similar to the matching described for the Lille algorithm. The difference is that after each step when a match has been done, the algorithm check if the equations in the matching so far is an MSS set or not.

The MSS:s obtained are $\{c_1, c_2, c_3, c_4\}, \{c_1, c_3, c_4, c_5\}$ and $\{c_2, c_5\}$. The calculations contain many details and is summarized in a table along with the MSS algorithm itself in Appendix A. The sets are sensitive to $\{f_u, f_y, f_{q0}\}, \{f_y, f_{q0}\}$ and $\{f_y, f_{q0}\}$ respectively. Using the same technique as described in Section 2.8, we see that the results are the same and the fault incidence matrix is shown in table 5.8.

	f_u	f_y	f_{q_o}
f_u	1	0	0
f_y	0	1	1
f_{q_o}	0	1	1

Table 5.8: Fault incidence matrix for the tank system.

5.4 Results

In this case the results are the same. The found MSS are also the same as shown in Sections 5.2.4 and 5.3.4 with the addition in the later case that also the differential constraint is a part of the MSS. This does not effect the fault incidence matrix though since no other fault is included by that equation.

Regarding the time aspects the differences are not that different in this small example. For both of the algorithms the system is solved in a couple of tenths of a second.

Chapter 6

Discussion concerning the algorithms

In this chapter the two algorithms are discussed from different point of views. It starts with a section concerning the main differences between the methods where the advantages and disadvantages are clarified. The next section contains a discussion of the algorithm properties regarding the four points given in Chapter 3. The last section is a discussion of how to improve the isolability properties for both of the algorithms.

6.1 Main Differences

In Chapter 5 the two methods were explained and illustrated with a small tank example. Already in that chapter some of the differences were obvious. In this section the most important differences are discussed further.

6.1.1 Handling of dynamic models

Both of the algorithms are considered as dynamic since both of them can analyze dynamic models. However, as mentioned in the previous chapters, the way of creating and handling dynamic relations differ.

Addition of differential constraints

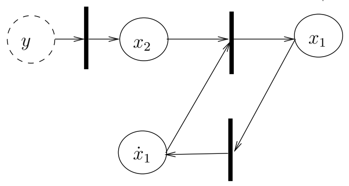
In the Lille algorithm differential constraints are added between the differentiated and undifferentiated variable. This approach is straightforward and is easily implemented. In theory the fact that variables can not be matched with its derivatives is handled, but not in in the implemented versions.

However the new differential constraint introduces a problem which has not yet been discussed; differential cycles. A differential cycle occurs when a set of variables are calculated in a "loop" and one of the variables is differentiated.

Example 6.1
This example will show the difference between an algebraic cycle and a differential one. Consider the following system:

$$\begin{aligned} \dot{x}_1 &= x_1 - x_2 \\ x_2 &= y \\ \dot{x}_1 &= \frac{dx_1}{dt} \end{aligned}$$

The system has three unknown variables and three equations. Its bipartite graph with one matching represented by the directed edges is:



The cycle consists of x_1 and $\dot{x}_1$. Starting to solve for x_1 gives the following equation:

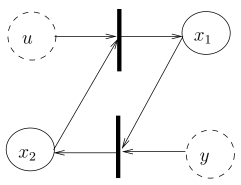
$$\begin{aligned} x_1 &= \dot{x}_1 + y \Rightarrow \\ x_1 &= \frac{dx_1}{dt} + y \Rightarrow \\ 0 &= \frac{dx_1}{dt} - x_1 + y \end{aligned}$$

The solution is $x_1 = y + Ce^t$. The constant, C, is not known and therefore x_1 cannot be calculated unambiguously.

Consider now the system:

$$\begin{aligned} x_1 &= x_2 + u \\ x_2 &= 2x_1 - y \end{aligned}$$

The system consists of two unknown variables and two equations. Its bi-partite graph (with a matching) is:



There exists a cycle in this graph too, but see what happens when x_1 is solved.

$$\begin{aligned} &x_1 = 2x_1 - y + u \Rightarrow \\ &x_1 = y - u \Rightarrow \quad x_2 = 2y - 2u - y = y - 2u \end{aligned}$$

Both of the variables can be determined unambiguously.

From Example 1 the following conclusion can be drawn: The difference between an algebraic cycle, that an algebraic just determined system can be represented by, and a differential cycle is that in the second case the initial value has to be known in order to find an unambiguous solution. This extra knowledge is in most cases not provided for in a real application.

Differentiating algorithm

In the Linköping approach the dynamics can be modelled according to all of the three points described in Section 2.3. What is specific for this algorithm however is the structural differentiation. The theory behind this is presented in [**?**].

The structural differentiation expand the system by adding differentiated versions of the equations to the model. The goal is to be able to solve the unknown variables. To achieve this, the known variables have to have calculable derivatives and it has to be known how many. To show the importance of knowing the known variables derivatives consider the following system:

$$\ddot{x}_1 = x_2$$
$$x_2 = u$$
$$x_1 = y$$

The incidence matrix is:

	x_1	$\dot{x}_1$	$\ddot{x}_1$	x_2	u	y	$\dot{y}$	$\ddot{y}$
e_1			1	1				
e_2				1	1			
e_3	1					1		

It consists of three unknown variables and three equations. Assume that y can be derived once, which implies that equation e_3 can be structurally differentiated. The new incidence matrix is:

	x_1	$\dot{x}_1$	$\ddot{x}_1$	x_2	u	y	$\dot{y}$	$\ddot{y}$
e_1			1	1				
e_2				1	1			
e_3	1					1		
$\dot{e}_3$		1					1	

Now, the system consists of four equations and four unknown variables. Lets assume that the second derivative of y can be calculated. The new incidence matrix is:

	x_1	$\dot{x}_1$	$\ddot{x}_1$	x_2	u	y	$\dot{y}$	$\ddot{y}$
e_1			1	1				
e_2				1	1			
e_3	1					1		
$\dot{e}_3$		1					1	
$\ddot{e}_3$			1					1

Here, there are five constraints and still four unknown variables. The system has redundant information and can be used to create an MSS. The more information that can be extracted from the known variables derivatives, the more possibilities to expand the system. It is not however easy to know how many derivatives that can be calculated due to noise and other factors.

6.1.2 Find equation sets for the fault isolation analysis

In this section it is important to emphasize that there are differences between the theoretic algorithms and the implemented ones. For example the constraint which state that some variables are not matchable due to differential constraints or non invertible functions exist only in theory. No implementations have been done so far even though both methods could be adapted to it.

Matching algorithm

Concerning the matchings in the Lille method, it is simple to implement how to deal with the one-way directed equations. Normally it is only to indicate in the matching step that a variable is not matchable. This could for example be done by changing a "1" into a "0" in the incidence matrix.

A consequence of forbidding some matching directions is that a system can lose unnecessary much information. If for example the system equations are:

$$\begin{aligned} x^2 &= u \\ y &= x^4 \end{aligned}$$

The incidence matrix would be:

	x	u	y
c_1	Δ	1	0
c_2	Δ	0	1

In this case it is not possible to match x, but there exists an ARR

$$r : \quad 0 = y - u^2$$

where r stands for the residual. The reason why this ARR can be created is that it is not based on the knowledge of the value x as is the case normally, but x^2. The conclusion of this is that forbidding certain senses for a matching can in some cases cut information that could have been useful.

MSS algorithm

The objectives of the Linköping algorithm is to find all MSS:s and then only use those which contributes to the isolability. Since the MSS algorithm has a lot in common with the matching algorithm in the Lille approach, it is also here easy to adjust the implemented version to handle one-way functions.

An advantage for the MSS algorithm is that is is time efficient. From the incidence matrix to the MSS sets the matrix only needs to be searched once in comparison to twice in the Lille matching algorithm. What makes this efficiency possible is that when a new variable has been matched, the matched variables and equations are tested if they constitute an MSS or not.

6.2 Concerning the desirable properties

In Chapter 3, four desirable properties were discussed. They will now be used to evaluate the algorithms.

6.2.1 Realizable solutions

When a fault isolation analysis has been made, it may be of interest to construct a diagnosis system based on the results. As described in Section 2.1, diagnosis systems can be based on residuals. In Section 3.1 two obstacles in creating residuals were discussed, noninjective functions and differential constraints.

The Lille method uses matching based ARR:s to create their residuals. This is a straightforward technique and it is easy to see how the ARR:s are constructed. With a small change in the implementation, differential constraints and injective functions are handled to prevent a match in the "wrong sense" between variables. The problem however is to prevent differential cycles. There is not yet a way to avoid differential cycles in the matchings.

The Linköping method does not specify how to create the residuals. One possible way is to use the same approach as in the Lille algorithm to make a matching based ARR. This works if the system does not contain any injective functions. If so is case, the following example illustrates why it is not suitable.

Example 6.2
Consider the functions

$$f = x^2 \text{ and } g = y^3.$$

Their derivatives are

$$\dot{f} = 2x\dot{x} \text{ and } \dot{g} = 3y^2\dot{y}$$

respectively. In the first example f is matchable while x is not. When derived however both $\dot{f}$, x and $\dot{x}$ are matchable. For the second example

g is matchable at first and only $\dot{g}$ and $\dot{y}$ after the derivation but y is still not matchable. The consequence it gives is that some matchings may be too optimistic and perhaps not realizable.

There are other ways too to create ARR:S, one is shown in Section 6.1.2.

6.2.2 Ability to handle different types of systems

In Section 3.2 two sorts of different systems was discussed, linear/nonlinear and static/dynamic.

Concerning linear/nonlinear systems, both of the algorithms can handle both types of systems. In the Lille case, no notion is taken on this difference since the algorithm does not need it, while in the Linköping case, the nonlinearity is stored in the structural model to be used in the structural differentiation.

Static systems are relatively simple to make a structural analysis upon. It is when dynamics are allowed it can be uncertain how the derivatives should be modelled and how the dynamics should be handled in the analysis. In the two algorithms studied in this report the dynamics have been modelled by assuming that x and $\dot{x}$ are two different variables connected by differential constraints and by structural differentiation. The difficulties consists once again of differential cycles and how injectivity can be inherited in a structural differentiation.

6.2.3 Time complexity

The largest factor for how time consuming the algorithms are, is the size of the incidence matrix when it is searched for matchings and MSS:es and how many of the entries in the matrix that consist of "1:es". The advantages for the Lille algorithm is that the system is rather small in comparison to a differentiated system however, in that case the ARR creation is done in two steps. First the matching is made and then another search through the matrix is done in order to extract which equations that are needed in order to create the ARR.

When the system has been differentiated the complexity increases, but in the same time many of the variables can be reduced afterwards with the canonical decomposition simplification that is made.

6.2.4 Completeness

In the static case, both of the algorithms give the same results. This due to the fact that the main difference which affects the number of MSS:s lies within the handling of derivatives. When the dynamics are added, the number of MSS:s starts to differ. The addition of the derived equations, allows more MSS:es to be found. However, it is not for certain that all newfound MSS:es contribute to the fault isolability. Consider for example the system:

$$y_1 = x + f_1 \tag{6.1a}$$
$$y_2 = x + f_2 \tag{6.1b}$$

Lets assume that the sensor signals y and the faults f can be derived at least once each. Differentiation of the equations gives the following system:

$$\begin{aligned} y_1 &= x + f_1 \\ y_2 &= x + f_2 \\ \dot{y}_1 &= \dot{x} + \dot{f}_1 \\ \dot{y}_2 &= \dot{x} + \dot{f}_2 \end{aligned}$$

Its incidence matrix is:

	x	$\dot{x}$	y_1	y_2	$\dot{y}_1$	$\dot{y}_2$	f_1	f_2	$\dot{f}_1$	$\dot{f}_2$
e_1	1		1				1			
e_2	1			1				1		
$\dot{e}_1$		1			1				1	
$\ddot{e}_2$		1				1				1

Here, two MSS:es can be found:

$$\begin{aligned} \{e_1,\ e_2\} &\rightarrow \{f_1,\ f_2\} \\ \{\dot{e}_1,\ \dot{e}_2\} &\rightarrow \{\dot{f}_1,\ \dot{f}_2\} \end{aligned}$$

In this case the two MSS:es hold separate information. If for example one or both of the faults are constant. $\dot{f}_1$ and $\dot{f}_2$ are zero. On the other hand, if the faults in Equations 6.1 were nonlinear, the structural differentiation would give the following incidence matrix:

	x	$\dot{x}$	y_1	y_2	$\dot{y}_1$	$\dot{y}_2$	f_1	f_2	$\dot{f}_1$	$\dot{f}_2$
e_1	1		1				1			
e_2	1			1				1		
$\dot{e}_1$		1			1		1		1	
$\ddot{e}_2$		1				1		1		1

The new MSS:es are:

$$\{e_1,\ e_2\} \rightarrow \{f_1,\ f_2\}$$
$$\{\dot{e}_1,\ \dot{e}_2\} \rightarrow \{f_1,\ \dot{f}_1,\ f_2,\ \dot{f}_2\}$$

Here the second fault sensitivity does not contribute to the fault isolability since the same information is obtained in the first set.

6.3 Improving isolability

When the fault incidence matrix from the algorithms are given, it may consist of "blocks" of faults which cannot be distinguished from one another. To improve the isolability, different actions can be taken. Three of them are discussed and exemplified in this section.

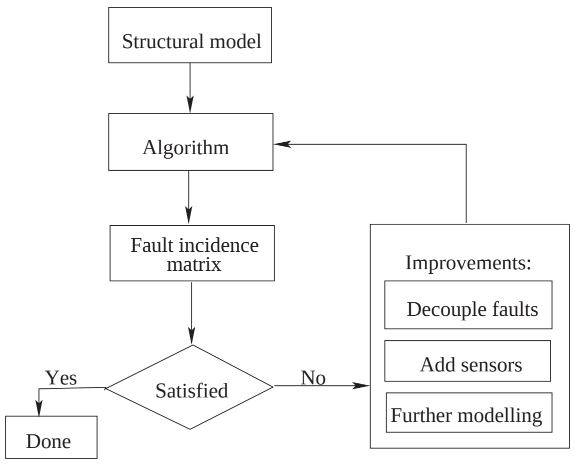


Figure 6.1: Improving the isolability.

6.3.1 Decoupling

Decoupling is a way to make the MSS:s unsensitive to a certain fault. This is done by adding the fault to the X set of the unknown variables, in other words say that the fault should be calculated as one of the wanted internal states. The fault can no longer make an ARR react, since an ARR does not consist of any calculated variables. The resulting ARR:s are therefore unsensitive to the decoupled fault.

To exemplify a situation where decoupling improve the isolability, consider the system:

$$\begin{aligned} y_1 &= x + f_1 + f_2 \\ y_2 &= x + f_1 \\ y_3 &= x + f_2 \end{aligned}$$

The incidence matrix is:

	x	y_1	y_2	y_3	f_1	f_2
e_1	1	1			1	1
e_2	1		1		1	
e_3	1			1		1

Here, an MSS can be derived. The MSS and its fault sensitivity is:

$$\begin{aligned} \{e_1,\ e_2\} &\rightarrow \{f_1,\ f_2\} \\ \{e_1,\ e_3\} &\rightarrow \{f_1,\ f_2\} \\ \{e_2,\ e_3\} &\rightarrow \{f_1,\ f_2\} \end{aligned}$$

Since the sets are sensitive to both of the faults, none of the faults are isolable. Lets now consider f_1 as an unknown variable. The new incidence matrix is:

	x	f_1	y_1	y_2	y_3	f_2
e_1	1	1	1			1
e_2	1	1		1		
e_3	1				1	1

Now there is one possible MSS:

$$\{e_1,\ e_2,\ e_3\} \rightarrow \{f_2\}$$

In the same way, f_2 can be decoupled which leads to an MSS set sensitive to f_1. With this extra piece of information added to the sets above the fault sensitivity matrix is:

	f_1	f_2
f_1	1	0
f_2	0	1

With the decoupling both of the faults are now isolable.

Logically this works better in a system with many relations in comparison to the number of variables. The reason is that one degree of liberty is taken away from the system for each decoupling. If an extra variable is added, and no new constraints are introduced, one of the redundant equations is needed in order to match the decoupled variable. The redundant relation is then "consumed".

6.3.2 Add sensors

If a fault is not isolable it can be interesting to add sensors. The advantage of adding sensors is that those relations are simple, often just between the state variable and the sensor itself. The disadvantage is that it can be expensive to find appropriate sensors and to maintain them. Here an example is shown how adding sensors can improve the fault isolability.

Consider the system which correspond to the following incidence matrix:

	x	x_2	u	y_1	y_2	f_1	f_2
e_1	1	1	1			1	1
e_2	1	1				1	
e_3	1			1			

One MSS can be found consisting of all three equations which is sensitive to both faults. Since one MSS never is enough to isolate two faults, extra knowledge about the system is required. Assume that x_2 can be measured by a sensor. Adding this to the model, the new incidence matrix becomes:

	x_1	x_2	u	y_1	y_2	f_1	f_2
e_1	1	1	1			1	1
e_2	1	1				1	
e_3	1			1			
e_4		1			1		

Now, two additional MSS:es can be created:

$$\{e_1,\ e_3,\ e_4\} \rightarrow \{f_1,\ f_2\}$$
$$\{e_2,\ e_3,\ e_4\} \rightarrow \{f_1\}$$

The new fault incidence matrix becomes:

	f_1	f_2
f_1	1	0
f_2	1	1

6.3.3 Further modelling

One of the main ideas of structural analysis is that it can use a relatively simple model to estimate the system's fault isolability properties. Nevertheless, if the results do not correspond to what is wanted, it can be necessary to add information about the model and the faults. One simple way of doing this can for example be if a fault is slowly changing, for example a clog fault, f_{clogg}. Then the extra information can be provided by adding the constraint $\dot{f}_{clogg} = 0$. This approach is used in [**?**]. Here an example is shown how extra modelling of one of the state variables can improve the isolability.

Consider the system which incidence matrix is:

	x_1	x_2	u	y_1	f_1	f_2
e_1	1	1	1		1	1
e_2	1			1		
e_3	1	1				1

An MSS can be derived consisting of all three equations and it is sensitive to both of the faults. Adding the information that x_2 is slowly varying the new incidence matrix becomes:

	x_1	x_2	u	y_1	f_1	f_2
e_1	1	1	1		1	1
e_2	1			1		
e_3	1	1				1
e_4		1				

This gives the two extra MSS:es:

$$\{e_1,\ e_3,\ e_4\} \rightarrow \{f_1,\ f_2\}$$
$$\{e_2,\ e_3,\ e_4\} \rightarrow \{f_1\}$$

Now, f_1 can be isolated.

Chapter 7

DAMADICS Example

DAMADICS stands for Development and Application of Methods for Actuator Diagnosis in Industrial Control Systems and is a research training network founded by the European Commission. Eleven partners from six countries participate in the project which goal is to develop online diagnostic tools for applications in power, food processing and chemical industries. The application is a valve in a sugar factory in Lublin, Poland.

My part in this project was to compare/evaluate the two ways of isolability calculations which have been presented in this report.

In this chapter the modelled valve is described followed by the isolability results from the algorithms. The focus is on the results and not the model itself. A similar study has been made before, see [**?**] where the Linköping algorithm is used to perform isolability calculations. To complement it, different cases have been studied regarding of how to model the derivatives. The goal is to see what effects different input models have on the number of MSS:s/ARR:s found.

7.1 Description of the Lublin sugar plant valve

In the sugar plant a specific part has been modelled consisting of a valve. A schematic figure of the valve is shown in Figure 7.1. The circles represent those variables which are measured. They are the valve plug position x, the fluid flow Q, fluid temperature T_1, up- and downstream pressure P_1 and P_2, and the transducer chamber pressure P_s.

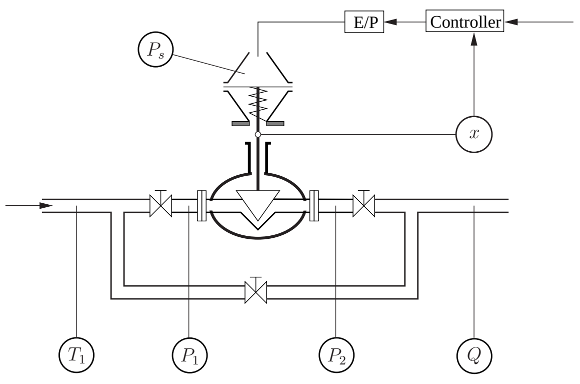


Figure 7.1: The DAMADICS valve.

The valve consists of three parts, the pneumatic servo-motor, the control valve and the bypass valve. To summarize, the model consists of 19 equations, 15 unknown variables, 14 faults and 9 known signals. Of the 9 known signals are 6 sensor signals and the others are the valve plug position controller reference value and output, and the position of the bypass valve. Its incidence matrix is shown in Figure 7.2.

In Appendix B all the variables are found with descriptions along with the model equations.

7.2 Test runs

In this section four different test runs are made of the DAMADICS valve, two with each method. The difference between the four cases are mainly on how to handle dynamic models. Note that it is the implemented versions of the algorithms that are used here. For example, in the Lille method the block that does not allow differential constraints to be matched in both directions is not implemented and one of the simplification steps in the Linköping neither.

The interesting parameters from the test runs are of course the fault incidence matrices, the number of MSS:es/ARR:s found and how much time it took to complete the computations.

For the Linköping algorithm it is interesting to see how the structural differentiation algorithm affects the result. Therefore the algorithm is used in two ways. First it is allowed to derive the variables structurally

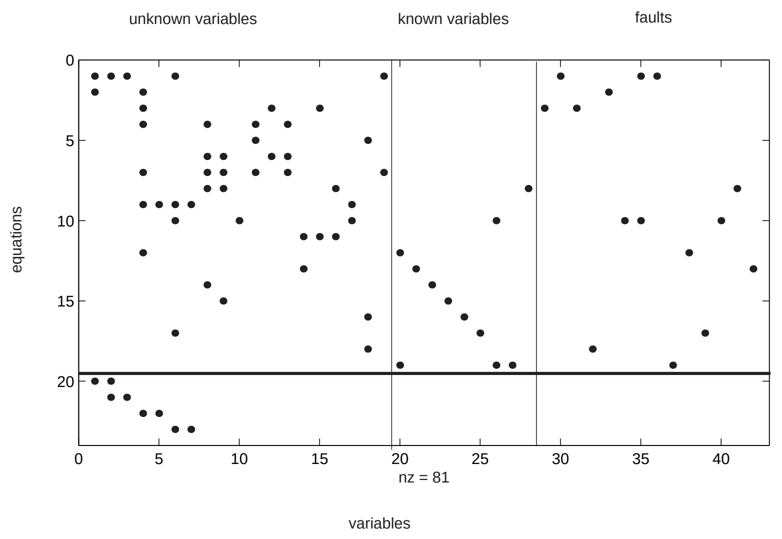


Figure 7.2: The DAMADICS valve incidence matrix. The order of the unknown variables are: x, $\dot{x}$, $\ddot{x}$, x_h, $\dot{x}_h$, P_s, $\dot{P}_s$, P_1, P_2, P_z, P_v, Δ_P, Δ_{P-a}, Q, Q_v, Q_{v3}, Q_c, T_1, F_{vc} . The last four equations are added as a possible connection between the derivatives and the original variables.

one time as described in Section 2.3.2. Secondly the derivatives are assumed to be calculable if the original variable is known according to what was described in Section 2.3.3. The Lille algorithm is tested with a model where x and $\dot{x}$ are considered as two different variables connected with differential constraints, described in Section 2.3.1 and on the static system. An overview of the different approaches is given in the following list.

1. Variables and derivatives are connected by structural differentiation. It is the algorithm that differentiates the variable and the equations. (Linköping)

2. The variables $\dot{x}$ and x are both given in the input model and they are considered as two different variables connected via differential constraints. The differential constraints are given explicitly. (Lille)

3. The derivative $\dot{x}$ is switched to x. All necessary derivatives are assumed to calculable if x is known. (Linköping)

4. Let the system be static, all derivatives are set to zero. (Lille)

7.2.1 Case 1

In this case the dynamics are modelled by saying that $\dot{x}$ is obtained by deriving the variable x. Effectively this is done by deleting the columns corresponding to the derived variables and add extra information in the non differentiated variables instead. In this run the variables are allowed to be differentiated ones. The isolability properties are predicted to be low, since there are second order derivatives in the model, but it will only be permitted one order derivatives in the residuals. Why this restriction is made here, is to show that the algorithm can be adjusted to conditions such as limitations in the sensor signals. If, for example the sensor signals can not be differentiated twice, a too optimistic fault incidence matrix would have been obtained if the differentiation limit would have been set higher. In Figure 7.3 the structure of the input model is shown. Note that the differential constraints not are used here.

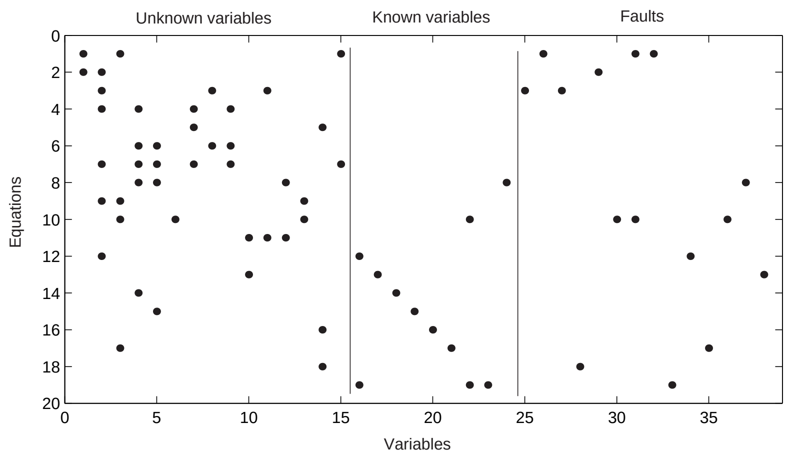


Figure 7.3: The DAMADICS valve's incidence matrix for the Linköping algorithm. The unknown variables are: x, x_h, P_s, P_1, P_2, P_z, P_v, Δ_P, Δ_{P-a}, Q, Q_v, Q_{v3}, Q_c, T_14, F_{vc}

The result from the Linköping algorithm is the fault incidence matrix:

$$\left(\begin{array}{cccccccccccccc}
\cdot & f_{16} & f_{14} & f_{11} & f_{10} & f_9 & f_8 & f_4 & f_{19} & f_{18} & f_{13} & f_5 & f_1 & f_7 & f_{12} \\
f_{16} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
f_{14} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
f_{11} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
f_{10} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
f_9 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
f_8 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
f_4 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
f_{19} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
f_{18} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
f_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
f_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
f_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
f_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
f_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right)$$

The matrix is generated from 50 MSS:s and took approximately 6 and a half minutes to perform without any of the simplifications. The original system of 19 equations was expanded to 38 equations and 33 unknown variables after the differentiation step. When the canonical decomposition simplification is used, the results are of course the same, but the elapsed time is reduced to approximately 5 seconds.

The isolation capabilities in this case is quite low. For the upper seven faults, nothing can be said. Only the last two faults f_7 and f_{12} can be isolated. One of the reasons behind this is that the system consists partly of variables with a second derivative. When creating the MSS:s the equations were only permitted to be derived ones and therefore those equations consisting of second derivatives cannot be used.

7.2.2 Case 2

The input in this case is the same as given in Figure 7.2. Here x and $\dot{x}$ are considered as different variables and they are connected by four differential constraints which can be seen in the bottom of Figure 7.2. The result is shown below:

$$\left(\begin{array}{cccccccccccccc c}
\cdot & f_{16} & f_9 & f_{19} & f_{18} & f_5 & f_1 & f_{14} & f_{11} & f_{10} & f_8 & f_4 & f_7 & f_{12} & f_{13} \\
f_{16} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
f_9 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
f_{19} & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
f_{18} & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
f_5 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
f_1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
f_{14} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
f_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
f_{10} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
f_8 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
f_4 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
f_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
f_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
f_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right)$$

This matrix is generated by 15 MSS:s and the time it took was about 25 seconds. The equivalent run with Linköping algorithm is about 5 sec. The isolation properties have improved, but still f_{16} and f_9 cannot be detected. A fault in one of the two blocks consisting of $\{f_{19},\ f_{18},\ f_5,\ f_1\}$and $\{f_{14},\ f_{11},\ f_{10},\ f_8,\ f_4\ \}$ respectively, are detectable, but cannot be isolated from the faults in the same group. Only the three last faults can be isolated.

7.2.3 Case 3

In this case it is assumed that all derivatives can be calculated if the original variable is known. The consequence of this assumption is that all occurrences of derived variables are exchanged into nonderived ones. It is sufficient to know the variables value in order to find all of its derivatives values.

The input to the algorithm is as shown in Figure 7.3 and the result is the fault incidence matrix:

$$\left(\begin{array}{ccccccccccccccc}
\cdot & f_{16} & f_9 & f_{19} & f_{18} & f_5 & f_1 & f_{14} & f_{11} & f_{10} & f_8 & f_4 & f_7 & f_{12} & f_{13} \\
f_{16} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
f_9 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
f_{19} & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
f_{18} & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
f_5 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
f_1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
f_{14} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
f_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
f_{10} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
f_8 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
f_4 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
f_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
f_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
f_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right)$$

This incidence matrix was generated from 15 MSS:s and that the generation took about 5 seconds. It has the same structure as in the previous run and it may be explained by the similarities in the input models. In Case 2 the differentiated variable can always be matched by the differential constraints 20 to 23. In Case 3 it was assumed that all derivatives could be calculated once the original variable has been matched. Both methods "guarantees" that once the unknown variable has been matched, its derivatives can be matched/calculed as well.

7.2.4 Case 4

In this case all derivatives are set to zero. The input model is as in Figure 7.3. The result is the same as in Case 2 and Case 3. The result can at first seem to be surprising, since no dynamic and maximum number of derivatives in the algorithms gives the same result, but when only one derivative is allowed, the isolability is worse as in Case 1. One explaining reason behind this can be that in Case 1, some of the equations could not be in an MSS, the information in those equations are "lost". In the static case, all equations are used and even if no derivatives are present, some information could be extracted.

7.3 Discussion

The best results concerning the isolability properties are obtained in Case 2, 3 and 4. However, in Case 4 the analysis was based on a static system, and therefore not as interesting. The reason why Case 1 shows less isolability can be explained by the fact that all the variables were only permitted to be derived once but in the equations there were second order of derivatives. These could not be used in the MSS:es. If the variables would have been permitted to be differentiated twice, the results would have been the same as in the other cases. The limit for differentiation of the variables is used to be able to control the maximal complexity of the ARR:s.

Concerning the time complexity, its importance is shown the best in Case 1. There, the same run with and without the canonical decomposition simplification shows a relatively large difference in time consumption. The DAMADICS valve is a relatively small example and in larger applications the difference can be expected to be even larger. It should also be added that this simplification could be more important for the Linköping algorithm then the Lille one. When structural differentiation is made the number of equations and variables increases rapidly. A consequence the Lille method does not have to take under consideration as much.

There is also a difference between the methods regarding the time complexity. In Case 2 it took approximately 25 seconds to perform the calculations with the Lille method and approximately five seconds for the Linköping algorithm. The difference lies in how the implementations have been made. For both methods the most time consuming part is when the incidence matrix is searched recursively. In the Lille case this is made two times, first when the matchings are made and once more when the equations for the ARR:s are found. The corresponding is made in one search in the Linköping implementation when the MSS:es are found.

In these tests, non of the proposed improvements from Section 6.3 have been made here. However, in [**?**] decoupling is used to try and isolate more faults. However it did not provide any extra information and extra modelling was tried instead. The faults were said to be slow varying and that information was added. With this extra knowledge the block of five ones in Section 7.2.2 got isolable.

Chapter 8

Programming contributions

Two sorts of programming contributions were made. First a graphical user interface was implemented, and then a routine to handling decoupling. Both were made using Matlab.

8.1 Graphical user interface

To simplify the handling of the Lille algorithm, a small GUI was designed. With the GUI, it is easy to decouple faults and to get information about the different steps in the algorithm, for example to see the overdetermined system and which equations that are needed in order to isolate the different faults.

In the sections below, the menus and functions are described. A schematic description of the interface is presented in Figure 8.1.

8.1.1 Starting the program

Before any calculations can be done, a structural model needs to be loaded. This is done in the first menu. The user is asked to type the input file name in the Matlab window. The file should contain the three matrices x, y and f which correspond to the incidence matrix divided in an unknown, a known and a fault part.

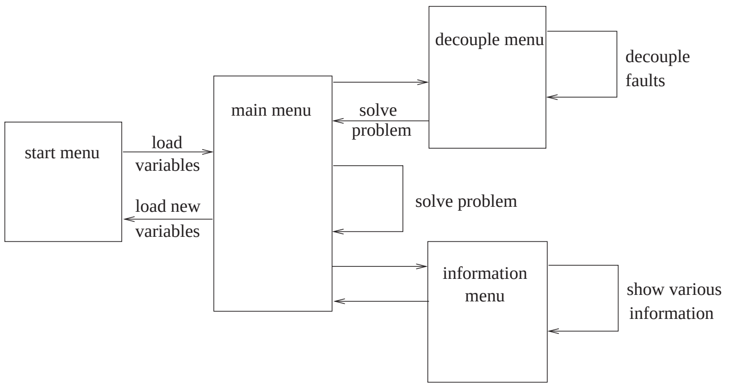


Figure 8.1: An overview of the graphical user interface.

8.1.2 The main menu

The options in the main menu are to solve the problem without decoupling, decouple one or several faults, to show information about the calculations or to load new variables. If "solve without decoupling" is chosen, the resulting incidence matrix is shown in the Matlab command window. If decoupling is wanted, a new menu administrates which faults that should be decoupled. After solving the decoupled system the result is shown in the Matlab command window and the main menu is reactivated.

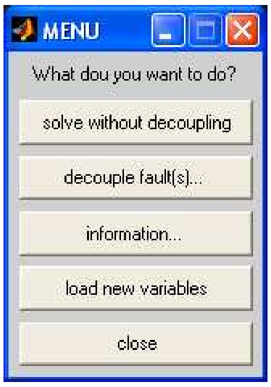


Figure 8.2: The main menu of the GUI.

8.1.3 The information menu

The information menu permits to view mid-way results in the Matlab command window. Such information can be the fault sensitivity matrix and the matrix stating which equations the ARR:s consist of.

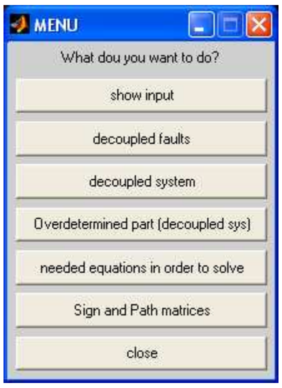


Figure 8.3: The Information menu of the GUI.

8.2 Decoupling algorithm

The main idea with decoupling faults is to think of the faults as something unknown, but possible to calculate. This implies that when the matchings are made, the decoupled fault needs to be matched as well as the unknown variables x. Mathematically, this means that when the possible matchings are made, the corresponding ARR:s are constructed in a way that the decoupled fault is always eliminated and hence will not change the value of the residual. If there exist an ARR, it can not be sensitive to the decoupled fault, and thereby the other faults can be isolated from the decoupled one.

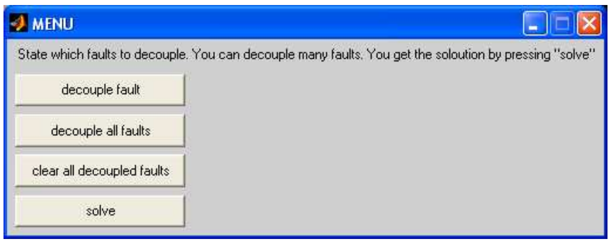

MENU
State which faults to decouple. You can decouple many faults. You get the soloution by pressing "solve"
decouple fault
decouple all faults
clear all decoupled faults
solve

Figure 8.4: The decoupling menu of the GUI.

Chapter 9

Concluding remarks

In this chapter my contributions and conclusions are listed with some ideas of future work. The main contribution of this thesis is the discussion concerning the two structural methods for fault analysis. On the way of making this discussion, structural analysis was explained in a way to suite the two algorithms, another contribution. In Chapter 3, four desirable properties were listed. They were derived from general desirable properties for diagnosis systems. In the following sections the points are summarized with the algorithms.

9.1 Summary

9.1.1 Differences

Two main differences were discussed, the handling of dynamic models and how the ARR/MSS algorithm work. The addition of differential constraints as in the Lille case is straightforward, but there is a risk of differential cycles. The ARR algorithm can take under consideration that differential constraint only are matchable in one sense, but on the other hand this restriction can lead to that not all ARR:s are found.

With structural differentiation redundancy can in some cases be obtained. The restriction of the number of known derivatives, increases the possibility to realize residuals from the obtained MSS:es. This because it is known on beforehand that all components are known in the set.

9.1.2 The four desirable points

Realizable solutions

Two obstacles for realizable solutions were discussed, differential constraints and non injective functions. For the Lille approach the greatest obstacle of creating ARR:s are differential cycles, a consequence of how the differential model is constructed and the matching based creation of ARR:s. When using structural differentiation all of the equations are algebraic, which implies that differential cycles cannot appear. High order of derivatives of the known variables in the ARR:s can make the ARR:s impossible to realize due to numerical problems in the differentiation. The Linköping method can control the maximum order of derivatives allowed in the MSS:es and therefor avoids ARR:s with non computable derivatives.

Handling different types of systems

Both methods can handle both linear and nonlinear systems. The Lille method does not separate linear and nonlinear functions while the Linköping method can, but do not necessarily have to, use the extra knowledge if an appearance is linear in the structural differentiation algorithm.

Time complexity

The time complexity depends on the size of the incidence matrix and the degree of completion. Amongst the algorithms the Linköping one is faster but it should also be stated that in that case the result is a set of equations that can be used to create a residual, while the Lille algorithm also gives knowledge of how to create the ARR. The time consuming part is when the incidence matrices are searched. Such a search is preformed twice in the Lille algorithm and once in the Linköping method.

Completeness

The Lille method finds all ARR:s which are based on matchings, but can miss ARR:s which are created in other ways. The Linköping algorithm gives information on which equations it is possible to combine to create redundancy, but does not say anything about *how* they should be realized. By this reasoning it should imply that the Linköping algorithm can find all MSS:es corresponding to the ARR:s in the Lille case, but not the other way around. It is not however for certain that all MSS:es contribute information.

9.1.3 The DAMADICS example

The two methods were used on a valve in a sugar plant in Poland. The test runs show similar results. There were some differences in number of ARR:s and MSS:es found. In case 1, 50 MSS:es were found and in case 2 and 3 there were 15. The difference depends on the structural differentiation which adds more constraints which permits more MSS:es to be found. In this case the higher number of MSS:es did not imply higher isolability however since the run was restricted to only permit first order derivatives and the system was a second order system.

9.1.4 The Graphical user interface

A graphical user interface was implemented for the Lille method which simplifies the fault analysis and to extract information from the calculations.

9.2 Further work

- Avoid matches which involve integration (In theory already done, but needs to be implemented)
- Investigate the meaning of differential cycles, how they can be avoided and managed?
- To implement a graphical user interface that operates with both of the algorithms.
- Investigate how multiple faults can be handled.
- Improve the search algorithms in the incidence matrices.

Appendix A

Matching algorithm for the Linköping method

Algorithm description taken from [**?**]

This section explains how the MSS sets are found for the example given in section 5.3.4. The task is to find all MSS sets in the model $M_simpwithequations$ $\{e_1, \ldots, e_n\}$. Let $M_k = \{e_k, \ldots, e_n\}$ be the last $n - k + 1$ equations. Let E be the current set of equations that is examined. The set of MSS sets found is denoted ω. Then the following algorithm Finds all MSS sets in M_{simp}.

Algorithm : Input: The model M_simp

1. Set $k = 1$ and $\omega = \emptyset$
2. Choose equation e_k. Let $E = \{e_k\}$ and $X = \emptyset$
3. Find all MSS sets that are subsets of M_k and include equation e_k
    a) Let $\tilde{X} = var_{X_u}(E) \setminus X$ be the unmatched variables.
    b) If $\tilde{X} = \emptyset$, then E is an MSS set. Insert E into ω.
    c) Else take a remaining variable $\tilde{x} \in \tilde{X}$ and let $X = X \cup \{\tilde{x}\}$. Let $\tilde{E} = equ_{M_k \setminus E}(\tilde{x})$ be the remaining equations. For all equations e in $\tilde{E}$ let $E = E \cup \{e\}$ and goto step a).
4. If k ¡ n set k = k + 1 and goto step number 2. Output: The set of MSS sets found, i.e. ω.

Finding MSS:s for the small tank example

round	step	action
1	1	$k = 1 \quad \omega = \emptyset$
1	2	$E = \{e_1\} \quad X = \emptyset$
1	3a)	$\tilde{X} = var_{X_u}(E) \setminus X = \{q_i\ q_o\ h\}$
1	3b)	$\tilde{X} \neq \emptyset$
1	3c)	$\tilde{x} = q_i \quad \tilde{X} = \{q_i\}$,
		$\tilde{E} = equ_{M_k \setminus E}(\tilde{X}) = \{\mathbf{e_2}\ e_5\}$
		$X = \{q_i\}$
		Continue with e_2
1.1	3a)	$e = e_2, \quad E = \{e_1\ e_2\}$
		$\tilde{X} = var_{X_u}(E) \setminus X = \{q_i\ q_o\ h\} \setminus \{q_i\} = \{q_o,\ h\}$
1.1	3b)	$\tilde{X} \neq \emptyset$
1.1	3c)	$\tilde{x} = q_o \quad \tilde{X} = \{q_i\ q_o\}$,
		$\tilde{E} = equ_{M_k \setminus E}(\tilde{X}) = \{\mathbf{e_3}\}$
		$X = \{q_i\ q_o\}$
1.1.1	3a)	$e = \mathbf{e_3}, \quad E = \{e_1\ e_2\ e_3\}$
		$\tilde{X} = var_{X_u}(E) \setminus X = \{q_i\ q_o\ h\} \setminus \{q_i\ q_o\} = h$
1.1.1	3b)	$\tilde{X} \neq \emptyset$
1.1.1	3c)	$\tilde{x} = h \quad \tilde{X} = \{q_i\ q_o\ h\}$,
		$\tilde{E} = equ_{M_k \setminus E}(\tilde{X}) = \{\mathbf{e_4}\}$
		$X = \{q_i\ q_o\ h\}$
1.1.1.1	3a)	$e = \mathbf{e_4}, \quad E = \{e_1\ e_2\ e_3\ e_4\}$
		$\tilde{X} = var_{X_u}(E) \setminus X = \{q_i\ q_o\ h\} \setminus \{q_i\ q_o\ h\} = \emptyset$
1.1.1.1	3b)	$\tilde{X} = \emptyset! \implies \omega = \{e_1\ e_2\ e_3\ e_4\}$
		Continue with e_5
1.2	3a)	$e = \mathbf{e_5}, \quad E = \{e_1\ e_5\}$
		$\tilde{X} = var_{X_u}(E) \setminus X = \{q_i\ q_o\ h\} \setminus \{q_i\} = \{q_o\ h\}$
1.2	3b)	$\tilde{X} \neq \emptyset$
1.2	3c)	$\tilde{x} = q_o \quad \tilde{X} = \{q_i\ q_o\}$,
		$\tilde{E} = equ_{M_k \setminus E}(\tilde{X}) = \{\mathbf{e_3}\}$
		$X = \{q_i\ q_o\}$
1.2.1	3a)	$e = \mathbf{e_3}, \quad E = \{e_1\ e_3\ e_5\}$
		$\tilde{X} = var_{X_u}(E) \setminus X = \{q_i\ q_o\ h\} \setminus \{q_i\ q_o\} = h$
1.2.1	3b)	$\tilde{X} \neq \emptyset$
1.2.1	3c)	$\tilde{x} = h \quad \tilde{X} = \{q_i\ q_o\ h\}$,
		$\tilde{E} = equ_{M_k \setminus E}(\tilde{X}) = \{\mathbf{e_4}\}$
		$X = \{q_i\ q_o\ h\}$
1.2.2	3a)	$e = \mathbf{e_4}, \quad E = \{e_1\ e_3\ e_4\ e_5\}$
		$\tilde{X} = var_{X_u}(E) \setminus X = \{q_i\ q_o\ h\} \setminus \{q_i\ q_o\ h\} = \emptyset$
1.2.2	3b)	$\tilde{X} = \emptyset! \implies \omega = \{e_1\ e_3\ e_4\ e_5\}$
		Start new round
2	1	$k = 2 \quad \omega = \emptyset$
2	2	$E = \{e_2\} \quad X = \emptyset$
2	3a)	$\tilde{X} = var_{X_u}(E) \setminus X = \{q_i\}$
2	3b)	$\tilde{X} \neq \emptyset$
2	3c)	$\tilde{x} = q_i \quad \tilde{X} = \{q_i\ q_o\ h\}$,
		$\tilde{E} = equ_{M_k \setminus E}(\tilde{X}) = \{\mathbf{e_5}\}$
		$X = \{q_i\}$
2.1	3a)	$e = \mathbf{e_5}, \quad E = \{e_2\ e_5\}$
		$\tilde{X} = var_{X_u}(E) \setminus X = \{q_i\} \setminus \{q_i\} = \emptyset$
2.1	3b)	$\tilde{X} = \emptyset! \implies \omega = \{e_2\ e_5\}$

Appendix B

The DAMADICS model

B.1 Variables in the model

variable	Description
	unknown variables
x	Valve-rod displacement
x_h	hysteresis of x
P_s	Pneumatic servo-motor chamber pressure
P_1	valve upstream pressure
P_2	valve downstream pressure
P_v	
Δ_P	Difference between P_1 and P_2
Δ_{P-a}	Allowed difference between P_1 and P_v
T_1	valve upstream temperature
Q_v	Flow past control valve
Q_{v3}	Flow past bypass valve
Q	Total flow
Q_c	net mass flow into the chamber
Pz	Supply pressure for electro-pneumatic trancducer
	known variables
?	One missing
C_v	Commanded displacement
CVI	Displacement controller output
y_{P_s}	server-motor chamber pressure
y_x	Measured rod position
y_Q	Total flow measurement
y_{P1}	Valve upstream pressure measurement
y_{P2}	valve downstream pressure measurement
y_T	Valve upstream temperature measurement

B.2 Possible faults

Fault	Description
	Control valve faults
f_1	valve clogging
f_2	valve or valve-seat sedimentation
f_3	valve or valve-seat erosion
f_4	increase of valve or bushing friction
f_5	external leakage (bushing, covers, terminals)
f_6	internal leakage (valve tightness)
f_7	medium evaporation or critical flow
	Pneumatic servo-motor faults
f_8	twisted servo-motor piston rod
f_9	servo-motor housing or terminals tightness
f_{10}	servo-motor diaphragm perforation
f_{11}	servo-motor spring fault
	Positioner faults
f_{12}	electro-pneumatic transducer fault
f_{13}	rod displacement sensor fault
f_{14}	pressure sensor fault
f_{15}	positioner feedback fault
f_{16}	positioner supply pressure drop
	General/external faults
f_{17}	Unexpressed pressure change across the valve
f_{18}	fully or partly opened bypass valve
f_{19}	flow rate sensor fault

B.3 Model equations

$$P_sA_e(1-f_{10}) = m\ddot{x} + k_v(1+f_4)\dot{x} + (k_s(1+f_{11})+k_d)x + +F_{vc} + (k_s(1+f_{11})+k_d)x_0 - mg \tag{B.1}$$

$$x_h = hyst(x)[f_8=0] + hyst(x;f_8)[f_8\neq 0] \tag{B.2}$$

$$Q_v = 100(1-f_5)(1+f_1)K_v(x_h)\sqrt{\frac{\Delta_p}{\rho}} \tag{B.3}$$

$$\Delta_{p-allow} = K_m(x_h)(P_1 - r_c(P_1)P_v) \tag{B.4}$$

$$\log(p_v) = -\frac{a}{T_1} + b \tag{B.5}$$

$$\Delta_p = (P_1-P_2)[P_1-P_2<\Delta_{p-allow}] + +\Delta_{p-allow}[P_1-P_2>\Delta_{p-allow}] \tag{B.6}$$

$$F_{vc} = \pi r^2 P_1\frac{-\Delta_p}{K_m(x_h))}[P_1-P_2<\Delta_{p-allow}] + +\pi r^2\frac{1}{K_m(x_h))}[P_1-P_2>\Delta_{p-allow}] \tag{B.7}$$

$$Q_{v3} = K_{v3}(x_3(1-f_{18}))\sqrt{\frac{P_1-P_2}{\rho}} \tag{B.8}$$

$$Q_cK_c = \dot{P}_sV(x_h) + P_s\frac{dV(x_h)}{dx}\dot{x_h} \tag{B.9}$$

$$Q_c = (1-f_9 2\cdot 10^{-6}\sqrt{P_s})(1-f_{10}2\cdot 10^{-6}\sqrt{P_s})|CVI| K_{c1}(P_z(1-f_{16})-P_s)[CVI\geq 0] - -|CVI|K_{c2}P_s[CVI<0] \tag{B.10}$$

$$Q = Q_v + Q_{v3} \tag{B.11}$$

$$y_x = x_h(1+1.25f_{13}) \tag{B.12}$$

$$y_Q = Q(1+f_{19}) \tag{B.13}$$

$$y_{P1} = P_1 \tag{B.14}$$

$$y_{P2} = P_2 \tag{B.15}$$

$$y_T = T_1 \tag{B.16}$$

$$y_{Ps} = max(0, min(1,(1+f_{14})P_s)) \tag{B.17}$$

$$T_1 = T_{10}[f_7=0] + (T_{10}+200+100f_7)[f_7\neq 0] \tag{B.18}$$

$$CVI = (1-f_{12})K_p(C_v - f(y_x)) \tag{B.19}$$