Structural Algorithms for Diagnostic System Design Using Simulink Models

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

by
Lars Eriksson

Reg nr: LiTH-ISY-EX-3601-2004

10th January 2005

Structural Algorithms for Diagnostic System Design Using Simulink Models

Master's thesis

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

by **Lars Eriksson**

Reg nr: LiTH-ISY-EX-3601-2004

Supervisor: **Mattias Krysander**
Linköpings Universitet
Mattias Nyberg
Scania CV AB

Examiner: **Assistant Professor Erik Frisk**
Linköpings Universitet

Linköping, 10th January 2005

Avdelning, Institution
Division, Department

Vehicular Systems,
Dept. of Electrical Engineering
581 83 Linköping

Datum
Date

10th January 2005

Språk
Language
☐ Svenska/Swedish
☒ Engelska/English
☐ _____

Rapporttyp
Report category
☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport
☐ _____

ISBN
—

ISRN
LITH-ISY-EX-3601-2004

Serietitel och serienummer **ISSN**
Title of series, numbering —

URL för elektronisk version
http://www.vehicular.isy.liu.se
http://www.ep.liu.se/exjobb/isy/2004/3601/

Titel Strukturella Algoritmer för Design av Diagnossystem med Simulink-modeller

Title Structural Algorithms for Diagnostic System Design Using Simulink Models

Författare Lars Eriksson
Author

Sammanfattning
Abstract

Today's society depends on complex and technically advanced mechanical systems, often containing a variety of different components. Despite careful development and construction, some of these components may eventually fail. To avoid unnecessary damage, for example environmental or financial, there is a need to locate and diagnose these faults as fast as possible. This can be done with a diagnostic system, which should produce an alarm if there is a fault in the mechanical system and, if possible, indicate the reason behind it.

In model based diagnosis, a mathematical model of a fault free system is used to detect if the monitored system contain any faults. This is done by constructing fault indicators, called fault tests, consisting of equations from different parts of the model. Finding these parts is a time-consuming and demanding task, hence it is preferable if as much as possible of this process can be automated. In this thesis an algorithm that finds all parts of a system that can be used to create these fault tests is presented. To make this analysis feasible, in industrial applications, a simplified version of a system model called a structural model is used. Since the models considered in this thesis are implemented in the mathematical software Simulink, a method for transforming Simulink models into analytical equations and structural models is described. As a way of increasing the diagnostic performance for a model based diagnostic system, information about different faults, called fault models, can be included in the model. However, since the models in this thesis are implemented in Simulink, there is no direct way in which this can be preformed. This thesis describes a solution to this problem. The correctness of the algorithms in this thesis are proved and they have been applied, with supreme results, to a Scania truck engine model.

Nyckelord
Keywords Model based diagnosis, Structural methods, Diagnostic systems, Graph theory, Decomposition

Abstract

Today's society depends on complex and technically advanced mechanical systems, often containing a variety of different components. Despite careful development and construction, some of these components may eventually fail. To avoid unnecessary damage, for example environmental or financial, there is a need to locate and diagnose these faults as fast as possible. This can be done with a diagnostic system, which should produce an alarm if there is a fault in the mechanical system and, if possible, indicate the reason behind it.

In model based diagnosis, a mathematical model of a fault free system is used to detect if the monitored system contain any faults. This is done by constructing fault indicators, called fault tests, consisting of equations from different parts of the model. Finding these parts is a time-consuming and demanding task, hence it is preferable if as much as possible of this process can be automated. In this thesis an algorithm that finds all parts of a system that can be used to create these fault tests is presented. To make this analysis feasible, in industrial applications, a simplified version of a system model called a structural model is used. Since the models considered in this thesis are implemented in the mathematical software Simulink, a method for transforming Simulink models into analytical equations and structural models is described. As a way of increasing the diagnostic performance for a model based diagnostic system, information about different faults, called fault models, can be included in the model. However, since the models in this thesis are implemented in Simulink, there is no direct way in which this can be preformed. This thesis describes a solution to this problem. The correctness of the algorithms in this thesis are proved and they have been applied, with supreme results, to a Scania truck engine model.

Keywords: Model based diagnosis, Structural methods, Diagnostic systems, Graph theory, Decomposition

Acknowledgments

This work has been carried out at the department of Electrical Engineering, division of Vehicular Systems at Linköpings University as a combined project between the department of Electrical Engineering and Scania CV AB in Södertälje.

I would like to express my gratitude to an number of people:

My excellent supervisors Mattias Krysander and Mattias Nyberg for many interesting and inspiring discussions during the work. The staff at Vehicular Systems, special thanks to Erik Frisk for being my examiner and Jan Åslund for helping me with some mathematical issues. My fellow master thesis students Gustav Arrhenius, Henrik Einarsson, Kristian Krigsman and John Nilsson at Scania, Södertälje. Mikael Rautio, Jonas Eriksson and especially Jonas Elmqvist for giving me valuable feedback on my report. Thank you all!

Finally I would like to thank my beloved girlfriend Tina for coping with me during this work, and my mother and father for always believing in and supporting me throughout my education.

Linköping, December 2004

Lars Eriksson

Contents

Notation

Operators

$\|S\|$	number of items in the set S
M	model i.e. a set of equations
X	set of unknown variables
$var_{X'}M$	set of variables in X' that are included in the model M
$equ_M X'$	set of equations in M that include some variable in X'
$\mathcal{G}(M, var_{X'}M)$	bipartite graph with vertex set M and X'
$var(\mathcal{G})$	set of all variable vertices in the graph $\mathcal{G}$
$equ(\mathcal{G})$	set of all equation vertices in the graph $\mathcal{G}$
$var(\gamma)$	set of all variable vertices connected to an edge in the edge set γ
$equ(\gamma)$	set of all equation vertices connected to an edge in the edge set γ

Abbreviations

SO	Structurally Overdetermined
MSO	Minimal Structurally Overdetermined
DSSM	Differentiated-Separated Structural-Model
DLSM	Differentiated-Lumped Structural-Model

Chapter 1

Introduction

This master's thesis was performed at the department of Electrical Engineering, division of Vehicular Systems at Linköpings University as a combined project between the department of Electrical Engineering and Scania CV AB in Södertälje. Scania is a worldwide manufacturer of heavy duty trucks, buses and engines for marine and industrial use. The work was carried out for the engine software development department, which is responsible for the engine control and the on board diagnostics (OBD) software.

1.1 Background

Today's society depends on complex and technically advanced mechanical systems. Despite rigorous and careful development and construction, some of the components in these systems may eventually fail. To avoid unnecessary damage there is a need to locate and diagnose these failures as fast as possible. This can be illustrated through numerous examples. Airplanes contain lots of safety critical systems. If a part of an engine begin to malfunction, it is important to detect this as soon as possible in order to carry out necessary maintenance. In the process industry a lot of money can be saved if a faulty component can be detected and replaced on a scheduled break in the manufacturing process, without causing any unplanned stop. The construction of diagnosis systems is therefore an important and necessary step in the development process for a lot of industrial applications.

Diagnosis of systems has been around for as long as there have been machines. In the beginning the diagnosis consisted in manual inspections. With the introduction of computers, new ways of checking the correctness of systems was developed. Initially, techniques for detecting

failures by analyzing signal levels of sensors were used. When a signal level exceeded a predefined level at a specific working point the conclusion was that a failure had occurred. In addition to that, model based diagnosis was introduced. Model based diagnostic systems uses the underlying mathematical models for the physical components to diagnose the components. This made it possible to base the entire diagnostic system on calculations. Using these model based diagnosis systems has made it possible to construct even more accurate and automated kinds of failure checking.

1.2 Related Work

This work can be seen as a link in a chain in the developing of model based diagnosis for engine systems. The objective is to develop methods so that the whole procedure is as automated as possible. The working process can schematically be described by the following list.

1. Construction of an engine model in Simulink.
2. Extracting model equations from the Simulink model.
3. Finding parts of the equation system that can be used for diagnosis.
4. Constructing executable diagnostic tests for these parts.

The modeling work described in step (1) has been carried out in a variety of different phases and by different persons. For modeling work related the specific Scania truck engine mentioned in this thesis see [3]. In this thesis, the emphasis is on step (2) and (3) in the list above. Work related to this has prior been presented by Mattias Krysander in [7]. Parallel to this work, methods for handling step (4) has been developed at Scania CV AB in Södertälje. For a description of this work see [5].

1.3 Problem Formulation

Previous on-board diagnostic systems at Scania have mainly been manually constructed. These have been obtained through hard testing and sometimes an intuitive feeling of what part of an engine that can be used in the construction of a diagnostic system. This approach is both time consuming and ineffective. If for example a last minute change is made in the engine, a total reconstruction of the diagnosis system may be necessary. Since the modeling work at Scania has been successful, creating models that are more and more accurate with low mean

errors, model based diagnosis is becoming a realistic alternative to existing methods. Model based diagnosis uses a mathematical description of the system to be diagnosed i.e. the system can be described by a set of equations. This makes it possible to use automated tools in the construction process of the diagnosis system. By using parts of the equation system, sensitive to specific faults, it is possible construct fault indicators, so-called fault tests. One problem is to find which parts of the equation system that can be used to construct these tests. Previous attempts to develop methods to find these parts have resulted in highly ineffective and computably demanding algorithms. This has made it impossible to develop model based diagnosis, in the form described by this work, for large and highly redundant models. Even though the part of the construction process described in this thesis do not have any real time computation demand, it is desirable that the algorithms produce results in matter of hours. Previous algorithms have had difficulties to produce any results at all for the type of engine models considered in this thesis, even if they have been running for days.

1.4 Objectives

The main objective of this thesis is to find all parts of an engine model that can be used in order to construct a diagnostic system i.e. to find subsets of equations in the system model that can be used to indicate if something has failed. The existing algorithms for doing this are inadequate since they, due to computational inefficiency, fails to present results for certain kind of systems. Hence, there is an obvious need to develop new versions of these algorithms. Since the model is implemented in Matlab/Simulink there is also a need to find ways of transforming a Simulink model into analytical equations. In order to increase the performance of the diagnostic system the concept of behavioral modes in Simulink will also be investigated.

1.5 Target Group

The target group of this work is primarily M.Sc./B.Sc. students with basic knowledge in modeling work and some experience in using the modeling and simulation software Matlab/Simulink.

1.6 Thesis Outline

This section describes the outline of this thesis.

Chapter 2 gives an introduction to diagnostic theory. In order to increase the ability for fault tests to pin point exactly what fault

that have occurred in a system, so-called behavioral modes will be introduced. The concept of structural models will be introduced as a solution to reduce the computing time for finding diagnostic fault tests.

Chapter 3 describes a solution to how a Simulink model may be transformed to model equations and how behavioral modes can be included into a Simulink model.

Chapter 4 presents new structural algorithms for extracting parts of a system model that can be used to construct diagnostic tests. In the end of this chapter, these new algorithms are mathematical verified by introducing a series of lemmas and theorems.

Chapter 5 present an industrial example with a Scania truck engine model. Compares and verifies the new algorithms by a series of execution tests.

Chapter 6 concludes this thesis by presenting the main results and discussing possible future work.

1.7 Contributions

In this section the main contributions of this thesis are listed.

- An algorithm that transforms Simulink models into analytical equations.
- A method that describes how fault models can be included graphically into a Simulink model.
- An algorithm that simplifies a structural model.
- Algorithms for finding all parts of a system model that can be used when constructing a diagnostic system.
- New mathematical properties concerning structural models and graph theory, for example the property of overdeterminedness.
- Mathematical verification of the derived algorithms through a number of proved theorems and lemmas.
- The algorithms in this thesis provides a foundation toward a completely automated construction process of model based diagnostic systems.

Chapter 2

Diagnostic Theory

In this chapter some basic theory about diagnostic systems and their properties are presented. In the first section some commonly used definitions regarding diagnosis will be explained. In the following section the general concepts behind diagnosis systems and behavioral modes are discussed. Finally in the last section the theory behind structural methods is introduced. The theory in these sections is explained with emphasis on what is needed for understanding the following chapters. For a more thorough description on the subject see [4] and [2].

2.1 Terminology and Definitions

In Figure 2.1 an engine containing a diagnostic system is shown. The engine behavior received from sensors y is compared to expected behavior $\hat{y}$ calculated from an engine model. If these values diverge from each other more then a predefined threshold j, the diagnostic system signals that something is wrong.

In this section some of the terminology used in the field of diagnosis and in this thesis are explained. The following definitions are suggested by the IFAC (International Federation of Automatic Control) Technical Committee SAFEPROCESS and are given in [9].

Fault: An unpermitted deviation of at least one characteristic property or parameter of the system from the acceptable/usual/standard condition.

Failure: A permanent interruption of a system's ability to perform a required function under specified operating conditions.

Disturbance: An unknown (and uncontrolled) input action on a system.

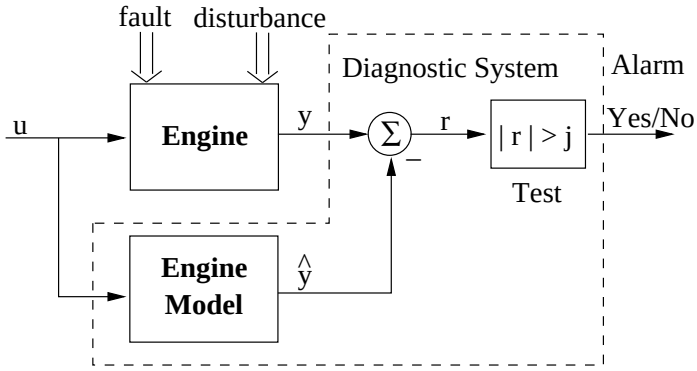

Figure 2.1: An example of model based diagnosis.

Residual: A fault indicator, based on a deviation between measurements and model-equation-based computations (see r in Figure 2.1).

Diagnostic Test: A binary test, designed to indicate if a residual exceeds a predefined threshold (see $Test$ in Figure 2.1.

Fault Diagnosis: Determination of the kind, size, location and time of detection of a fault. Follows fault detection. Includes fault isolation and identification.

The definition fault diagnosis is in this thesis only used to denote the actions taken by a so-called diagnostic system in order to determine the kind and location of a fault (see Figure 2.1. The concept of diagnostic systems is explained more thorough in the following section.

2.2 Diagnostic Systems

As mentioned in the introduction the main objective of this thesis is to find parts of a model that can be used to construct a diagnostic system. In this section the basic concepts of diagnostic systems will be explained.

The general idea of a diagnostic system is to indicate if a system behavior diverges from the expected behavior. If, for example, a valve in an engine gets stuck, the diagnostic system should detect this and indicate that something has failed. In diagnostic theory this is called fault detection. If the diagnostic system could point out exactly what part that has failed, in this case the valve, it is called fault isolation

	Faults	
	Valve stuck	Oil leak
Test 1	x	x
Test 2	x	0

Table 2.1: An example of a decision structure for a simple diagnostic system of an engine.

and is usually the aim of a diagnostic system.

In Figure 2.2 a schematic picture of the architecture in a diagnostic system is shown. Given observations, i.e. sensor values and actuator values, the diagnostic system runs a number of diagnostic tests. The tests are checking different parts of the system and returns binary values indicating if the test detected a fault in the system or not, see Figure 2.1. Since each of these tests usually is sensitive to a number of faults, isolation cannot directly be obtained. Therefore the results are sent to a fault isolation function. Using the combined results from the tests the diagnostic system tries to isolate what specific part of the system that has failed and presents this as a diagnostic statement. In Table 2.1 a simple example of fault isolation in a diagnostic system is shown with a decision structure [4]. If a test in the diagnostic system becomes non-zero the structure shows possible explanation. A x in the structure shows what faults a test may react to whereas the number 0 indicate that the test is not sensitive to this specific fault. If for example Test 1 in Table 2.1 becomes non-zero the only possible explanations are that there must be a stuck valve or an oil leak. If both tests signal that they are indicating a fault, the fault isolation part of the diagnostic system will conclude that the valve is stuck.

In order for a diagnostic system to obtain fault indication there needs to be residuals able to indicate when a fault occur. In their simplest form these residuals are just a set of equations that equals zero if the supervised system is non-faulty. For example consider a system modeled as

$$\begin{aligned} x &= u \\ y &= x \end{aligned} \tag{2.1}$$

where u is an actuator signal, y is a sensor signal and x is an internal variable. A residual that checks the correctness of this equation system can then be constructed as $r_1 = y - u$. If the system behavior do not coincide with the system model, the residual will not equal zero and there is said to be a faulty system. As mentioned earlier the main goal in most diagnostic systems is to isolate faults when they occur.

However the problem is to conclude what faults a residual is reacting to. This can be solved by including fault behavior in the model, so-called fault modeling. Assume that there is a need to detect bias faults in sensor y. By adding the fault model of a bias fault θ to the sensor equation in (2.1), the equation system can be written as

$$\begin{aligned} x &= u \\ y &= x + \theta \end{aligned} \tag{2.2}$$

where $\theta = 0$ in the fault free case and $\theta \neq 0$ when there is a bias fault. By using this knowledge it is now possible to construct a residual $r = y - x$. Since $r = \theta$ the test will react when a bias fault is present, i.e. the test is sensitive to bias faults. By using fault models more detailed information about faults are included in the model. In doing this the diagnosis performance may increase i.e. smaller faults may be detected in shorter detection time.

The system (2.2) is overdetermined, i.e. more equations then unknown variables. This is a necessary condition in order for a model to be useful in the construction of residuals, i.e. residual generation. The reason for this is that there must be a way of eliminating the unknown variables from the equations used in a residual. In residual r mentioned above, the unknown signal x can be eliminated using the first equation in (2.2). A property closely related to overdetermined systems this is analytical redundancy [4], formally defined as follows

Definition 2.1 (Analytical Redundancy)**.** *There exists analytical redundancy if there exists two or more ways to determine a variable x by only using the observations z i.e. $x = f_1(z)$ and $x = f_2(z)$, where $f_1(z) \not\equiv f_2(z)$.*

In [4] it also gives that analytical redundancy is a sufficient and necessary condition in order to find residuals in a model. The overdetermined equation system (2.2) contains analytical redundancy since the variable x can be determined by only using the observations u and y. In fact there can be shown that this is the case for all overdetermined systems. Analytical redundancy is an important property since it narrows down the part of the model that can be used for constructing residuals i.e. if there is a part of a model that is non-redundant it cannot be used in residual generation.

2.3 Behavioral Modes

A concept deeply related to fault modeling is behavioral modes. In equation (2.2) knowledge about how the fault θ affects the system was included in the model i.e. a fault model was used. In order to perform

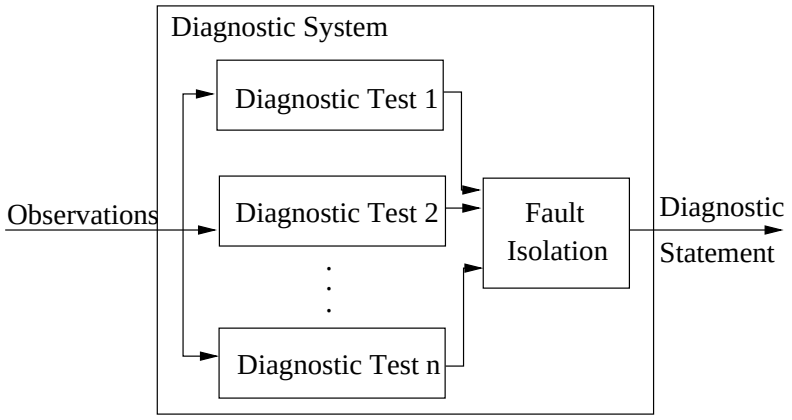

Figure 2.2: The architecture of a diagnostic system.

systematic fault identification there is a need to classify and separate different faults from each other. This can be done using so-called behavioral modes. For example consider the model (2.2) again. Here two behavioral modes can be defined, no fault (NF) and bias fault (B). If $\theta = 0$ the system is in behavioral mode NF and if $\theta \neq 0$ the system is in behavioral mode B. Now assume that the system (2.2) has two sensors y_1 and y_2 both measuring the signal x. Also assume that the first sensor y_1 can sustain a bias fault (B) while the other sensor y_2 is sensitive to a gain fault (G). The bias fault is in this case modeled by the variable θ_1 whereas the gain fault is model by θ_2. The system can then be written as

$$\begin{aligned} x &= u \\ y_1 &= x + \theta_1 \\ y_2 &= \theta_2 x \end{aligned} \tag{2.3}$$

By letting $\theta = [\theta_1\ \theta_2]$ be a fault state vector the behavioral modes can be defined as

Behavioral mode	Fault State
NF	$\theta = [0\ 1]$
B	$\theta = [\theta_1\ 1; \theta_1 \neq 0]$
G	$\theta = [0\ \theta_2; \theta_2 \neq 1]$

The reason for defining these modes is to be able to identify and isolate the corresponding faults when they occur. For example consider the residuals $r_1 = y_1 - x$ and $r_2 = y_2 - x$ of system (2.3). If residual r_1 is non-zero the system is said to be in behavioral mode B whereas residual r_2 is reacting then the system is said to be in behavioral mode G. If none of these tests react, then the system is in behavioral mode NF.

	Behavioral Modes			
	NF	B	G	UF
r_1	0	x	0	x
r_2	0	0	x	x

Table 2.2: Decision structure for system (2.3).

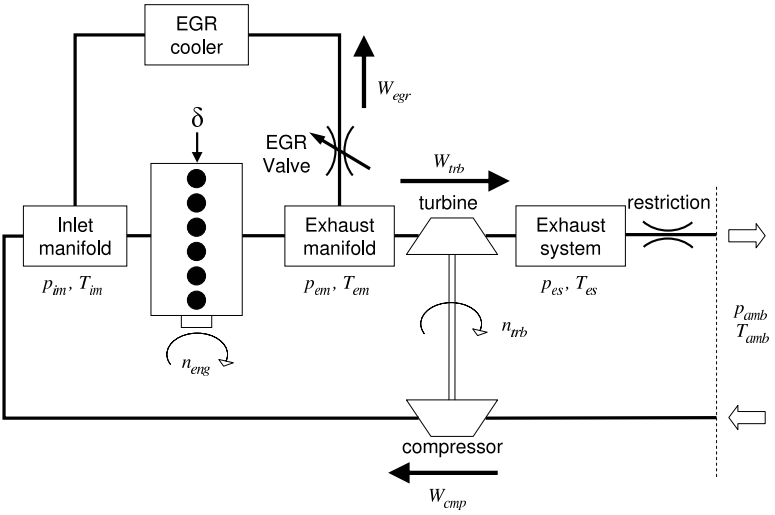

Figure 2.3: Example of a Scania truck engine.

When discussing behavioral modes, fault modes are used to denote all behavioral modes excluding the NF case. The kind of fault modes a model contains depends highly on the type of system. Finding all possible fault modes can be a complex and sometimes an even impossible task. It is therefore often customary to introduce an extra fault mode called unknown fault (UF), which covers all the unmodeled fault types. If no other behavioral mode can explain the behavior, the system is said to have an unknown fault i.e. it is in behavioral mode UF. The whole diagnostic system for (2.3) presented with a decision structure is shown in Table 2.2.

2.4 Structural Models and their Properties

The diagnosis system is in this thesis applied to a Scania truck engine shown in Figure 2.3. The engine model is a large and complex system, containing eight actuator signals, four sensor signals and four states. The whole model is expressed using approximately 280 differential-algebraic equations. By Definition 2.1 there need to be analytical redundancy in order to construct a residual, i.e. there must be a way of eliminating the unknown variables in the equation set used for a residual. For example consider a system where u and y are known signals and $x^3 = u$ and $y = x$. In order derive a residual based on these equations the unknown variable x needs to be eliminated. This is done by first letting $x = \sqrt[3]{u}$ and then constructing the residual as

$$r = y - \sqrt[3]{u} \tag{2.4}$$

Hence all unknown variables can be eliminated and the two equations $x^3 = u$ and $y = x$ can be used to construct a valid residual.

While calculating the simple test in (2.4) one has to compute the cubic-root of a variable, a fairly demanding operation to perform. In the engine model it will not exist any tests of this simple form. In fact experiences have shown that the residuals in a Scania truck engine will include up to 300 equations. Due to combinatorial effects the number of possible tests in the system will be in the range of 1000-10000. To analytically eliminate all the unknown variables in the search for valid residuals, like the one in (2.4), will be highly computational demanding, if even possible. In this thesis this problem is solved by using structural models described in the following section.

2.4.1 Structural Models

The basic idea behind structural models is presented in [6]. Instead of looking on the whole analytical expression of an equation, a less detailed version is used. This is done by just including information of which variables that are present in each equation. For example, consider an equation $e_1 : f(x, z) = 0$, where f is the analytical function, x is an unknown variable and z is a known variable. The structural model corresponding to this equation will just contain information about that variable x and z are included in equation e_1, the analytical expression of function f is ignored in the structural model. Consider the following equation system

$$\begin{aligned} e_1 &: \ \dot{x}_1 = x_2 + 3u \\ e_2 &: \ x_1 = 4x_2^3 + 5 \\ e_3 &: \ y = x_2 \end{aligned} \tag{2.5}$$

where x_i are internal variables, y is a sensor signal and u actuator signal. u and y can be considered to be known signals in this case. By using an biadjacency matrix, see [1], the system (2.5) can be written in its structural form as

equation	unknown			known	
	$\dot{x}_1$	x_1	x_2	y	u
e_1	x		x		x
e_2		x	x		
e_3			x	x	

Table 2.3: DSSM representation of equation system (2.5)

The structural model shown in Table 2.3 is called a Differentiated-Separated Structural-Model (DSSM) [7]. In a DSSM a variable and its derivatives are treated as separate variables, e.g. x_1 and $\dot{x}_1$ in Table 2.3. There exists an even simpler version of a structural model called Differentiated-Lumped Structural-Model (DLSM). In a DLSM a variable and its derivatives are treated as the same variable. This is possible by letting the variables in a DLSM represents functions of time rather then values. The DLSM of (2.5) is shown in Table 2.4. A model of DSSM type contains the most information. However for the application types used in this thesis the information loss of using DLSMs instead of DSSMs has proven to not be substantial.

By using the structural model (2.4) of (2.5) instead of the analytical expressions a much simpler system is obtained. Since structural models can be represented as simple one/zero matrices, the computation time for finding residuals can be significantly reduced compared to using the analytical model.

equation	unknown		known	
	x_1	x_2	y	u
e_1	x	x		x
e_2	x	x		
e_3		x	x	

Table 2.4: DLSM representation of equation system (2.5)

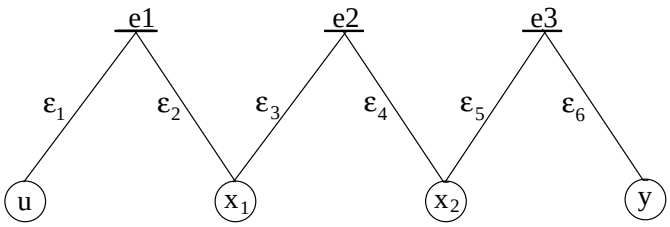

Figure 2.4: Equation system (2.5) as a bipartite graph.

2.4.2 Bipartite Graphs

Like biadjacency matrices mentioned above, bipartite graphs is a way of displaying structural models. The reason for introducing them, is that there already exists a large amount of theory concerning graphs that will be used later in this thesis. In Figure 2.4 the system (2.5) is shown as a bipartite graph. In the graph in Figure 2.4, $e1$, $e2$ and $e3$ corresponds to the equations and are called equation nodes or equation vertices. The circles u, y, x_1 and x_2 represent the variables and are referred to as variable nodes or variable vertices. Between variable nodes and equation nodes are lines, ε_1,.., ε_6, connecting equations to their corresponding variables. In graph theory these lines are called edges. A bipartite graph is in this thesis referred to as $\mathcal{G}(M, X)$, where M is the included equation nodes and X is the variable nodes. The graph in Figure 2.4 would in this representation look like $\mathcal{G}(\{e1, e2, e3\}, \{u, x_1, x_2, y\})$.

2.4.3 Structural Properties

In order to construct a diagnostic system, it is, as mentioned earlier, important to find what part of a model that contain analytical redundancy. To make this analysis easier some structural properties need to be defined. In this section this is done by using some properties of bipartite graphs. First in Definition 2.2-2.4 a graph property called matching is defined. This is later used in Definition 2.5 and Definition 2.6 in order to explain the concepts of graph paths and free nodes. Finally in Definition 2.7-2.9 this is used in order to conclude what parts of a model that can be used in order to construct a diagnostic system. Especially Definition 2.9 will be of interest in this thesis.

Definition 2.2 (Matching Γ_m)**.** *Given a graph with edges* Γ *a matching is a set of edges* $\Gamma_m \subseteq \Gamma$ *such that not two edges have a vertex in common.*

A matching will be denoted Γ_m. In Figure 2.4 the edge set $\{\varepsilon_1, \varepsilon_3\}$ is an example of an matching since they do not have any vertices in common.

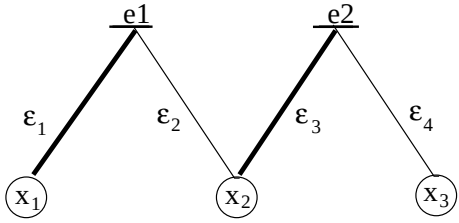

Figure 2.5: An example of a bipartite graph with a matching shown as thicker lines.

Definition 2.3 (Maximal Matching Γ_{mm})**.** *Given a graph with edges Γ a matching Γ_{mm} is a maximal matching if $\Gamma_{mm} \subseteq \Gamma$ and $\Gamma_0 \subseteq \Gamma$ where $|\Gamma_0| > |\Gamma_{mm}|$ implies that Γ_0 is not a matching in the graph.*

A maximal matching will be denoted Γ_{mm}. In Figure 2.4 the edge set $\{\varepsilon_1, \varepsilon_3, \varepsilon_5\}$ is a matching since these edges do not have any vertices in common. Since there only exists three equation nodes, there is no way in which there can exist a matching containing four edges and hence the edge set $\{\varepsilon_1, \varepsilon_3, \varepsilon_5\}$ is also a maximal matching.

Definition 2.4 (Complete Matching Γ_{cm})**.** *Given a bipartite graph $\mathcal{G}(M, X)$, a complete matching Γ_{cm} of M into X is a matching such that all nodes in M is the endpoint of an edge. A complete matching can equally well be a complete matching of X into M.*

A complete matching will be denoted Γ_{cm}. In Figure 2.4 the edge set $\{\varepsilon_1, \varepsilon_3, \varepsilon_5\}$ is an example of a complete matching of the equation set $\{e1, e2, e3\}$ into the variable set $\{u, x_1, x_2, y\}$.

In graph theory a set of connected vertices and the edges joining them are called a path. In Figure 2.5 the edge and node set $\{x_1, \varepsilon_1, e1, \varepsilon_2, x_2\}$ is an example of a path through a part of the graph. An important concept is an alternating path defined as follows

Definition 2.5 (Alternating Path)**.** *Consider a matching Γ_m in a graph with edges Γ. Then an alternating path relative to Γ_m is a path whose edges is alternately in Γ_m and in $\Gamma \backslash \Gamma_m$.*

In Figure 2.5 a matching is shown as thicker lines. The edge and node set $\{x_1, \varepsilon_1, e1, \varepsilon_2, x_2, \varepsilon_3, e2, \varepsilon_4, x_3\}$ is an example of an alternating path in this graph since the edges $\{\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4\}$ is alternately in the matching. In order to define certain properties the concept of free equations and free variables are of use. These can be defined as follows

Definition 2.6 (Free Node)**.** *Given a graph with a matching, a node is said to be free if it is not part of the matching.*

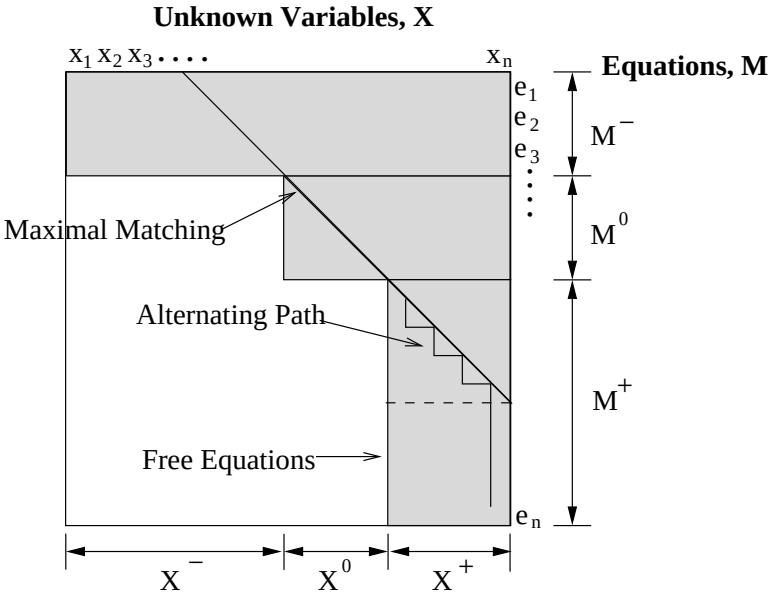

Figure 2.6: Schematic illustration of canonical decomposition on a model M. The gray rectangles represent matrices that may contain non-zero variables whereas all the entries in the white area are zero.

The main objective in this thesis is, as mentioned before, to find equation sets in a system that can be used in the construction of a diagnosis system. By Definition 2.1 the redundant or overdetermined part of a model seems to be of special interest. There is in other words a need to divide a model into different parts and in some way extract the overdetermined one. In [8] it is proven that there always exists a unique structurally overdetermined part in a model, a way to obtain this part called canonical decomposition is also presented. Canonical decomposition can be explained by the following definition

Definition 2.7 (Canonical Decomposition)**.** *Let Γ_{mm} be a maximal matching in the bipartite graph $\mathcal{G}(M, var_X M)$. Denote the equation nodes and variable nodes in Γ_{mm} with M_m and X_m respectively. Then the set of all equation nodes in M such that they are reachable via an alternating path from $M \backslash M_m$ is the structurally overdetermined part M^+. The structurally underdetermined set M^- is the set of all equation nodes in M such that there is an alternating path from $var_X M \backslash X_m$. The remaining part of the model is the structurally justdetermined part denoted M^0. The structurally overdetermined, justdetermined and underdetermined variables are defined as*
$X^+ = var_X M^+$,

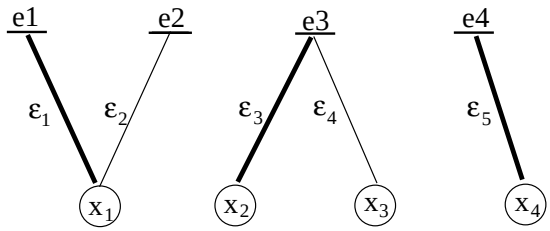

Figure 2.7: Equation system (2.6) shown as a bipartite graph. A maximal matching is marked with thicker lines.

$X^0 = var_X M^0 \backslash X^+$ *and*
$X^- = var_X M^- \backslash (X^+ \cup X^0)$
respectively.

Using Definition 2.7 it is possible to define the relation between the number of equations and the number of variables in M^+, M^- and M^0 as $|M^+| > |X^+|, |M^0| = |X^0|$ and $|M^-| < |X^-|$ respectively.

In Figure 2.6 a schematic illustration of some structural properties is presented graphically. The figure shows the connection between some graph and model properties mentioned above. The gray areas represent matrices that may contain non-zero variables whereas all the entries in the white area zero. For example, this figure shows that equation nodes in M^- may be connected through edges to any variable node in X. Equation nodes in M^+ on the other hand, can only be connected to variable nodes in X^+. The diagonal line in Figure 2.6 shows what equation and variable nodes that are part of the maximal matching. This figure also illustrates the connection between the overdetermined part M^+ of a model and the concept of an alternating path. If a equation node is, as mentioned above, reachable through an alternating path form the maximal matching, it is in the overdetermined part M^+ of the model. In order to illustrate this further consider the following equation system

$$\begin{aligned} e1: &\quad x_1 = u \\ e2: &\quad y = x_1 \\ e3: &\quad x_2 = x_3 + 5 \\ e4: &\quad x_4 = 4 \end{aligned} \tag{2.6}$$

where u and y are known signals. By only considering the unknown signals, the equation system can be represented with the bipartite graph shown in Figure 2.7. In this figure edges part of the maximal matching are marked with thicker lines. According to Definition 2.7 it should be possible to dived the bipartite graph in Figure 2.7 into three dif-

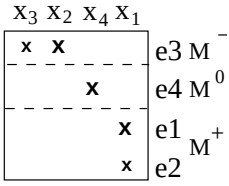

Figure 2.8: Equation system (2.6) divided into three parts by canonical decomposition. Variables and equations nodes included in the maximal matching are marked with bigger x:s.

ferent parts. The overdetermined part is all equation nodes part of an alternating path from equation nodes outside the maximal matching, i.e. equation nodes $e1$ and $e2$. The underdetermined part is all equation nodes part of an alternating path from variable nodes outside the maximal matching, i.e. equation node $e3$. The justdetermined part is the rest of the equation nodes in the graph, i.e. equation node $e4$. In Figure 2.8 this is shown in the same format as the system in Figure 2.6 discussed above, variables and equations nodes included in the maximal matching are marked with bigger x:s.

Since the redundant part of a model is the one used to construct diagnostic tests, models that consist of only overdetermined parts are of special interest. Therefore it is convenient to make the following definitions

Definition 2.8 (Structurally Overdetermined)**.** *An equation set M is said to be Structurally Overdetermined (SO) if $M = M^+$.*

From Definition 2.7 and Definition 2.8 it follows that if M is SO then $M^- = \emptyset$ and $M^0 = \emptyset$.

In the design of a diagnostic system small overdetermined models are of special interest [7]. This follows from the fact that it is in general easier to construct residuals that are based on few equations and hence are sensitive to a few number of faults. One type of sets that are especially important in this sense, are Minimal Structurally Overdetermined sets or MSO sets. In fact in [7] it is shown that the problem of finding all parts of a model that can be used in order to construct residuals can, when using structural models, be reduced to finding all MSO sets (see Chapter 4). MSO sets are formally defined as follows

Definition 2.9 (Minimal Structurally Overdetermined)**.** *A structurally overdetermined set is a Minimal Structurally Overdetermined set (MSO) if none of its proper subsets are structurally overdetermined.*

An overdetermined equation system contains more equations then unknown variables. In order to make it easier to express how many more equations then variables there is in such a system and hence define the degree of redundancy, the following definition is convenient

Definition 2.10 (Degree of Overdeterminedness)**.** *The degree of overdeterminedness for an overdetermined set of equations M^+ is defined as*

$$n = |M^+| - |var_X M^+| \tag{2.7}$$

An overdetermined set of equation with a degree of overdeterminedness n will be denoted M^{+n}, also called a $+n$ system. For example, the overdetermined part M^+ in Figure 2.8 should be denoted M^{+1}, since the set of equations in M^+ got a degree of overdeterminedness equal to one.

Chapter 3

Algorithms for Simulink

At Scania, engine models are implemented in Simulink. In order to find parts, i.e. equation sets, of a model that can be used to construct diagnostic test there is a need, as mentioned before, to perform analytical and structural computations on the model. Necessary calculations can not however be performed directly in a Simulink model, since it is presented graphically. In this chapter some newly developed algorithms for handling analytical and structural transformations will be described, as well as a method of inserting behavioral modes into a Simulink model. First in Section 3.1 a way of transforming a Simulink model into analytical equations is introduced. Then in Section 3.2 the transformation from analytical equations to a structural model is described. Finally in Section 3.3 a solution to the problem of including behavioral modes into a Simulink model is proposed.

3.1 Transforming Simulink Models to Analytical Equations

This section contains a description of an algorithm that retrieves all analytical equations from a Simulink file. Since the engine models, treated in this thesis, are implemented in Simulink this is a necessary step in the process of constructing a diagnostic system. The algorithm can be summarized by the following steps.

Algorithm 3.1.

1. *Simulink Simplification: Structuring the contents in the Simulink file in order to construct a less complex Simulink model.*

2. *Deriving Analytical Equations: Creates an analytical model from the structure given in step (1).*

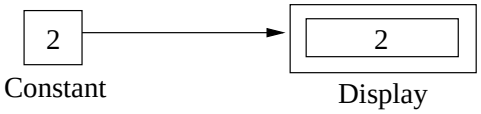

Figure 3.1: Example of a Simulink file(simex.mdl).

3. *Analytical Simplification: Simplifies the analytical model in order to decrease the complexity of the system.*

In step (1) the Simulink file is parsed and the important parts, i.e. parts that contain analytical information, are extracted and transformed into a less complex structure. Step (2) turns the structure created in step (1) into analytical equations. Finally in step (3) an analytical simplification is made in order to reduce the size of the model. A more thorough description of the individual steps in Algorithm 3.1 is given in the sections that follows.

3.1.1 Simulink Simplification

Simulink files are stored as text files. In order to decrease the time spent reading files, there is a need to transform these into a more comprehensible format.

Example 3.1
The Simulink model in Figure 3.1 is in this example first shown as a text file and then transformed into a less complex object oriented structure.

```
Model {
   Name "simex"
   System {
      Name "simex"
      Block {
         BlockType  Constant
         Name       "Constant"
         Value      "2"
      }
      Block {
         BlockType  Display
```

```
         Name       "Display"
         Ports      [1]
      }
      Line {
         SrcBlock   "Constant"
         SrcPort    1
         DstBlock   "Display"
         DstPort    1
      }
   }
}
```

By parsing the file above and extracting the different levels of classes, the model in the file can be transformed to the following, less complex, structure.

```
simex.Constant.Blocktype = Constant
simex.Constant.Name      = "Constant"
simex.Constant.Value     = "2"

simex.Display.Blocktype  = Display
simex.Display.Name       = "Display"
simex.Display.Ports      = [1]

simex.Line.SrcBlock      = Constant
simex.Line.SrcPort       = 1
simex.Line.DstBlock      = "Display"
simex.Line.DstPort       = 1
```

As shown in Example 3.1, Simulink files are built hierarchical. This means that they are on an ideal format to be transform into a sort of object oriented format, with super and subclasses. For example, if a Simulink block named *sum* is contained in a subsystem named *subsystem*1, then it is possible to transform these into a structure like *subsystem*1.*sum*. Hence, an object oriented structure can easily be created by parsing Simulink files and extracting blocks and lines as explained by Example 3.1. By transforming Simulink files into structures the time spent reading files are minimized. Since file access is fairly time demanding, the execution time is hence reduced.

3.1.2 Deriving Analytical Equations from Simulink

In a Simulink file signals are represented as lines (see Figure 3.1). These lines are connected to blocks which, with some exceptions, execute a

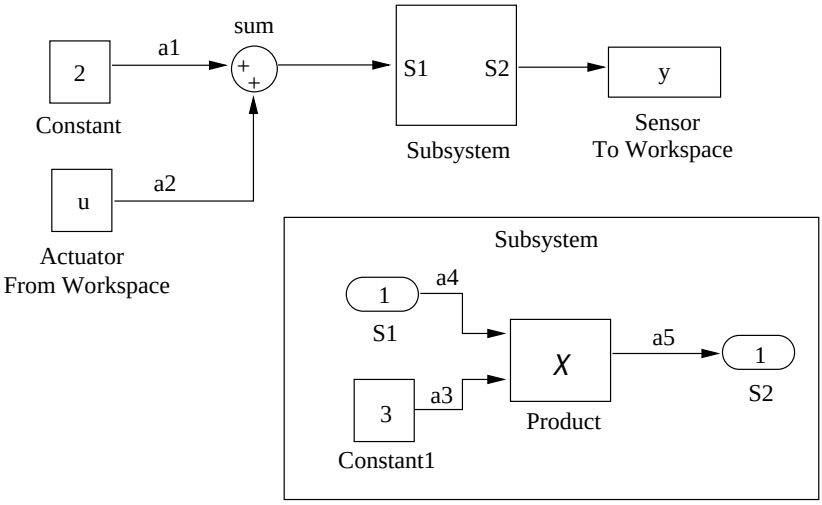

Figure 3.2: Example Simulink model with a subsystem.

computation on the signals. This process coincide with the behavior of a mathematical function, i.e. a block can be compared with a function $Y = f(X)$, where X is a vector of input signals and Y is a vector with output signals. Hence, by defining the analytical function f for each block in a model it should be possible to convert the model into an analytical equation system. If there, for example, exists a summation block, e.g. *sum* in Figure 3.2, there must exist a definition that this block corresponds to equation $S1 = a_1 + a_2$. In this thesis this was solved by using a special definition file that contains the analytical functions for all the block types needed. Another problem is that Simulink models often contain unnamed lines. Since this corresponds to unnamed variables in the equation system, these need to be named before they can be included into a model. This can however be solved automatically by the algorithm and causes no problem. The algorithm can now be written as follows.

Algorithm 3.2.
Input: An object oriented structure S of a Simulink model given from step (1) in Algorithm 3.1.

1. *Let B be the set of all blocks and L be the set of all lines in S.*
2. *For all unnamed lines in L assign a unique variable name.*
3. *Set $i = 1$.*
4. *Select block $b_i \in B$.*

5. *Let X be all input signal names and Y all output signal names for the signals connected to b_i.*
6. *Get the analytical function f corresponding to the block type of b_i.*
7. *Save the analytical function for b_i as $Y = f(X)$ together with a unique equation name.*
8. *If $|B| \neq i$ set $i = i + 1$ and goto step (4), otherwise exit.*

Output: The analytical equations for the Simulink model given from S.

By applying Algorithm 3.2 on the Simulink model in Figure 3.2 the following analytical equation system is obtained.

```
e1 Constant                 : a1 = 2
e2 Actuator From Workspace  : a2 = u
e3 Sum                      : S1 = a1 + a2
e4 Sensor To Workspace      : y  = S2
e5 Subsystem.S1             : a4 = S1
e6 Subsystem.Constant1      : a3 = 3
e7 Subsystem.Product        : a5 = a4 * a3
e8 Subsystem.S2             : S2 = a5
```

Note that in the equation system above the block names for the blocks corresponding to each equation are also shown.

Special Simulink Blocks

In order for the transformation toward an analytical system model to be correct, some special feature blocks need to be defined. These blocks are listed below.

Subsytem Treated as container component. Do not have any block function other then naming signals connected to its ports (see S1 and S2 in Figure 3.2).

To Workspace Treated as sensor signal in the model if the block name is tagged with the letters Sensor (see "Sensor To Workspace" in Figure 3.2).

From Workspace Treated as actuator signal in the model if the block name is tagged with the letters Actuator (see "Actuator From Workspace" in Figure 3.2).

The sensor and actuator blocks are important since they indicate signals which are considered known in the model. This information is later used when transforming the analytical model into a structural model (see Section 3.2).

3.1.3 Analytical Simplification

Structural methods can be an efficient tool when trying to find parts of a model that can be used to construct diagnostic tests (see Chapter 4). This must be kept in mind if an analytical model is simplified, otherwise structural information may be lost. By examining the analytical equations generated by Algorithm 3.2 above, it can be seen that it is possible to simplify analytically. For example, $a5$ can easily be eliminated by replacing it with $S2$ and hence can equation $e8$ be removed. The problem that arises when doing this is to not merge equations that can appear in different residuals. If this is done, information will be lost and it may be impossible to structurally generate all possible tests for a model. For example, consider the following analytical model.

$$\begin{aligned} e_1: &\quad y_1 = x_1 \\ e_2: &\quad y_2 = x_2 \\ e_3: &\quad x_1 = x_2 \\ e_4: &\quad x_1 = u \end{aligned} \tag{3.1}$$

where y_1, y_2 and u are considered to be known variables. By doing a correct, in the analytical sense, simplification this model can be reduced to

$$\begin{aligned} \{e_1, e_4\}: &\quad y_1 = u \\ \{e_2, e_3, e_4\}: &\quad y_2 = u \end{aligned} \tag{3.2}$$

For (3.1) it is possible to structurally extract three residuals $r_1 = y_1 - u$, $r_2 = y_2 - u$ and $r_3 = y_1 - y_2$. However, from the "correct" simplified model it is only possible to structurally construct two residuals as $r_1 = y_1 - u$ and $r_2 = y_2 - u$, hence one residual will be lost. This follows from the fact that the structural methods used in this thesis do not allow known variables to be eliminated. The solution to this problem is to just remove simple assignments of unknown variables from analytical equation systems. By doing this, the model can be reduced without causing the loss of structural information. For example consider the model in (3.1). By letting $x_1 = x_2$ and then replace any occurrence of x_1 with x_2 the model can be reduced to

$$\begin{aligned} e_1: &\quad y_1 = x_2 \\ e_2: &\quad y_2 = x_2 \\ e_3: &\quad x_2 = x_2 \\ e_4: &\quad x_2 = u \end{aligned} \tag{3.3}$$

and hence, equation e_3 can be removed without causing any information loss. In test runs this reduced an engine model with approximately 360 equations to about 170 equations. This will later on be proven to be extremely vital since the processing cost for finding residuals is highly dependent on the model size (see Chapter 4).

3.2 Structural Transformation

Earlier in Section 2.4 it was mentioned that by trying to find residuals in a structural model instead of in an analytical model, the computation time could be significantly reduced. Hence it is essential that equations derived from the Simulink model can be transformed to their structural form. However, this is not difficult since the only information that should be included is which variables each equation contains and if they are consided to be known or unknown.

The analytical equations for the system in Figure 3.2 was presented in Section 3.1.2. Now the equations can be transformed to the following structural model.

equations	unknown							known	
	a1	a2	a3	a4	a5	S1	S2	u	y
e_1	x								
e_2		x						x	
e_3	x	x				x			
e_4							x		x
e_5				x		x			
e_6			x						
e_7			x	x	x				
e_8					x		x		

As mentioned in Section 3.1.2, Sensor tagged `To Workspace` and Actuator tagged `From Workspace` blocks are treated as known signals. Hence variable u and y are shown as known variables in the structural model.

3.3 Behavioral Modes in Simulink

The main objective in this thesis is to extract equations from a model that later can be used to construct diagnostic tests. As explained in Section 2.2 and 2.3 the use of fault models and behavioral modes in a model may increase the possibility for the diagnostic systems to isolate faults, and hence increase the diagnostic performance. Since the models considered in this thesis are implemented in Simulink, which is based on a graphical interface, it would be preferred if the behavioral modes can be included directly in the visual model. This section describes a general solution of how this can be done without changing the model appearance significantly. Note that there is a difference between fault simulation and fault modeling in Simulink. Since the Simulink meth-

Figure 3.3: Example of a Simulink model.

ods described in this thesis only focus on extracting equations from Simulink models, there is no guarantee that the fault model solutions described in this section can run under a Simulink simulation. However, the method proposed in this section has been developed with this in mind and a solution to the simulation problem will not be hard to obtain.

3.4 Fault Modeling in Simulink

When modeling faults graphically in Simulink, enough information must be included into the model so it is possible to extract all behavioral modes. For example, consider the Simulink model in Figure 3.3. The corresponding model can be written as

$$y = u \tag{3.4}$$

where y is a sensor measuring the signal u. Assume now that the sensor has three possible fault models: a constant bias fault (B), a short circuit (SC) and some unknown faults (UF). Let Ω denote the set of all behavioral modes in the system as $\Omega = \{NF, B, SC, UF\}$ where NF denotes the No Fault mode. An example of how this model may look like can be written as follows.

$$\begin{array}{ll} \text{Ass.} & \text{Equation} \\ NF & y = u \\ B & y = u + b \\ SC & y = 0 \\ UF & y \in \mathcal{R} \\ \Omega & \dot{b} = 0 \end{array} \tag{3.5}$$

where Ass. is the behavioral mode assumption. If the system behavior diverge for any equation above the conclusion is that the system is not in the corresponding behavioral mode. For example, if $u \neq 0$ and $y = 0$ then the conclusion will be that that the system is not in the fault free NF state. In order to introduce systems like (3.5) into a Simulink model there must first be a way to include this information graphically. Algorithm 3.2, that transforms Simulink models into analytical equations must be modified in order to connect the right equations to

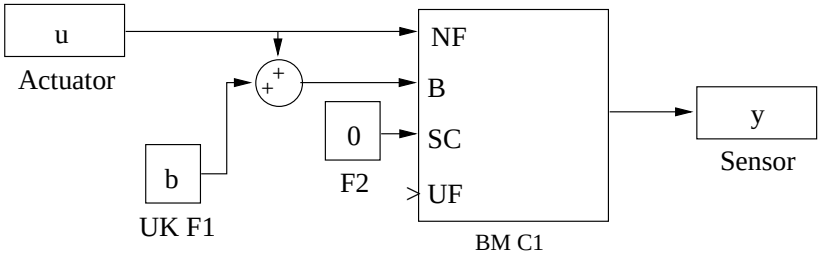

Figure 3.4: The Simulink model in Figure 3.3 with the fault models given in (3.5).

the corresponding behavioral mode. A general way of describing model (3.5) as a Simulink model is shown in Figure 3.4. This solution uses a behavioral mode component ($BM\ C1$ in Figure 3.4) modeled as a subsystem to connect the equations to their corresponding behavioral mode. In order to separate these behavioral mode components from ordinary subsystems they are tagged with the letters BM in the beginning of their names.

When modeling faults there is a need to include information about how unknown fault signals, see b in Figure 3.4, behaves in different states, e.g. if they are static or if they obtain some known value in a specific behavioral mode. For example, in order to model (3.5) properly there must be a way to express that $\dot{b} = 0$. This can be done in many ways, in this thesis it was solved by tagging unknown fault signal blocks with the letters UK in the beginning of their block names and interpret the block types. If a UK tagged fault signal is given by a constant block it is considered to be static, whereas the fault signal is given by a form workspace block it is considered to be dynamic.

The fault modeling solution mentioned above, provides a way of connecting an equation to the corresponding behavioral mode as well as a way to indicate how different fault signals behaves, hence makes it a relatively general solution. Algorithm 3.2 can now be modified according to Algorithm 3.3 in order to manage these new system types.

Algorithm 3.3.
Input: An object oriented structure S of a Simulink model.

1. *Let B be the set of all blocks and L be the set of all lines in S.*
2. *For all unnamed lines in L assign a variable name.*
3. *Set $i = 1$.*

4. *Select block $b_i \in B$.*
5. *Let X be all input signal names and Y all output signal names for the signals connected to b_i.*
6. *If b_i is a constant block tagged with the letters UK giving a signal θ, save $\theta' = 0$ as an analytical function together with a unique equation name.*
7. *If not b_i is a subsystem tagged with the letters BM goto step (9).*
8. *For all $x \in X$ save $x = Y$ as an analytical function together with a unique equation name and the behavioral mode name given from b_i, then goto step (11).*
9. *Get the analytical function f corresponding to the block type of b_i.*
10. *Save the analytical function for b_i as $Y = f(X)$ together with a unique equation name.*
11. *If $|B| \neq i$ set $i = i + 1$ and goto step (4) otherwise exit.*

Output: The analytical equations for S.

The analytical equations for the Simulink model in Figure 3.4 derived by Algorithm 3.3 together with corresponding block types is given by the following model.

```
e1 From Workspace    : a1 = u
e2                   : b' = 0
e3 Constant          : a2 = b
e4 Sum               : a3 = a1 + a2
e5 Subsystem    {NF} : a4 = a1
e6 Subsystem    {B}  : a4 = a3
e7 Subsystem    {SC} : a4 = 0
e8 Subsystem    {UF} :
e9 To Workspace      : y = a4
```

As seen Algorithm 3.3 manage to recreate the model in (3.5) from the Simulink model in Figure 3.4.

Chapter 4

Structural Algorithms

As explained in Chapter 2, minimal structurally overdetermined sets are of special interest when designing diagnostic tests, since they are in general small and sensitive to few faults. In [7], it is shown that by using the MSO sets contained in a model it is possible to construct sufficiently accurate diagnostic system. Hence, in order to construct a solid diagnostic system, just using structural models, it is essential that all MSO sets are found. In [7], an algorithm for extracting all MSO sets from a system model is presented. However, some of the steps in this algorithm are highly ineffective and computational expensive for the kind of models considered in this thesis. This chapter presents newly developed algorithms for extracting all MSO sets contained in a structural model. In Section 4.1 the main objectives for the new algorithm are given. In Section 4.2 and 4.3 a more detailed description of the different steps in the algorithm are presented. In Section 4.4 a partially changed algorithm for finding all MSO sets in a system containing behavioral modes is shown. Finally in Section 4.5 the correctness of the new algorithms is proved. The input to all these algorithms is a structural model extracted from the model equations of the engine, see Section 3.2.

4.1 Steps Toward Finding All MSO Sets

In [7] Algorithm 8.1 introduces a way of finding all MSO sets in a structural model. In this section a new improved and partially changed version of this algorithm is presented. The algorithm can be described by the following structure

Algorithm 4.1.
Input: A structural model M.

1. *Removing derivatives: This algorithm does not separate a variable from its derivatives, hence the structural model M must be transformed to a DLSM (see Section 2.4.1).*

2. *Extracting the overdetermined part of the model: Since MSOs can not contain any justdetermined or underdetermined parts, these are removed (see Definition 2.8 and Definition 2.9).*

3. *Structural Simplification: Simplifies the model by combining equation sets that have to be used together in an MSO set. This produces a less complex model, fewer equations and unknown variables, and hence decreases the computing time in the next step.*

4. *Finding all MSO sets: Searching the structural model obtained from the last step and extracting all MSO sets.*

Output: All MSO sets contained in the model M.

As mentioned above Algorithm 4.1 is a modified version of an algorithm presented in [7]. However the results given from the two algorithms are the same. Compared to the algorithm in [7] it is mainly step (3) and step (4) of Algorithm 4.1 that have been improved. Hence, in the following sections a more thorough description of these steps will be given. For more information concerning the other steps see [7].

4.2 Structural Simplification

As mentioned earlier in Section 2.4, finding all diagnostic tests for a model is a complex task. This is still the case when using structural methods. Hence every simplification of the structural model is helpful. In [7] a structural simplification algorithm is introduced. The algorithm creates a simpler model by merging those equations that must be in the same MSO. The merging is achieved by looking for unknown variables that are present in only two equations. If such a variable is found the two equations are combined and the variable is removed. This can be described according to the following

Algorithm 4.2.
Input: A structural overdetermined model M'

1. *Set $X' = var_X M'$.*

2. *For all variables $x \in X'$ do step (3).*

3. *If $|equM'x| = 2$ merge the equations in $equM'x$ and remove the variable x.*

4. *If any equations was merged in step (3) goto step (1) otherwise exit.*

Output: The simplified model M'.

This algorithm can be illustrated through a simple example. Consider the following analytical equation system

$$\begin{aligned} e_1 : &\quad y_1 = x_1 + u_1 \\ e_2 : &\quad y_2 = x_1 + u_2 \end{aligned} \tag{4.1}$$

where the variables y_i and u_i are known while x_1 are unknown. In this case the unknown variable x_1 is present in only two equations. Hence, according to the Algorithm 4.2 the two equations can be merged and the variable x_1 removed. By doing this the following equation system is derived

$$\{e_1,\ e_2\} : \quad y_1 - y_2 = u_1 - u_2 \tag{4.2}$$

The algorithm repeats these steps until no more simplification is possible. By applying this on structural models it produces an effective simplification. However Algorithm 4.2 is not able to handle all structural cases. Consider the structural model describing an MSO in Table 4.1. In the first run with Algorithm 4.2 variable x_3 is found to be present in only two equations, e_3 and e_4. Hence x_3 is removed and equations e_3 and e_4 merged together. The result is shown in Table 4.2. Since there not exists any more variables contained in only two equations Algorithm 4.2 will now exit.

equation	unknown			known	
	x_1	x_2	x_3	y	u
e_1	x	x		x	
e_2	x	x			x
e_3	x	x	x	x	
e_4			x		x

Table 4.1: An example of a structural model before the simplification step.

In Table 4.1 all equations can be merged together since all equations is needed to remove the unknown variables in the system, and hence must be used together in an MSO. However, Table 4.2 clearly shows that Algorithm 4.2 fails to find all possible ways of merging equations and simplifying the model. This problem can be solved by constructing a new simplification algorithm that uses the properties of canonical

equation	unknown		known	
	x_1	x_2	y	u
e_1	x	x	x	
e_2	x	x		x
$\{e_3, e_4\}$	x	x	x	x

Table 4.2: The structural model in Table 4.1 after the simplification made by Algorithm 4.2.

decomposition in the merging part. By Definition 2.7 a canonical decomposition divides a model into three different parts. One structurally overdetermined part M^+, one structurally justdetermined part M^0 and one structurally underdetermined part M^-. Since MSO sets by Definition 2.9 are overdetermined they can not contain any underdetermined or justdetermined parts. This knowledge can be used in order analyze what equations that must be grouped together when constructing an MSO.

The basic idea behind this algorithm is to remove an equation from a structurally overdetermined model and then, with the help of canonical decomposition, analyze if this produce any underdetermined or justdetermined part. If this is the case the removed equation must always be used together with the equations in the obtained underdetermined or justdetermined part, if included in an MSO. Consider a structurally overdetermined model $M = \{e_1, e_2, e_3, e_4\}$ with the structure shown in Table 4.3.

equation	unknown	
	x_1	x_2
e_1	x	x
e_2	x	x
e_3	x	
e_4	x	

Table 4.3: An example of a structural model.

If, for example, equation e_1 is removed from M the canonical decomposition will divide the model into $M^+ = \{e_3, e_4\}$ and $M^0 = \{e_2\}$. The reason for this is that with e_1 removed there is no way in which the unknown variable x_2 can be eliminated from the remaining system. In fact the equations e_1 and e_2 must always be used together in order to eliminate x_2 from the model i.e. if any of e_1 or e_2 is included in an MSO both equations must be included. Hence these equations can just as well be merged together and variable x_2 removed. The whole

equation	known	
	y	u
$\{e_1, e_2, e_3, e_4\}$	x	x

Table 4.4: The structural model in Table 4.1 after simplification made by Algorithm 4.3.

algorithm can now be written as

Algorithm 4.3.
Input: A structurally overdetermined set of equations M.

1. *Set $i = 1$.*
2. *Select an equation $e_i \in M$.*
3. *Set $M' = M \backslash \{e_i\}$.*
4. *Do a canonical decomposition of M' to get M'^{+}, M'^{0} and M'^{-}.*
5. *Merge the equations in $M \backslash M'^{+}$ and let them replace equation e_i in M.*
6. *Remove the variable sets X^0 and X^- from M.*
7. *If $i \neq |M|$ set $i = i + 1$ and goto step (2) otherwise exit.*

Output: The simplified model M.

The basic idea behind Algorithm 4.3 is shown in Figure 4.1. Here M is a structurally overdetermined set of equations containing the set of variables X. If equation e_1 is removed as shown in the figure, the equations e_1, e_2 and e_3 will be merged and the variable sets X^0 and X^- removed by Algorithm 4.3. The result of using Algorithm 4.3 to simplify the structural model in Table 4.1 is presented in Table 4.4. It can be seen that this algorithm manage to simplify the whole structure to its minimal form.

4.3 Finding All MSO Sets

This step is the most important one in Algorithm 4.1 and also the most complex problem to solve. The MSO algorithm in [7] is, as mentioned above, quite ineffective and this step is actually the main cause of this inefficacy. In order to find equation sets that could be used to construct an MSO the algorithm in [7] performs a full tree search through every equation and variable of the model, searching for sets of equation where all unknown variables can be eliminated. For large systems this leads

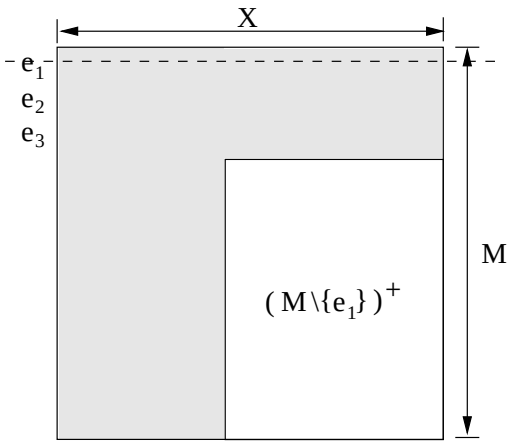

Figure 4.1: Schematic illustration of an equation system with variable set X and equation set M. If equation e_1 is removed the overdetermined part of the model will be the white area and hence equation e_1, e_2, e_3 will be merged.

to an enormous amount of computations in order to find the equation sets. The new algorithm described in this section uses some properties of MSO sets that reduce the computational cost to a minimum.

4.3.1 Basic Algorithm

In [7] the following Lemma and its proof are given

Lemma 4.1. *The set of equations M is an MSO set if and only if M is structurally overdetermined and $|M| = |var_X M| + 1$.*

Note that this is by Definition 2.10 a M^{+1} system. Consider now that there exists an overdetermined set of equations $M = M^{+4}$ i.e. if there are n unknown variables, the number of equations is $n + 4$. By removing equations a less overdetermined model is created from this system. In fact, for every equation removed from an overdetermined set of equations the degree of overdeterminedness decreases with one, for the proof of this see Lemma 4.4. This means that by removing an equation and extracting the overdetermined part from M the system will become a M^{+3} system. By repeating this procedure three times we will get a M^{+1} system. As given by Lemma 4.1 above this is an MSO. Now let E denote the set of removed equations from M^{+4}, i.e. it will consist of three equations. By removing different equations from M^{+4} it is possible to get different MSOs. Hence by selecting all combinations of equations E to be removed, it is possible to derive all MSOs from

M^{+4}. This can be formalized by the algorithm that follows below. The input to this algorithm is the simplified model derived by Algorithm 4.3.

Algorithm 4.4.
Input: A structurally overdetermined set of equations $M^{+n} = \{e_1, .., e_n\}$.

1. *Set* $i = 1$.
2. *Select an equation* $e_i \in M^{+n}$.
3. *Let* $M^{+(n-1)} = (M^{+n} \backslash \{e_i\})^+$.
4. *If* $n - 1 \neq 1$ *call this algorithm with* $M^{+(n-1)}$.
5. *If* $n - 1 = 1$ *save* $M^{+(n-1)}$ *as an MSO.*
6. *If* $i \neq |M^{+n}|$ *set* $i = i + 1$ *and goto step (2) otherwise exit.*

Output: All MSO sets in M^{+n}.

Algorithm 4.4 performs a systematical reduction of the equation set M^{+n} until all M^{+1} systems are derived. Step (2) chooses an equation from M^{+n} that is used in step (3) to decrease the degree of overdeterminedness n with one. The equation set obtained is according to Definition 2.10 a $+(n-1)$ system, hence the chosen notation M^{+n-1}. If an MSO is derived it is saved and the algorithm chooses the next equation in the set, otherwise the algorithm calls itself with the new $M^{+(n-1)}$ system. In Example 4.1 a run with this algorithm is shown.

Example 4.1
Consider the following structurally overdetermined model consisting of four equations and two unknown variables

equation	unknown	
	x_1	x_2
e_1	x	
e_2	x	x
e_3		x
e_4		x

Input to the algorithm: $M^{+2} = \{e_1, e_2, e_3, e_4\}$

Iteration 1:
Step (1-3): $M^{+1} = (M^{+2} \backslash \{e_1\})^+ = \{e_3, e_4\}$.
Step (4-5): Since this is a M^{+1} system save $\{e_3, e_4\}$ as an MSO.
Step (6) : Since $i = 1 \neq 4 = |M^{+2}|$ goto step (2).

Iteration 2:
Step (2-3): $M^{+1} = (M^{+2} \backslash \{e_2\})^+ = \{e_3, e_4\}$.
Step (4-5): Since this is a M^{+1} system save $\{e_3, e_4\}$ as an MSO.
Step (6) : Since $i = 2 \neq 4 = |M^{+2}|$ goto step (2).

Iteration 3:
Step (2-3): $M^{+1} = (M^{+2} \backslash \{e_3\})^+ = \{e_1, e_2, e_4\}$.
Step (4-5): Since this is a M^{+1} system save $\{e_1, e_2, e_4\}$ as an MSO.
Step (6) : Since $i = 3 \neq 4 = |M^{+2}|$ goto step (2).

Iteration 4:
Step (2-3): $M^{+1} = (M^{+2} \backslash \{e_4\})^+ = \{e_1, e_2, e_3\}$.
Step (4-5): Since this is a M^{+1} system save $\{e_1, e_2, e_3\}$ as an MSO.
Step 6 : Since $i = 4 = |M^{+2}|$ exit.

The output from this algorithm is all derived MSO sets, i.e. $\{e_3, e_4\}$, $\{e_3, e_4\}$, $\{e_1, e_2, e_4\}$ and $\{e_1, e_2, e_3\}$. Note that the MSO set $\{e_3, e_4\}$ was obtained two times.

4.3.2 Improvements

The basic Algorithm 4.4 is not optimal since it for some systems may do the same thing multiple times. However Algorithm 4.4 can be improved and the performance increased for these types of systems.

Consider an overdetermined equation system M^{+n} with k number of equations i.e. $|M^{+n}| = k$. The number of iterations executed by Algorithm 4.4 can then be calculated as

$$\begin{pmatrix} k \\ n-1 \end{pmatrix} = \frac{!k}{(n-1)!(k-(n-1))!} \tag{4.3}$$

Large systems, i.e. a high k, with high redundancy n will seemingly result in a large number of iterations. However, by using information about these systems, some improvements can be made to the algorithm in order to elevate performance. Large systems often result in that the same overdetermined subsystem is produced in different parts of the algorithm. This can even be seen in the small Example 4.1 where the

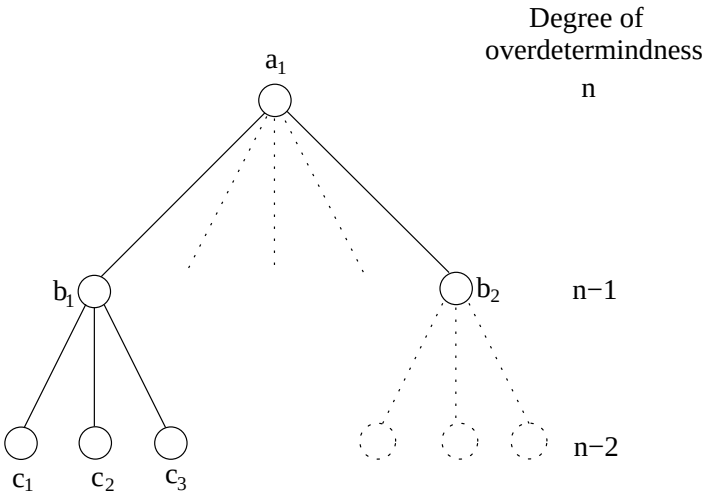

Figure 4.2: A part of an execution with Algorithm 4.4 represented as a tree, where each node represents a set of equations M^{+n}.

MSO $\{e_3, e_4\}$ was derived two times. Figure 4.2 describes a run with the algorithm presented as an execution tree, where each node represents a set of equations M^{+n}. Assume now that the equation set b_1 in the left most path is the same as b_2 in the right most path in Figure 4.2. Since these are the same it is obvious that they will result in the same children in the execution tree, in this case c_1, c_2 and c_3. Hence, it is unnecessary to continue the execution in the right path after node b_2. This can be solved by keeping track of the equation sets produced at each level in the tree and hence avoid any identical execution.

Another improvement uses the same principle as the new structural simplification step described in section 4.2. In step (3) of Algorithm 4.4, an equation e_i is removed and a new overdetermined part is extracted as

$$M^{+(n-1)} = (M^{+n} \backslash e_i)^+ \tag{4.4}$$

Let $E = M^{+(n-1)} \backslash M^{+n}$. According to Lemma 4.6 an equivalent E will be derived as long as $e_i \in E$ in (4.4). Hence, it is unnecessary to select more than one equation $e_i \in E$, since this will produce the same result. By keeping track of equations that will produce equivalent results in step (3) of Algorithm 4.4 the number of iterations will be reduced. This simplification step uses the same theory as the structural simplification Algorithm 4.3. Hence, there is no need to do any structural simplification before this new improved MSO algorithm is executed.

Let $\mathcal{M}_{so}$ be a set containing structurally overdetermined equation sets, the improved algorithm will then look as follows.

Algorithm 4.5.
Input: A structurally overdetermined set of equations $M^{+n} = \{e_1, .., e_n\}$..

1. $\mathcal{M}_{so} = \emptyset$.
2. *Let* $E = M^{+n}$.
3. *Select an equation* $e \in E$.
4. *Let* $M^{+(n-1)} = (M^{+n} \backslash e)^+$.
5. *Set* $E = E \backslash (M^{+n} \backslash M^{+(n-1)})$.
6. *If* $M^{+(n-1)}$ *already exists in* $\mathcal{M}_{so}$ *goto step (3) otherwise let* $\mathcal{M}_{so} = \mathcal{M}_{so} \cup \{M^{n-1}\}$.
7. *If* $n - 1 \neq 1$ *call this algorithm with* $M^{+(n-1)}$.
8. *If* $n - 1 = 1$ *save* $M^{+(n-1)}$ *as an MSO.*
9. *If* $E \neq \emptyset$ *goto step (2) otherwise exit.*

Output: All MSO sets in M^{+n}.

In Example 4.2 a run with Algorithm 4.5 on the same model as in Example 4.1 is shown.

Example 4.2
Consider the structurally overdetermined model in Example 4.1 again.

equation	unknown	
	x_1	x_2
e_1	x	
e_2	x	x
e_3		x
e_4		x

Input to the algorithm: $M^{+2} = \{e_1, e_2, e_3, e_4\}$

Iteration 1:
Step (1) : $E = \{e_1, e_2, e_3, e_4\}$.
Step (2) : Select an equation from $E = \{e_1, e_2, e_3, e_4\}$.
Step (3) : $M^{+1} = (M^{+2} \backslash \{e_1\})^+ = \{e_3, e_4\}$.
Step (4) : $E = E \backslash (M^{+2} \backslash \{e_3, e_4\}) = \{e_3, e_4\}$.
Step (5) : Since M^{+1} has not been obtained earlier goto step (6).
Step (6-7): Since this is a M^{+1} system save $\{e_3, e_4\}$ as an MSO.
Step (8) : Since $E \neq \emptyset$ goto step (2).

Iteration 2:
Step (2) : Select an equation from $E = \{e_3, e_4\}$.
Step (3) : $M^{+1} = (M^{+2} \backslash \{e_3\})^+ = \{e_1, e_2, e_4\}$.
Step (4) : $E = E \backslash (M^{+2} \backslash \{e_1, e_2, e_4\}) = \{e_4\}$.
Step (5) : Since M^{+1} has not been obtained earlier goto step (6).
Step (6-7): Since this is a M^{+1} system save $\{e_1, e_2, e_4\}$ as an MSO.
Step (8) : Since $E \neq \emptyset$ goto step (2).

Iteration 3:
Step (2) : Select an equation from $E = \{e_4\}$.
Step (3): $M^{+1} = (M^{+2} \backslash \{e_4\})^+ = \{e_1, e_2, e_3\}$.
Step (4) : $E = E \backslash (M^{+2} \backslash \{e_1, e_2, e_3\}) = \{\}$.
Step (5) : Since M^{+1} has not been obtained earlier goto step (6).
Step (6-7): Since this is a M^{+1} system save $\{e_1, e_2, e_3\}$ as an MSO.
Step (8) : Since $E = \emptyset$ exit.

The output from this algorithm is all derived MSO sets, i.e. $\{e_3, e_4\}$, $\{e_1, e_2, e_4\}$ and $\{e_1, e_2, e_3\}$.

As seen in Example 4.2 Algorithm 4.5 manage to avoid the double execution made by Algorithm 4.5 in Example 4.1. The number of iterations is, in this example, reduced with 25%. In larger models the improvements in Algorithm 4.5 shows to have an even bigger impact on the number of iterations and hence reduces the execution times dramatically (see Chapter 5).

4.4 Finding All MSO Sets in a Behavioral Mode System

The insertion of behavioral modes into a model is not uncomplicated. New problems arise such as increased execution time and the production of invalid MSO sets. As explained in Section 3.3, behavioral modes in this thesis appear in a system model as new equations, i.e. each be-

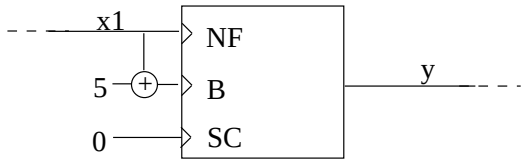

Figure 4.3: Example of a behavioral mode component.

havioral mode correspond to one equation in the system model. For example, the behavioral mode component shown in Figure 4.3 can be translated into

$$\begin{array}{ll} NF: & y = x_1 \\ B: & y = x_1 + 5 \\ SC: & y = 0 \end{array} \tag{4.5}$$

Equations connected to a specific behavioral mode component, like the ones in (4.5), are supposed to be valid in separate fault modes. If for example the component in Figure 4.3 is in a fault free environment the first equation in (4.5) is valid, whereas if the component got a bias fault of 5 the second equation is the correct one. This means that there should not be possible to include two or more equations from the same behavioral mode component in an MSO. In the present form the MSO Algorithm 4.5 does not prevent this. Hence, it will produce some invalid MOS sets if it is executed on a model, containing behavioral modes equations like (4.5). This can easily be solved by simply removing all invalid MSO sets after the algorithm has finished. However, since new equations containing the same variables are included into the model, when new behavior modes are added, this will make the model more redundant. As explained in Section 4.3.2 the number of iterations performed by Algorithm 4.5 is highly dependent on the redundancy of the system. Introducing behavioral modes into the system will cause the execution time to increase significantly. This problem is solved by tagging the equations connected to a specific behavioral mode component with the name of that behavioral mode component. Algorithm 4.5 can then be modified to disallow two equations with identical tags to be included into the same MSO. Let $equ_{C_i}(M^{+n})$ be the equations in a structurally overdetermined set of equations M^{+n} tagged with the behavioral mode component C_i. Note that this is an extension of the $equ_X(M)$ notation motioned earlier. The modified version of Algorithm 4.5 can then be written as follows.

Algorithm 4.6.
Input: A structural overdetermined set of equations M^{+n}.

1. *If there exists any i such that $|equ_{C_i}(M^{+n})| > 1$ select one such C_i and let $E = equ_{C_i}(M^{+n})$, else let $E = M^{+n}$.*

2. *Select an equation* $e \in E$.
3. *Let* $M^{+(n-1)} = (M^{+n} \backslash \{e\})^{+}$.
4. *Set* $E = E \backslash (M^{+n} \backslash M^{+(n-1)})$.
5. *If* $M^{+(n-1)}$ *already exist in the MSO-structure goto step (2).*
6. *If* $n - 1 \neq 1$ *call this algorithm with* $M^{+(n-1)}$.
7. *If* $n - 1 = 1$ *and there not exists any* i *such that* $|equ_{C_i} M^{+n}| > 1$ *save* $M^{+(n-1)}$ *as an MSO.*
8. *If* $E \neq \emptyset$ *goto step (2).*

Output: All MSO sets in M^{+n}.

In Algorithm 4.6 step (1), (2) and (8) have been changed or added in comparison to Algorithm 4.5. Step (1) and (2) checks if there are two or more equations with identical behavioral mode component tags in the model. If this is the case one of these equations is removed. Due to the systematical reduction of the model this will lead to that every possible combination of behavioral mode equations will be included in the MSOs as long as there never is more than one with a specific tag. Note that Algorithm 4.6 is not general since it assumes that all behavioral modes can be expressed with one equation. This may not necessarily be the case when concerning models not generated thought the algorithms described in this thesis. However, Algorithm 4.6 can easily be extended to handle all model types. Algorithm 4.6 shows to be effective even in highly redundant behavioral mode systems. Test runs of large engine models containing behavioral modes, shows that Algorithm 4.6 finds all MSO sets more than ten times faster than Algorithm 4.5.

The algorithms presented in this chapter require some new properties of structural models that has not previously been proved. In the following section some newly developed lemmas and proofs are presented in order to analyze the correctness of the algorithms. This part of the thesis is however quite theoretical and is possible to omit.

4.5 The Correctness of the Algorithms

In order to prove the correctness of Algorithm 4.1, in particular the the simplification step described in Section 4.2 (see Lemma 4.6) and the MSO algorithm presented in Section 4.3 (see Theorem 4.1), a few supporting lemmas need to be defined.

The following lemma provides a connection between the matching property of bipartite graphs and structurally overdetermined models, see Section 2.4.

Lemma 4.2. *If an equation set M is structurally overdetermined then there exists a complete matching Γ_{cm} of $var_X M$ into M in $\mathcal{G}(M, var_X M)$.*

Proof. Consider a structurally overdetermined set of equations M and assume that there is no complete matching of $var_X M$ into M in $\mathcal{G}(M, var_X M)$. Now for any maximal matching Γ_{mm} in $\mathcal{G}(M, var_X M)$ there must exist a free variable vertex v. Then for all $e \in equ_M(v)$ a path $P(v, e)$ is an alternating path in $\mathcal{G}(M, var_X M)$. Hence by Definition 2.7 it follows that $e \in M^-$, i.e. $M^- \neq \emptyset$. According to Definition 2.8 this contradicts the assumption that M is structurally overdetermined. □

In the next lemma an important consequence of structurally overdetermined equation sets is given

Lemma 4.3. *Let M be a set of equations. If $M' \subseteq M$ is an SO model then $M' \subseteq M^+$.*

Proof. Consider an arbitrary structurally overdetermined set of equations $M' \subseteq M$. From Lemma 4.2 it follows that there must exist a complete matching Γ_{cm} of $var_X M'$ into M' in $\mathcal{G}(M', var_X M')$. This means that $equ(\Gamma_{cm}) \subseteq M'$ and $var(\Gamma_{cm}) \subseteq var_X M'$. Set $X' = var_X M \backslash var_X M'$ and let Γ_{mm} be a maximum matching in $\mathcal{G}(M \backslash M', X')$. Then it follows that $equ(\Gamma_{mm}) \subseteq (M \backslash M')$ and $var(\Gamma_{mm}) \subseteq X'$. From the definition of X' this gives that $X' \cap var_X M' = \emptyset$ which implies that $var(\Gamma_{cm}) \cap var(\Gamma_{mm}) = \emptyset$, i.e. the node sets of Γ_{cm} and Γ_{mm} are disjunct. From this it follows that $\Gamma_{cm} \cup \Gamma_{mm}$ will be a matching in $\mathcal{G}(M, var_X M)$. Since $var_{X'} M' = \emptyset$ and Γ_{cm} is maximal in $\mathcal{G}(M', var_X M')$ and Γ_{mm} is maximal in $\mathcal{G}(M \backslash M', X')$, this implies that $\Gamma_{cm} \cup \Gamma_{mm}$ will be a maximal matching in $\mathcal{G}(M, var_X M)$.

Now take an arbitrary equation $e \in M'$. From Definition 2.8 and given that Γ_{cm} is a maximal matching in $\mathcal{G}(M', var_X M')$, it follows that there must exist an alternating path P in $\mathcal{G}(M', var_X M')$ between e and a free equation $e' \in M'$. From Definition 2.7 and given that $\Gamma_{cm} \cup \Gamma_{mm}$ is a maximal matching in $\mathcal{G}(M, var_X M)$, this also implies that P will be an alternating path in $\mathcal{G}(M, var_X M)$. Furthermore since $e' \in M'$ is a free equation in $\mathcal{G}(M', var_X M')$ w.r.t. Γ_{cm} and given that the node sets of Γ_{cm} and Γ_{mm} are disjunct, it follows that e' is also a free equation w.r.t. $\Gamma_{cm} \cup \Gamma_{mm}$ in $\mathcal{G}(M, var_X M)$. From this, the properties of P and Definition 2.7 it follows that $e \in M^+$. Since e was arbitrary

chosen this implies that every equation in M' is a part of M^+ and hence $M' \subseteq M^+$. □

One of the most important features in Algorithm 4.4 is included in step (3), namely if one equation is removed from a structural overdetermined set of equations, the degree of overdeterminedness will decrease with exactly one. This is formalized by the following lemma

Lemma 4.4. *If M is a structurally overdetermined set of equations with the degree of overdeterminedness n and $e \in M$, then $(M^{+n}\backslash\{e\})^+$ has degree of overdeterminedness $n-1$.*

Proof. Consider the structurally overdetermined set of equations M. From Lemma 4.2 it follows that there exist a complete matching Γ_{cm} of $var_X M$ into M in $\mathcal{G}(M, var_X M)$. By Definition 2.3 it follows that Γ_{cm} also is a maximal matching. According to theorem 5.1.7 in [1] a new maximal matching is obtained with one sequence of transfers along an alternating path if the path consists of an even number of edges. Definition 2.7 gives that a free equation node is always reachable from any equation in M via an alternating path in $\mathcal{G}(M, var_X M)$. This means that for any equation $e \in M$ it is possible to find a path P in $\mathcal{G}(M, var_X M)$ where $|equ(P)| = |var(P)| + 1$. This gives that

$$|P| = |var(P)| + |equ(P)| - 1 = 2|var(P)| \tag{4.6}$$

i.e. for all equations in a SO set of equations it is always possible to find a path with an even number of edges. From (4.6) and theorem 5.1.7 in [1] it now follows that if an equation $e \in \Gamma_{mm}$ in a graph $\mathcal{G}(M, var_X M)$ where M is SO, then it is possible to construct a new maximal matching Γ'_{mm} so that $e \notin \Gamma'_{mm}$. This together with (4.6) gives that there exist a complete matching of $var_X M$ into $M\backslash\{e\}$ where $e \in M$. By Definition 2.7 this yields that $(M\backslash\{e\})^- = \emptyset$ i.e.

$$M' = M'^+ \cup M'^0 \tag{4.7}$$

where $M' = (M\backslash\{e\})$. Consider now that M^+ is an M^{+n} system and M'^+ is an M'^{+m} system. By using Definition 2.8, Definition 2.10 and (4.7) the degree of overdeterminedness for M'^{+m} can be calculated as

$$m = |M'^+| - |var_X M'| = |M| - |var_X M| - |\{e\}| - |M'^0| + |var_X M'^0| \tag{4.8}$$

Definition 2.7 gives that $|M'^0| = |var_X M'^0|$ which yields that $m = n-1$ since $|M| - |var_X M| = n$. From this it follows that $M'^{+m} = (M^{+n}\backslash\{e\})^+ = M^{+(n-1)}$. □

In the next lemma and corollary some characterizations of the degree of overdeterminedness in structural overdetermined sets of equations will be presented.

Lemma 4.5. *Let M_0^{+m} and M_1^{+n} be two overdetermined set of equations where $M_0^{+m} \subseteq M_1^{+n}$ then it follows that $m \leq n$.*

Proof. Consider the set of equations $E = M_1^{+n} \backslash M_0^{+m}$. If $E = \emptyset$ then $M_1^{+n} = M_0^{+m}$ and hence $n = m$, otherwise let $M_2 = M_1^{+n} \backslash \{e\}$ where $e \in E$. Since E and M_0^{+m} are two disjunctive sets and $M_0^{+m} \subseteq M_1^{+n}$ it follows that

$$M_0^{+m} \subseteq M_2 \tag{4.9}$$

Lemma 4.3 and (4.9) now gives that

$$M_0^{+m} \subseteq M_2^{+k} \tag{4.10}$$

Using Lemma 4.4 on (4.10) it also follows that

$$M_2^{+k} = (M_1^{+n} \backslash \{e\})^+ \tag{4.11}$$

where $k = n - 1$. Expression (4.10) and (4.11) implies that

$$M_0^{+m} \subseteq M_2^{+(n-1)} \tag{4.12}$$

Since E and M_0^{+m} are tow disjunctive sets it is possible to remove all equations in E and from M_0^{+m} and still get (4.12). Hence, $E \neq \emptyset$ gives that $m < n$. Since $E = \emptyset$ implies that $m = n$ it follows that $m \leq n$. □

Corollary 4.1. *Let M_0^{+m} and M_1^{+n} be two overdetermined set of equations where $M_0^{+m} \subseteq M_1^{+n}$ and $n = m$ then it follows that $M_0^{+m} = M_1^{+n}$.*

Proof. Let $M_0^{+m} \subseteq M_1^{+n}$ where $n = m$ and consider the set of equations $E = M_1^{+n} \backslash M_0^{+m}$. If not $M_0^{+m} = M_1^{+n}$ then it follows that $E \neq \emptyset$. By using the same discussion as in the proof of Lemma 4.5 it is then possible to show that $M_0^{+m} \subseteq M_1^{+(n-1)}$ and hence $m < n$. This contradicts the assumption that $m = n$. □

The simplification step presented in Section 4.2 as well as one of the improvements to Algorithm 4.4 presented in Section 4.3.2 uses the assumption that if a set of equations do not belong to the overdetermined part in a model they can be merged. This property is concluded in the following lemma

Lemma 4.6. *Let M be a structurally overdetermined set of equations then it follows that if*

$$E = M \backslash (M \backslash \{e\})^+ \tag{4.13}$$

and

$$E' = M \backslash (M \backslash \{e'\})^+ \tag{4.14}$$

then

$$E' = E \text{ iff } e' \in E \tag{4.15}$$

Proof. Let M^{+n} be an overdetermined set of equations. Consider a equation $e \in M^{+n}$ and let $e' \notin (M^{+n} \backslash \{e\})^{+}$. Then it follows that

$$(M^{+n} \backslash \{e\})^{+} \subseteq M^{+n} \backslash \{e'\} \tag{4.16}$$

Lemma 4.3 and (4.16) implies that

$$(M^{+n} \backslash \{e\})^{+i} \subseteq (M^{+n} \backslash \{e'\})^{+j} \tag{4.17}$$

where $i = j = n - 1$ according to Lemma 4.4. Since

$$(M^{+n} \backslash \{e\})^{+(n-1)} \subseteq (M^{+n} \backslash \{e'\})^{+(n-1)} \tag{4.18}$$

Corollary 4.1 gives that

$$(M^{+n} \backslash \{e\})^{+(n-1)} = (M^{+n} \backslash \{e'\})^{+(n-1)} \tag{4.19}$$

□

In Section 4.3 it is claimed that Algorithm 4.4 finds all overdetermined sets $M' \subseteq M$ by systematically reducing the equation set and extracting overdetermined parts. In order to prove this let $alg(M)$ denote the set of all structurally overdetermined sets $M' \subseteq M$ found by Algorithm 4.4 and let $SO(M)$ denote the set of all structurally overdetermined sets that actually are contained in M. Then the following theorem can be written.

Theorem 4.1. *Let M be a structurally overdetermined set of equations then it follows that $alg(M) = SO(M)$.*

Proof. Assume that $M_0^{+m} \in SO(M)$ is an arbitrary overdetermined system where $M = M^{+n}$ so that

$$M_0^{+m} \subseteq M^{+n} \tag{4.20}$$

If $M_0^{+m} = M^{+n}$ the set is already found and there is nothing to prove, so assume that $M_0^{+m} \subset M^{+n}$. Lemma 4.5 then implies that $m < n$. Using basic set operation it also follows that

$$M_0^{+m} \subseteq M^{+n} \backslash \{e\} \tag{4.21}$$

if $e \notin M_0^{+m}$. Lemma 4.3 and (4.21) gives that

$$M_0^{+m} \subseteq (M^{+n} \backslash \{e\})^{+} \tag{4.22}$$

where $e \notin M_0^{+m}$. By using Lemma 4.4, (4.22) this can be written as

$$M_0^{+m} \subseteq M^{+n-1} \tag{4.23}$$

Since $m < n$ Algorithm 4.4 give two different cases

Case I: $m = n - 1$. From (4.23) it follows that $M_0^{+m}{=}M^{+n-1}$ and hence the set M_0^{+m} is found.

Case II: $m < n - 1$. Algorithm 4.4 sets $M^{+n}{=}M^{+n-1}$ and starts over. According to Lemma 4.4 the overdeterminedness of M will decrease with one step for each run by the algorithm and thus it will eventually end in Case I.

Since M_0^{+m} was arbitrary chosen this gives all structural overdetermined subsystem $SO(M) \subseteq alg(M)$ is found by Algorithm 4.4. By using canonical decomposition Algorithm 4.4 only returns structural overdetermined sets, hence it follows from the definition of $SO(M)$ that $alg(M) \subseteq SO(M)$. By using these results it follows that $alg(M) = SO(M)$.

□

Note that since $MSO \subseteq SO(M)$, Theorem 4.1 also implies that Algorithm 4.4 finds all MSO sets in a structurally overdetermined set of equations M.

Chapter 5

An Engine Model Example

In order to test the efficiency and, in some extent, to verify the algorithms in Chapter 4 a number of execution tests were preformed. The algorithms were tested on a variety of different engine models delivered by Scania CV AB. In this chapter some of these tests are presented. In Section 5.1 efficiency tests of different variants of the new algorithms presented in this thesis in comparison to the old algorithm in [7], will be presented. In [5] the MSO sets derived by Algorithm 4.1 has been used to construct residuals. In Section 5.2 some of these residuals will be analyzed as well as some of the MSO sets created by the algorithms in this thesis.

5.1 Algorithm Efficiency

In order to verify the efficiency of the new algorithms presented in this thesis, execution tests have been carried out. The algorithms were tested on an engine model containing 281 differential-algebraic equations and with a degree of overdeterminedness of four. The different algorithms tested are presented in Table 5.1. The analytical simplification step mentioned in Table 5.1 is presented in Section 3.1.3 whereas the structural simplification is presented in Section 4.2. In Figure 5.1 the execution times for the different versions of the MSO algorithms are shown. Note that the execution of Algorithm a) in Table 5.1 was aborted after 48 hours without any results. As shown in Figure 5.1 the simplification steps highly reduce the execution time. This shows that the basic Algorithm 4.4 is extremely dependent on the model size, completely in line with the results presented in equation (4.3).Note also that the new improved Algorithm 4.5 without the structural simplifi-

a) The old MSO algorithm presented in [7] with structural simplification.
b) The old MSO algorithm presented in [7] with structural and analytical simplification.
c) The basic MSO Algorithm 4.4 without structural and analytical simplifications.
d) The improved MSO Algorithm 4.5 without structural and analytical simplifications.
e) The basic MSO Algorithm 4.4 with structural but not analytical simplifications.
f) The basic MSO Algorithm 4.4 with structural and analytical simplifications.
g) The improved MSO Algorithm 4.5 with structural and analytical simplifications.

Table 5.1: Different versions of the MSO algorithm.

cation step (c) has the same execution time as the basic Algorithm 4.4 including the structural simplification step. This is however quite obvious since the improvement presented in Section 4.3.2 uses the same property as the structural simplification. Hence Algorithm 4.5 has the structural simplification directly included in the algorithm and there is no need for any structural presimplification. As shown in Figure 5.1 the new algorithms presented in this thesis have dramatically shortened the execution time for finding all MSO sets.

5.2 MSO Validation

As mentioned in Section 1.2 this thesis is part of a project which aims to automatically produce a diagnostic system from an engine model. The MSO sets derived with the help of the algorithms in this thesis are used to construct residuals that can be executed in real time in the engine (see [5]). The MSO sets as well as the residuals have undergone a series of tests in order validate their correctness. In this section some of those validations will be presented.

In Figure 5.2 a selection of MSO sets of equations derived from an engine model with the algorithms presented in this thesis is shown. The plot shows which equations and variables that are contained in each MSO after an attempt to structural eliminate all unknown variables, e.g. the variable n_{eng} is contained in the first MSO equation set since it is marked with a circle corresponding to n_{eng} in the figure. Note that the number of equations for the different MSO sets are to large to be fully displayed. By definition MSO sets are overdetermined

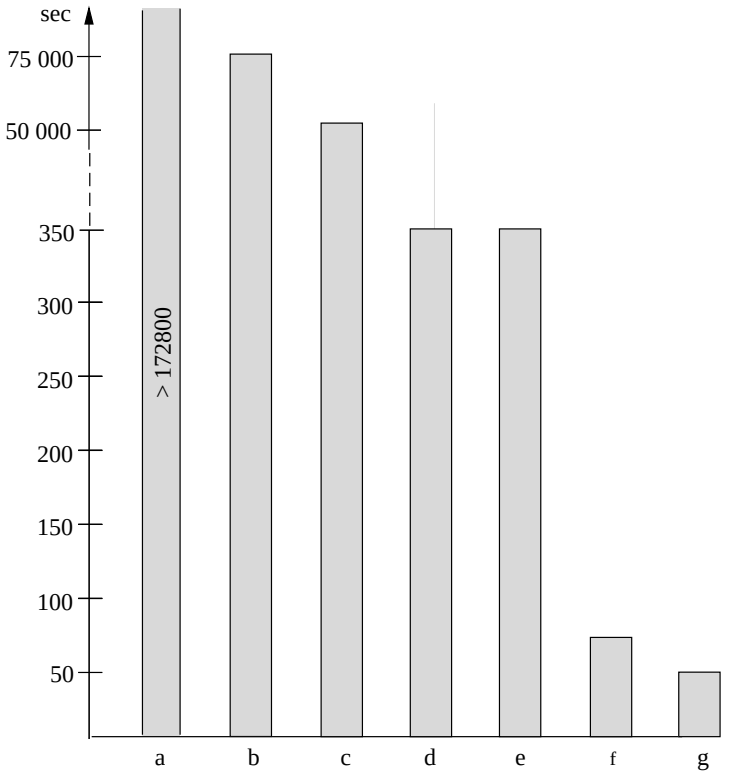

Figure 5.1: Execution time for the different versions of the MSO algorithm presented in Table 5.1.

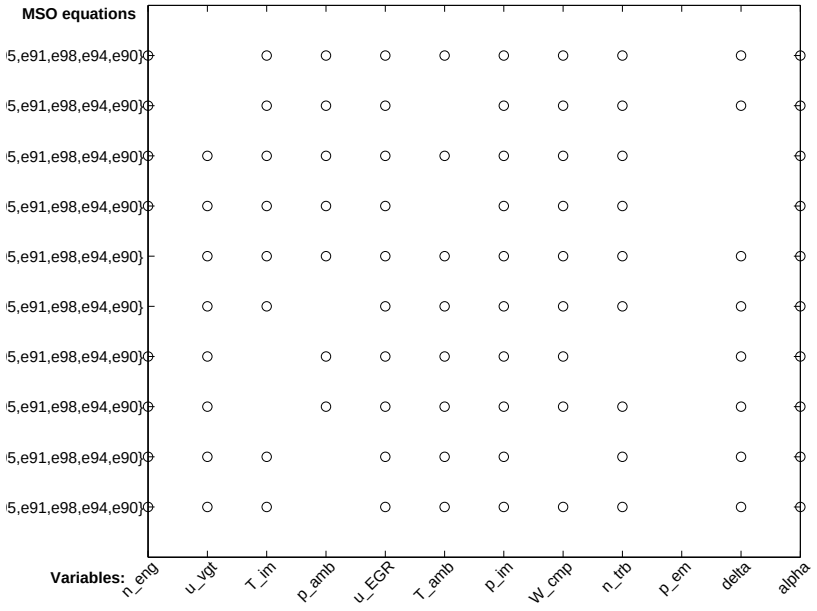

Figure 5.2: An example of MSOs extracted from a engine model.

systems with one more equation the unknown variables. Hence, since it is possible to structurally eliminate all unknown variables in an MSO, the obtained MSO sets should not contain any unknown variables. If this is the case the algorithm must be incorrect. The variables in Figure 5.2 is however known in the engine model, i.e. they are received through sensors and actuators in the system, and hence these MSO sets are valid. This has also shown to be the case for all MSO sets produced, for the engine models, by the algorithms in this thesis.

As mentioned above, residuals has been created from the MSO equation sets produced by the algorithms in this thesis. These residuals have been validated with simulation data in order to see if they behave in the supposed way, i.e. if they really can be used to indicate faults in an engine. This is however out of the scope of this thesis. However, in order to illustrate how this is done, one of these tests will be used to exemplify the technique. In Figure 5.3 the residual of one of these tests is shown. This particular residual is created from the first MSO in Figure 5.2. Hence, since the MSO contains the variable n_{eng}, the residual is supposed to be sensitive to faults concerning the signal n_{eng}. The signal n_{eng} is a notation for an actuator providing the engine with revolutions per minute (rpm) information. Figure 5.3 show

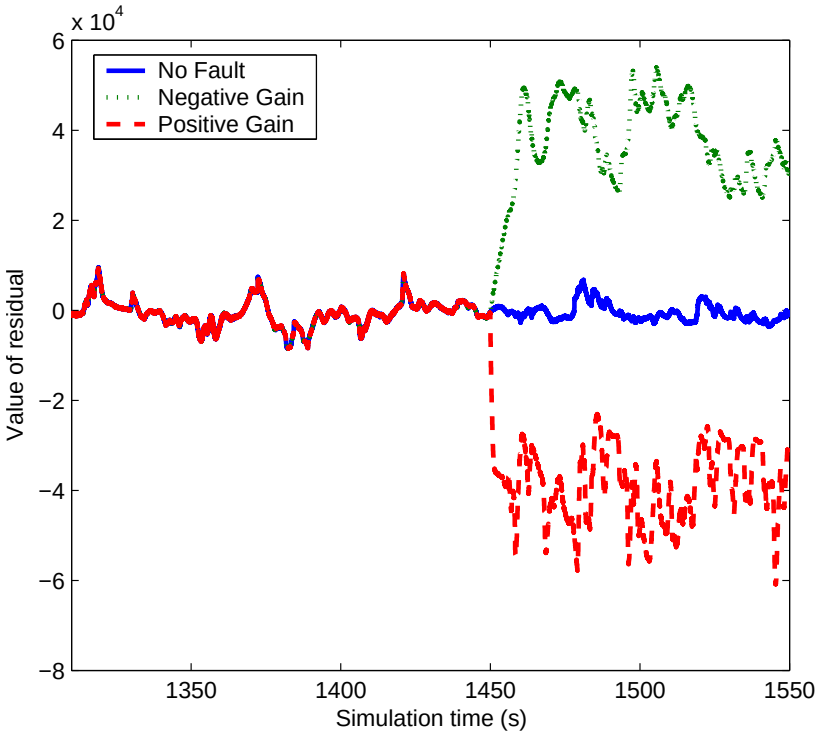

Figure 5.3: An example of an residual derived form an MSO given by Algorithm 4.5.

the residual reaction of two different types of gain faults affecting the n_{eng} actuator after 1450 seconds of simulation. As seen the residual clearly is sensitive to these faults. Hence, it is possible to construct a diagnostic test for these gain faults with the help of this residual. For more information regarding residual validations see [5].

Chapter 6

Conclusions and Future Work

In this chapter the results and conclusions are presented in Section 6.1. Then in Section 6.2 a possible extension to the work in this thesis is proposed.

6.1 Conclusions

Today's society depends on complex and technically advanced mechanical systems. To design diagnostic systems for these systems are a time demanding and complex task. Hence, it is preferable, if as much of the construction process as possible can be automated. This thesis has presented a number of algorithms and methods in order to make this possible.

Since the models considered in this thesis were implemented in Simulink, there was a need to develop an algorithm that could transform these models into analytical equations. By treating Simulink blocks as equations and lines as variables, a transformation into analytical equations was possible, see Section 3.1.2. Section 3.3 introduced a way in which fault models can be included graphically directly into a Simulink model, in order to increase the diagnostic performance of the resulting diagnostic system. The main objective in this thesis was to find the parts of a model that can be used to construct diagnostic tests. It proved to be an almost impossible task to perform this analytically, on the models considered in this thesis. Hence, the analytical models were transferred into a less complex mathematical format called structural models. Structural models allow a complex analytical model to be presented as a simple matrix, which decreases the execution times on

certain calculations significantly. By using graph theory, it has been shown that the task of finding all parts of a model that can be used in the construction of a diagnostic system can be reduced to finding all minimal structurally overdetermined (MSO) sets in these models. An algorithm for finding all MSO sets in a structural model has been developed prior to this thesis. However, in test runs this algorithm showed to be extremely computationally ineffective and even unable to produce results for the models considered in this work. In Chapter 4 new algorithms for finding these MSO sets are presented. Especially the structural simplification Algorithm 4.3 and the improved MSO finding Algorithm 4.5, proved to be important in order to perform the extraction of the MSO sets efficiently. Test runs verified the efficiency of these algorithms. It took less than a minute to extract all MSO sets from the models considered in this thesis. In Section 4.5 the correctness of the new MSO algorithms were proved mathematically. The concept of overdeterminedness introduced in Chapter 2 proved to be essential for developing theory concerning the correctness of the algorithms.

All algorithms in this thesis have during this work been implemented and tested on various truck engine models provided by Scania CV AB. The conclusion drawn from these tests is that finding all parts of a model that can be used to construct diagnostic test, can be preformed in an efficient and automated way. In Chapter 5 it was shown that these MSO sets could be used to form residuals. Hence, this thesis has constructed a foundation toward a completely automated construction process of model based diagnostic systems. In theory this means that a diagnostic system can be constructed from, for example an engine model, completely automatic and in a fraction of the time it would take to do it manually.

6.2 Future Work

The structural algorithms in this thesis only treat DLSMs (see Section 2.4.1), i.e. structural models where a variable and its derivatives are treated as the same signal. However, in some cases simpler residuals can be obtained form MSO sets of DSSMs, as the next example will show. Consider the following model

$$\begin{aligned} e_1 &: \quad y = \dot{x}_1 \\ e_2 &: \quad \dot{x}_1 = x_1 \\ e_3 &: \quad x_1 = u \end{aligned} \tag{6.1}$$

where y and u can be considered to be known. By using a DLSM this model can be transformed into the following structure

equation	unknown	known	
	x_1	u	y
e_1	x		x
e_2	x		
e_3	x	x	

where three MSO sets can be obtained as $\{e_1, e_2\}, \{e_1, e_3\}, \{e_2, e_3\}$. By using these MSO sets three residuals can be constructed as $r_1 = y - \dot{y}$, $r_2 = \dot{u} - y$ and $r_3 = u - \dot{u}$. However, by using a DSSM instead, (6.1) can be transformed into the following structure

equation	unknown		known	
	$\dot{x}_1$	x_1	u	y
e_1	x			x
e_2	x	x		
e_3		x	x	

with only one MSO $\{e_1, e_2, e_3\}$. By using this MSO it is now possible to create a residual like $r_4 = y - u$, and hence completely avoid any derivatives. For some model types, using DSSMs instead of DLSMs is therefor an advantage. This causes, on the other hand, another problem. By using the three residuals derived from the DLSM above it is possible to separate faults in the signals y and u from each other. This is not the case for the residual r_4, which is sensitive to both faults. A solution to this problem is to include derivatives of certain equations into the structural model. However, when doing this new problems arise. Firstly the structural model will become significantly larger and secondly, it may be uncertain how many derivatives that must be extracted from a specific equation.

Seemingly, there are pros and cons with both structural model types and today it is therefore impossible to say that one structure is better then the other. In order to investigate this further, algorithms for handling DSSMs could be seen as an extension to the work presented in this thesis.

References

[1] Denley T. & Häggkvist R. Asratian A. *Bipartite Graphs and their Applications*. Cambridge University Press, Cambridge, United Kingdom, 1998.

[2] Lunze J. & Staroswiecki M. Blanke M., Kinnaert M. *Diagnosis and Fault-Tolerant Control*. Linköping, Sweden, 2004.

[3] Ericson C. Mean value modelling of a poppet valve egr-system. Master's thesis LiTH-ISY-EX-3542-2004, Department of Electrical Engineering, Linköpings Universitet, Linköping, Sweden, May 2004.

[4] Nyberg M. & Frisk E. *Model Based Diagnosis of Technical Processes*. Linköping, Sweden, 2004.

[5] Krigsman K. &. Nilsson J. Code generation for efficient real-time execution of diagnostic residual generators. Master's thesis IMIT/LECS-2004-66, The Department of Microelectronics and Information Technology, Royal Institute of Technology, Stockholm, Sweden, December 2004.

[6] Cassar J. & Staroswiecki M. A structural approach for the design of failure detection and identification systems. In *IFAC Control of Industrial Systems*, Belford, France, 1997.

[7] Krysander M. Design and analysis of diagnostic systems utilizing structural methods, September 2003.

[8] Dulmage A. & Mendelsohn N.S. Coverings of bipartite graphs. *Canadian Journal of Mathematics*, (10):517–534, January 1958.

[9] Isermann R. & Ballé P. Trends in the application of model-based fault detection and diagnosis of technical processes. *Control Eng. Practice*, 5(5):709–719, March 1997.

Copyright

Svenska

English

Linköping, 10th January 2005