Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2017

Structural Diagnosis Implementation of Dymola Models using Matlab Fault Diagnosis Toolbox

Petter Lannerhed

Master of Science Thesis in Electrical Engineering

Structural Diagnosis Implementation of Dymola Models using Matlab Fault Diagnosis Toolbox

Petter Lannerhed

LiTH-ISY-EX--17/5031--SE

Supervisor: **PhD student Viktor Leek**
ISY, Linköpings universitet
PhD Ylva Nilsson
Aeronotics, Saab

Examiner: **PhD Mattias Krysander**
ISY, Linköpings universitet

Division of Vehicular Systems
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

For most diagnosis all that is needed is an ounce of knowledge, an ounce of intelligence, and a pound of thoroughness.
- John Murtagh

Abstract

Models are of great interest in many fields of engineering as they enable prediction of a systems behaviour, given an initial mode of the system. However, in the field of model-based diagnosis the models are used in a reverse manner, as they are combined with the observations of the systems behaviour in order to estimate the system mode. This thesis describes computation of diagnostic systems based on models implemented in Dymola. Dymola is a program that uses the language Modelica. The Dymola models are translated to Matlab, where an application called Fault Diagnosis Toolbox, FDT is applied. The FDT has functionality for pinpointing minimal overdetermined sets of equations, MSOs, which is developed further in this thesis. It is shown that the implemented algorithm has exponential time complexity with regards to what level the system is overdetermined, also known as the degree of redundancy. The MSOs are used to generate residuals, which are functions that are equal to zero given that the system is fault-free. Residual generation in Dymola is added to the original methods of the FDT and the results of the Dymola methods are compared to the original FDT methods, when given identical data. Based on these tests it is concluded that adding the Dymola methods to the FDT results in higher accuracy, as well as a new way to compute optimal observer gain.

The FDT methods are applied to 2 models, one model is based on a system of JAS 39 Gripen; SECS, which stands for Secondary Enviromental Control System. Also, applications are made on a simpler model; a Two Tank System. It is validated that the computational properties of the developed methods in Dymola and Matlab differs and that it therefore exists benefits of adding the Dymola implementations to the current FDT methods. Furthermore, the investigation of the potential isolability based on the current setup of sensors in SECS shows that full isolability is achievable by adding 2 mass flow sensors, and that the isolability is not limited by causality constraints. One of the found MSOs is solvable in Dymola when given data from a fault-free simulation. However, if the simulation is not fault-free, the same MSO results in a singular equation system. By utilizing MSOs that had no reaction to any modelled faults, certain non-monitored faults is isolated from the monitored ones and therefore the risk of false alarms is reduced.

Some residuals are generated as observers, and a new method for constructing observers is found during the thesis by using Lannerheds theorem in combination with Pontryagin's Minimum Priniple. This method enables evaluation of observer based residuals in Dymola without any selection of a specific operating point, as well as evaluation of observers based on high-index Differential Algebraic Equations, DAEs. The method also results in completely different behaviour of the estimation error compared to the method that is already implemented in the FDT. For example, one of the new observer-implementations achieves both an estimation error that converges faster towards zero when no faults are implemented in the monitored system, and a sharper reaction to implemented faults.

Acknowledgments

I would like to take this opportunity to express my gratitude to Erik Frisk at the Department of Vehicular Systems at Linköping university who directed me towards this position at Saab and provided me with my fundamental knowledge in the field of diagnosis and supervision. I am also very grateful to Ylva Nilsson at Saab Aeronautics for giving me a chance to put my engineering skills to the test in this thesis and for giving unyielding support throughout the thesis. I would also like to thank supervisor Viktor Leek as well as examiner Mattias Krysander for their input, which has been crucial for the development of the project. The contributions of Dag Bruck at Dassualt has been vital as well. Also many thanks to Modelon for their great support regarding their products at the start of the thesis. I would also like to thank Ingela Lind at Saab Aeronautics, who provided great contributions to the thesis through her next-to-divine skills in Dymola. Also many thanks to all my colleagues at Saab Aeronautics for this opportunity to demonstrate and improve my skills in the field of engineering. The master thesis of my colleague Karin Lockowandt was a corner-stone in the foundation of this thesis and I'm grateful for her technical support and cooperation during the project. Last but no least, I want to direct my gratitude towards my family for their encouragement and support, including all free meals and housing through my years as a bouncer and engineering-student.

Linköping, June 2017
Petter Lannerhed

Contents

Notation

Method Abbreviations

Notation	Meaning
PD	Proportional, differential (regulator)
FDT	Fault Diagnosis Toolbox

Sets

Notation	Meaning
PSO	Proper Set of Overdetermined Equations
MSO	Minimal Set of Overdetermined Equations
$\boldsymbol{X}$	Set of all unknown variables
$\boldsymbol{Z}$	Set of all known variables
$\boldsymbol{F}$	Set of all fault variables

Model Abbreviations

Shortening	Meaning
SECS	Secondary Enviromental Control System
PECS	Primary Enviromental Control System
CPC	Cabin Pressure Control
HFM	High Fidelity Model
XML	eXtensible Markup Language
ECS	Enviromental Control System
LL	Liquid Loop
LFM	Low Fidelity Model

Parsing Abbreviations

Abbreviation	Meaning
XSD	XML Schema Definition
XML	eXtensible Markup Language
XMP	XML to Matlab Parser

1

Introduction

The emotional response to faults in technology is one thing that unites humanity in the modern world, regardless if it is engineers who struggles to control a nuclear plant or beginners who are trying to install their first PC. Finding and/or repairing the faults is convenient when installing computers, and when controlling nuclear plants it is quite crucial.

When searching for faults in a system, PC or nuclear plant alike, understanding the behaviour of the specific system when it is flawless (or faultfree) is a necessity for detecting any malfunctions in the system. In order to achieve this necessity the client of this project, Saab, has a model of their systems Cabin Pressure Control (CPC) and Secondary Enviroment Control System (SECS), which are implemented in Dymola. Simulation of one of these models can give hints of what behaviour is to be expected of the flawless system under a variety of conditions.

As said before, knowledge of the flawless system is necessary to be able to tell the difference between the system being functional or non-functional. It is however, not sufficient, the knowledge of the model must be processed in order to be able to draw correct conclusions based on the available observations of the system. This process is known as model-based diagnosis as the model of the system is used to perform diagnostics on the system. By using methods from model-based diagnosis a diagnostic system can be computed. However, doing this process manually requires lots of calculations as well as the people doing the calculations being experts of the specific system. Therefore, automation of this process enables Saab to save both time and manpower as well as providing them with the benefits of fault-detection. This project, Structural Diagnosis Implementation of Dymola Models using Matlab Fault Diagnosis Toolbox, therefore has the general goal to supply Saab with an automatic method for generating diagnostic

systems from Dymola models.

1.1 Purpose

By supplying Saab with a method to automatically generate a diagnostic system, detection and isolation of faulty behaviours in their systems can be achieved. This also enables analysis of what locations sensors should be placed in order for these systems to achieve their full diagnostic potential.

Previous Saab methods for diagnosis and supervision has mostly been manual and model-specific, which often results in simple solutions that lack in generality and performance. By automatizing the process and utilizing the benefits of *structural methods*, great benefits regarding generality and performance can be achieved. Before implementing this automation in full scale however, a couple of questions must be investigated. One of the main purposes of this thesis was to investigate these questions , that are summarized in Section 1.2. The questions were answered mainly by using the Fault Diagnosis Toolbox (FDT) developed in Matlab [6] by Erik Frisk and Mattias Krysander at Linköping university, the book *Model Based Diagnosis of Technical Processes* [18] by Erik Frisk and Mattias Nyberg and earlier non-linear observer implementations [12]. More details on the sources used in the thesis is found in Section 2.6.

1.2 Research Questions

Based on the SECS, CPC and other simpler models, properties of the FDT was investigated. The questions that were answered in this thesis are summarized below.

1. Is it possible to parse CPC from Modelica to FDT-compatible form in Matlab?

2. What time complexity is achieved when computing a diagnostic system for SECS in the FDT?

3. What isolability is currently achievable in the SECS model according to the FDT given no demands on the residuals causality?

4. How does causality restrictions on the residuals effect the generated diagnostic system?

5. What benefits and drawbacks follow from adding residual-implementations in Dymola to the current FDT?

6. Will additional sensors improve the isolability found in issue 3?

If so, what placement of sensors will result in the greatest gain for a given limit of the number of sensors?

7. Can false alarms be prevented in the diagnostic system generated for SECS? If so, in what way?

1.3 Delimitation

The time budget of the project is limited and therefore the subject treated in this thesis also has certain limits. The project will not include any additional modelling or modification of the given models. This is avoided in order to remain in compliance with demands from the client of the project, Saab. Certain data, such as details on limits of the given system and measurements that reveal the behaviour of the system in that state, may not be accessible because of confidentiality. The limits of the system and treatment of data surrounding that state will therefore not be included in the project.

Furthermore, the residual evaluation will be based on simulations of the corresponding model, no real measurements are taken into account in this thesis. The specifics of all Dymola-implementations are seen in Table 1.1 and all FDT implementations are based on the version of the FDT from 2016-12-22 and Matlab R2016B.

Table 1.1: *Dymola specifics for the implementations of the thesis*

Version	2018 alpha
Solver name	Dassl
Solver tolerance	0.0001
Number of Intervals	500
Start time	0
Stop time	250
Step length	Adaptive

Given the high complexity of the CPC model it will not be translated completely to Matlab. However, a structural model, resembling what variables that are present in what equations, will be computed. This delimitation is motivated by the complexity, which originates mostly from the thermodynamic components of the Modelica library 'medium' and non-linear functions that lack counterpart in Matlab. The SECS model however will be treated as a complete model, although a the low fidelity version of the model (LFM) is treated.

When investigating the time complexity of the diagnostic system generation, only the computation time of MSOs (see Section 2.2.2) in relation to redundancy (see Section 2.1) is taken into account, as time-consumption for model-specific design choices are harder to perform experiments on in a consistent way.

2

Background Theory

When reading this thesis, certain knowledge in the fields of model-based diagnosis, signal processing, and optimal control is necessary for fully understanding the implementations. This knowledge is provided in this chapter. For the advanced reader, the summary in Section 2.6 might suffice, while the other sections provides a more detailed description of the field stated in the title of the corresponding section.

2.1 Diagnosis and Supervision

When technology is functional, all the performance of a diagnostic system might seem like a waste of resources. This loss however is compensated by many benefits when the system is malfunctioning. These include information about whether something in the monitored system is malfunctioning, 'detectability', and in certain cases also the ability to pinpoint the specific fault, 'isolability'. These properties provides how much potential the diagnosis system has when it comes to generating a diagnosis, that is, whether something is wrong or not, and what faults that might cause the incorrect behaviour. In other words, the diagnosis describes what mode the system is currently in. These terms are clarified by the following example of how a diagnostic system can be implemented.

Example 2.1

A system consists of a tank with pressure p, which is monitored by two pressure sensors, p1 and p2. The measurement signal of these sensors are denoted y_{p1} and y_{p2}, which results in the equations

$$y_{p1} = p \tag{2.1}$$

$$y_{p2} = p \tag{2.2}$$

Equation 2.1 and equation 2.2 indicates hardware redundancy, as there are multiple sensors that measure the pressure p. The hardware redundancy can be used to construct a diagnostic system. This is practically achieved by measuring the difference between the measurement signals of the two sensors, resulting in the following diagnostic system;

$$y_{p1} - y_{p2} \neq 0 \rightarrow \textit{The system is in a malfunctioning mode} \tag{2.3}$$

The diagnostic system described by (2.3) clearly has detectability for faults in the measurements provided by the sensors. But, given that it cannot be determined which of the two sensors that is malfunctioning, the system lacks in isolability.

The lack of isolability in Example 2.1 results from the lack of redundancy, which means to what extent the system is overdetermined. This small example has two equations, that follows from the measurements of each of the sensors, and one unknown quantity; the pressure. This results in the level (or degree) of redundancy being $2 - 1 = 1$. Therefore, in the current situation, there is only one way to construct a residual from the given equations. The residual is a relation between known signals that is equal to zero when no faults are present in the system, see Section 2.3. The residual of Example 2.1 is therefore described by (2.3), as both sensors must provide the same correct measurement if the system is flawless. It is worth mentioning that the number of possible residuals is not equal to the level of redundancy in the general case, see the example of Section 2.1.1 for instance.

2.1.1 Model-based diagnosis

While achieving higher degree of redundancy by hardware redundancy is an easy method to achieve highly reliable fault detection, the additional sensors also results in increased costs and complexity [18]. By utilizing a model of the system instead, additional equations can be used to increase the constraints on the monitored system, and thereby also increase the level of redundancy according to the reasoning in the end of Section 2.1. The usage of models for diagnosis in this manner is known as model-based diagnosis. Similar to the previous subsection, this is concretized by an example.

Example 2.2

Assume the same premises as in the example of Section 2.1, with the addition of

$$G(w) = p \tag{2.4}$$

where G is a known function that follows from elementary chemistry, and the weight w is measured by a scale w1 with measurement signal y_{w1}. This measurement can be written as

$$y_{w1} = w + f_{w1} \tag{2.5}$$

where f_{w1} denotes the measurement error of $w1$.

The addition of (2.4) and (2.5) results in the degree of redundancy being $4-2=2$. Another consequence of this addition is that there are at least 2 additional ways to achieve overdetermined sets of equations by combining equation (2.8) with each sensor. Lets assume that only measurement faults are considered, that is, the set of possible faults being $F = \{ f_{p1}, f_{p2}, f_{w1} \}$ and corresponding system behaviour modes then being F_{p1}, F_{p2}, F_{w1}. The resulting equation system with these faults implemented is;

$$y_{p1} = p + f_{p1} \tag{2.6}$$
$$y_{p2} = p + f_{p2} \tag{2.7}$$
$$y_{w1} = w + f_{w1} \tag{2.8}$$
$$G(w) = p \tag{2.9}$$

The equations in (2.6) to (2.9) results in the following diagnostic system

$$y_{p1} - y_{p2} = r_1 \neq 0 \rightarrow D_1 = F_{p1} \vee F_{p2} \tag{2.10}$$
$$y_{p1} - G(y_{w1}) = r_2 \neq 0 \rightarrow D_2 = F_{p1} \vee F_{w1} \tag{2.11}$$
$$y_{p2} - G(y_{w1}) = r_3 \neq 0 \rightarrow D_3 = F_{p2} \vee F_{w1} \tag{2.12}$$

where D_i corresponds to diagnosis i.

The diagnostic system of Example 2.2 clearly has advantages compared to the system of Example 2.1. For example; each residual reacts to a different subset of F. The system of Example 2.2 also resulted in that every fault in F is not causing any reaction in one of the residuals r_1 to r_3. The achievement of this non-reaction is called decoupling of the corresponding fault. For instance it follows from equation 2.10 that f_{w1} is decoupled in r_1. Decoupling is necessary in order to achieve isolability according to [18].

A compact way of describing what faults that are decoupled in each residual is by defining a structure known as the influence structure, which is basically a matrix where each row corresponds to the sensitivity of one test to each of the faults, specified by the element of the corresponding column. The definitions of [18, p. 25] is followed here for simplicity, which means that 0 corresponds to the fault being decoupled, X corresponds to the test reacting to the fault but not with any guarantee, while 1 corresponds to a guaranteed reaction. For this specific example, the influence structure is summarized in Table 2.1.
The reason that the influence structure in Table 2.1 consists of elements with value X instead of 1 is that there is a risk that faults cancel each other out in the residuals and therefore it cannot be guaranteed that these residuals react to the undecoupled faults under every circumstance. Another interpretation is that no conclusion can be drawn from a residual that has no reaction in this case, while an influence structure with ones would exclude any detectable faults from the diagnosis in the event of no reaction. For general systems an additional reason for having X:es in the influence table is that some faults may only be detectable

Table 2.1: *Influence structure of resulting diagnostic system from Example 2.2*

	f_{w1}	f_{p1}	f_{p2}
r_1	0	X	X
r_2	X	X	0
r_3	X	0	X

when the system is in a special operation point. It is clearly beneficial to have 1s instead of X:es from the viewpoint of isolablility.

Formally, the isolability between the system modes b_1 and b_2, with the corresponding observations O_{b1} and O_{b2}, can be defined as follows;

Definition 1. [18] A system behavioral mode b_1 is isolable from the system behavioral mode b_2, if Observations of the system being in mode b_1 is not a subset of the Observations of the system being in b_2, or $O_{b1} \notin O_{b2}$.

The definition of isolability from Definition 1 will now be clarified by using the influence structure of Table 2.1. By assuming that the faults f_{w1}, f_{p1}, f_{p2} does not cancel each other out, it can be concluded that observations of r_1, r_2, r_3 are sufficient in order to achieve isolability for single faults according to Definition 1. This follows from the fact that there is one observable residual that decouples each fault, while reacting to the other monitored faults. From the fact that all 3 monitored faults are present in the influence structure it follows that the faults are isolable from the mode of the system that is faultfree, also known as the nofault-mode, NF. This provides a more solid definition of the detectability described in the introduction of Section 2.1;

Definition 2. A fault f is detectable if the corresponding behavorial mode is isolable from NF.

It is worth to consider that these definitions does not take any reactions to non-monitored faults or disturbances into account. Also note that definition 1 does not consider any degree of difference between O_{b1} and O_{b2}, just that O_{b1} is not a subset of O_{b2}. For example, there are residuals that are sensitive to a certain fault, but still results in the steady state responses of the residuals being zero. This situation corresponds to the fault being weakly detectable in that residual.

2.2 Structural Methods

The examples of diagnostic system computation that has been shown so far has been solved by hand. However, in general, a more sophistical method is needed in order to find the equations that are of interest for diagnostic purposes. One way to find these set of equations is by using structural methods.

Similar to many other search-algorithms, one key ability is the ability to label certain information as non-essential, also known as the ability of pruning the search-tree. Search-algorithms typically achieves this by using so-called heuristics which approximates the possible future contribution of any node in the searched tree. When designing a system used for diagnosis, similar pruning of the available information can be achieved by just using the model structure alone. The model structure contains the information of what variables that are present in each equation, but no information regarding in what context the variable is present [13]. For clarity, the model structure from Example 2.2 is shown in Table 2.2.

Table 2.2: *The model structure of the monitored system in Example 2.2*

	y_{p1}	y_{p2}	y_{w1}	w	p	f_{w1}	f_{p1}	f_{p2}
(2.6)	X				X		X	
(2.7)		X			X			X
(2.8)			X	X		X		
(2.9)				X	X			

Structural methods describes methods that uses the model structure when designing the system used for diagnosis. According to [13] only the subset of the models equations that is overdetermined can be used for diagnosis implementations. This follows from the fact that an overdetermined system got more equations than unknowns, which results in redundancy under the assumption that each unknown variable that is present in an equation also is explicitly expressible using this equation. Under this assumption, each variable can be linked to an equation, which represents the variable being expressed by that same equation. The process of combining equations and unknowns in this manner can be loosely interpreted as matching [8].

2.2.1 Matching

As matching is a key concept in the computation of diagnostic systems, the definition provided so far is too vague. A clarification is needed. In order to achieve this however, certain theory must be considered. This theory is summarized in this section.

From earlier it is clear that the analysis done by structural methods, or structural analysis, does not rely on what type of analytical expression that combines the variables to a specific equation. Instead, structural analysis is performed based on the set of unknowns $\boldsymbol{X}$, the set of known signals $\boldsymbol{Z}$ and the set of faults $\boldsymbol{F}$, or more specifically, the subset of each of those sets that are present in each of the models equations. Each equation in the system contains a subset of elements from each of these sets. The expressibility of these element in each equation is described by the Bi-Adjacency Matrix, that is, what variables that exist in each of

the equations, and whether integration or differentiation is required during the extraction of these variables. For example, see Figure 2.1 from [8], in which the equations are represented as rows and elements of $\boldsymbol{X}$ is represented as columns. This results in the element in row i and column j represents the expressibility of 'Unknown j' in equation i. In Figure 2.1 the value X represents the unknown being expressible, while I and D represent expressibility through integration and differentiation. Given that equations act as constraints on the unknowns, equation i is denoted as c_i in this context, see Figure 2.1. Using this notation, it is clear that c_{10} to c_{12} represents equations of the type $\dot{x} = \frac{dx}{dt}$.

Figure 2.1 clarifies the topic of matching as a set of valid connections between the rows (representing the set of equations) and columns (representing X), where a connection is valid if and only if the variable is present in the equation that it is connected to and the variable/equation is not used in another connection. An example of a valid connection is therefore c_1, q_1 according to Figure 2.1.

	q_0	q_1	q_2	q_3	p_1	p_2	p_3	$\dot{p}_1$	$\dot{p}_2$	$\dot{p}_3$
c_1		X			X	X				
c_2			X			X	X			
c_3				X			X			
c_4	X	X						X		
c_5		X	X						X	
c_6			X	X						X
c_7					X					
c_8			X							
c_9	X									
c_{10}					I			D		
c_{11}						I			D	
c_{12}							I			D

Figure 2.1: *Example of a Bi-Adjacency Matrix. Source [8]*

By combining this information of each equation with the fact whether the equation is included in the underdetermined, exactly determined, or overdetermined part of the system, the Dulmage-Mendelsohn Decomposition is achieved [8].

2.2.2 Minimal sets of Overdetermined equations

The sorting that is achieved by the Dulmage-Mendelsohn Decomposition results in knowledge of what subsets of variables $\boldsymbol{X}$', faults $\boldsymbol{F}$' and equations $\boldsymbol{EQ}$' that are present in the overdetermined part of the system. Given that this describes the maximum subset of the model that can be used for diagnosis, the maximum

detectability is obviously given by $\boldsymbol{F}'$. In order to achieve isolability, decoupling is necessary, see Section 2.1.1. Given the following assumption from [8]

Each fault is present in only one equation (2.13)

, all sets that are containing a minimal amount of equations while remaining overdetermined are of great interest, as they achieve the full decoupling potential. A set that fulfils this property is known as a Minimal set of Overdetermined equations (MSO). An example of an MSO is the set of equation (2.6) and equation (2.7).

2.3 Residuals

The role of residuals has been discussed throughout this thesis. Before the process of residual generation that was used in this thesis is described however, a thorough background on the residual-subject must be given. This includes the theoretical definition as well as properties of residuals that are worth considering when constructing a diagnostic system. One certain preconditions that must be met when constructing residuals is that the included equations must compose an overdetermined system, which follows from the definition of redundancy given next.

Definition 3. [18] There exist analytical redundancy if there exist two or more different ways to determine a variable x by only using observations z, that is $x = f_1(z)$ and $x = f_2(z)$, where $f_1(z) \not\equiv f_2(z)$.

By using the definition of analytical redundancy from Definition 3, the following general expression of a residual can be formed;

$$r = f_1(z) - f_2(z) \tag{2.14}$$

From equation (2.14) it follows that $r \neq 0$ provides an observation that is different from the observation when the system is in a faultfree mode. This enables isolation from the mode according to Definition 1. This specific isolability is identical to detectability.

The analytical redundancy can occur both as static redundancy and temporal redundancy. Static redundancy corresponds to f_1 and f_2 not containing any differential operators, that is, no numerical integration or differentiation of the elements in z are necessary. Temporal redundancy means the opposite; that at least one numerical integration or derivation is necessary of the known quantities in order to achieve the residual. Whether a redundancy is temporal or static is of great importance as it effects the properties of the computed residual. One of these properties is causality which is covered in Section 2.3.1.

2.3.1 Sequential residuals

As previously stated in this thesis, only overdetermined set of equations can be used for generation of residuals. This follows from the definition of redundancy,

Definition 3. In an underdetermined or exactly determined system, there is only one way (at the most) to describe each unknown quantity. This means that there is no redundancy, and therefore no residual can be generated. Sets that does not contain any underdetermined or exactly determined parts are therefore of great interest, these sets are known as Proper Sets of Overdetermined Equations (PSOs).

If the system S is overdetermined however, then there are multiple subsystems S' that are exactly determined and contains all unknowns of S. This means that every unknown variable in S' can be expressed using S' alone. By substituting the unknown variables in $S \backslash S'$ with these expressions, the redundancy of Definition 3 is achieved.

If the redundancy is temporal, then numerical integration, differentiation or a combination of these must be applied to the known quantities of S in order to achieve a residual. This corresponds to the residual having integral , derivative or mixed causality. Selecting S' therefore effects the causality of the resulting residual, given that the redundancy is temporal. If the redundancy is static however, no differentiation or integration of the known quantities is necessary. Given that only algebraic substitutions are sufficient for residual generation in this case, the causality is called algebraic. For more information regarding the definition of causality, see [8].

2.3.2 Observer based residuals

While the implementation of sequential residuals is straight-forward, it is also sensitive to measurement noise. One way to suppress this noise is by following the methodology described in Section 2.4 assuming that an overdetermined system with non-algebraic causality is available. This system can be written in the form of equation (2.17), from which the residual can be computed as the estimation error.

2.4 Observers

When estimating a certain quantity in a system that is modelled, there are two sources of information in the general case. The first source is the model itself. By initiating the model with an estimation of the current state of the system, certain prediction of the systems behaviour can be achieved by simulating the model. Relying on this source alone however, results in high demands, both on the accuracy of the model and the ability to estimate the initial state of the system. Given that the initial state can be regarded as an unknown quantity, this results in a situation where the estimation method only can provide accurate estimates if certain accurate estimates are already given. This situation is resolved by using the second source of information, which consists of the measurements provided by the sensors of the modelled system. The combination of these sources results

in an estimator with maintained predictive ability and less dependency of the initial state estimate. This combination is achieved by an observer. Consider the following linear state space model

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{2.15}$$

where x denotes the state, u denotes the input and y denotes the measurements. The coefficients A, B and C are matrices that describes how the previous denoted quantities are connected. Let $\hat{x}$ denote the estimated state. By utilizing the model from equation (2.15) and using proportional feedback from the difference between measured y and estimated measurements $C\hat{x}$, also known as the estimation error, correction of the estimated state can be achieved. The gain of the proportional feedback; K, is known as the observer gain. Based upon this information the observer can be defined as the following;

$$\dot{\hat{x}} = A\hat{x} + Bu + K(y - C\hat{x}) \tag{2.16}$$

The resulting observer of equation (2.16) is clearly depending on the model of the system, the provided measurements and the observer gain K. The details of determining K is not treated in this thesis, but is referred to other implementations, such as the continuous variance-minimizing Kalman filter [2].

2.4.1 Observers of Non-Linear Systems

Unfortunately, far from all systems are linear and on state-space form such as the system in (2.15). Now consider the following non-linear differential algebraic model

$$\begin{aligned} \dot{x}_1 &= f(x_1, x_2, z, t) \\ 0 &= h(x_1, x_2, z, t) \end{aligned} \tag{2.17}$$

where the unknown variables x has been split up into the state variables x_1 and the algebraic variables x_2. The known signals, such as measurements and inputs are described by z.

In order to achieve an observer with desired behaviour the nonlinearities must be handled in some fashion. The easiest method would include simplification of the model by expressing x_2 as a function of x_1, t and z followed by a linearization of the model. Then the Kalman filter can be applied to these linearized models, which results in the Extended Kalman filter described in [10, Chapter 7].

Before discussing the observer implementations used in this thesis, a background on Differential Algebraic Equations (DAEs) and more specifically their index is needed. An example of a DAE is seen in equation (2.17), however, as stated in [14] the equations must not necessarily have the derivatives expressed explicitly as in equation (2.17). The index of the DAE corresponds to the number of differentiations needed in order to achieve that expressibility according to [14]. Based

on these definitions of DAEs and index, the observer implementation can be specified.

This thesis uses the observer implementation described in [12], see equation (2.18). This is motivated by the fact that the current implementation of observers in the FDT only supports low-index problems (index=1), which is complemented by the index-reduction capabilities of the method described in equation (2.18).

Background on Method for Non-Linear Observers

Similar to the regular observer in equation (2.16), the method presented in this section uses feedback from the estimation error r in order to minimize the difference between the estimated and measured state. The difference from many simpler applications, such as those described in Section 2.4 is that this method uses a feedback based on the estimation error r with both proportional and derivative terms. Based on the model seen in (2.17), this results in the following system

$$\begin{aligned} \dot{\hat{x}}_1 &= f(\hat{x}_1, \hat{x}_2, z, t) + F(t)\dot{r} + G(t)r \\ 0 &= h(\hat{x}_1, \hat{x}_2, z, t, r) \end{aligned} \tag{2.18}$$

where the coefficients $G(t)$ and $F(t)$ provide proportional and differential feedback in the constructed observer, and the specific presence of r in h is left as choice of design. In this thesis, the method presented in [12] is used to determine the coefficient $F(t)$. As seen in equation (2.18) this method utilizes the information from the estimation error by using feedback provided from a PD-controller. By selecting the coefficients of the controller according to Theorem 1 and Theorem 2 in [12], low-index and asymptotic stability can be guaranteed. In this thesis however, only theorem 1 will be used. Before the theorem is given, let h_x denote the gradient of h with respect to x and $N(A)$ denote the right null space, that is $\{x : Ax = \vec{0}\}$.

Theorem 1 Given a model in the form of equation (2.17) a low-index observer on the form of equation (2.18) can be guaranteed if the following criterions are met.

1. h_x have full row rank for all x.
2. h_{x_2} must have full column rang for all x_2.

The first criterion corresponds to no rows in the gradient of h being linearly dependent. The second criterion corresponds to the requirement that every component of h, must have derivatives based on elements from x_2 such that these derivatives results in columns that are linearly independent of each other. Or, in less mathematical terms, that it is possible to solve for x_2 in the algebraic part of equation (2.17).

The criterions mentioned so far are prerequisites and doesn't reveal any information regarding the specific design of $F(t)$. This is described by the third criterion,

stated below. Let N denote the null space. Also, let V and W be defined in the following manner

$$V = \{ x_1 : \exists x_2 : (x_1, x_2) \in N(h_x) \} \tag{2.19}$$

$$W = \{ x_1 : x_1 \notin V \} \tag{2.20}$$

If the previous requisites are fulfilled and $\mathrm{Im}\ F(t) = W$ (the Image of $F(t) = \mathrm{W}$), then the resulting observer is low-index which concludes Theorem 1 from [12]. The proof of this theorem is found in [12].

2.5 The Optimal Control Problem

Similar to the chess player planning their next move, the field of optimal control seeks the optimal sequence of actions. Expressing these actions or controls as functions of the current state or current position of the pieces on a chessboard if you wish, is known as expressing a policy. Finding an expression for the optimal policy is the main goal in the field of optimal control. It must however be specified in what way a certain policy is optimal. This information is found in the formulation of the optimal control problem

$$\min_{u(.)} . \ \phi(x(t_f)) + \int_{t_i}^{t_f} f_0(t, x(t), u(t))dt \quad subject\ to \begin{cases} \dot{x} = f(x(t), u(t)) \\ u(t) \in U^* \\ x(t) \in X^* \end{cases} \tag{2.21}$$

which in the content of this thesis got a predetermined initial time t_i and end time t_f. While the problem described by equation (2.21) looks very complex at first sight it is basically a consideration between the penalty of effort, measured by the integral-part of the objective function and the penalty of final performance, which is measured by ϕ. For the chess-player for example, ending in the state of having the opponent in checkmate, is a very successful state to achieve. Therefore it should correspond to a very low value of ϕ. With similar reasoning, ending up being checkmated by the opponent should correspond to a high value of ϕ. Similar to chess, the state and control signal cannot change in an arbitrary fashion, there are a set of constraints or rules regulating what states and control signals that are valid. These constraints are mathematically described by X^* and U^* and corresponds to the chess player not being allowed to have pieces outside of the board or move diagonally with the knight for example.

When solving the problem described in equation (2.21) the field of optimal control provide two general analytical methods that are applicable to the given problem. The Hamilton-Jacobi-Bellman equation (HJBE), which provides sufficient and necessary conditions for optimality and the Pontryagin's Minimum Principle (PMP), which only provides necessary conditions for optimality [25]. In this thesis only PMP has been used, and therefore the focus will remain on this method from this point on.

2.5.1 Pontryagin's Minimum Principle

The mathematical process of optimization is often simple, given the context of the upper secondary school. A typical approach is differentiation of the non-linear objective function and comparing all values of the objective function that arises from the zeros of this derivative. However, when optimizing with constraints, as in equation (2.21), the complexity of the problem suddenly rises above the level of upper secondary school.

When the constraints consists of an state space model, Pontryagin's Minimum Principle (PMP) enables calculation of candidates for an optimum in a similar fashion to the derivative-solution above. The clearest similarity is that PMP also provides necessary (but not sufficient) requirements for optimums. It is therefore only used to compute candidates which must be compared manually in order to verify the optimum.

Hamiltonian and Equations

The computation of the optimum of equation (2.21) must somehow take both the goal-function and the constraints into account. PMP solves this matter by combining the $f_0(t, x(t), u(t))$ with the differential constraint of the state $f(x(t), u(t))$ with the help of the adjoint variable known as λ. This combination is described by the Hamiltonian;

$$H(x(t), u(t), \lambda, t) = f_0(t, x(t), u(t)) + \lambda \cdot f(x(t), u(t)) \tag{2.22}$$

Given that λ has been introduced, additional equations must be added in order to achieve a solvable equation system. The resulting equation system is

$$Constant = \min_{u(.)} . \ H(x^*(t), u^*(t), \lambda, t) \tag{2.23}$$

$$\dot{\lambda} = -H_x \tag{2.24}$$

$$\lambda(t_i) \perp x^*(t_i) \tag{2.25}$$

$$\lambda(t_f) - \nabla\phi(x^*(t_f)) \perp x^*(t_f) \tag{2.26}$$

For more information see [9]. In this thesis, only the pointwise minimization from equation (2.23) and the adjoint equation from equation (2.24) will be used. Given that equation (2.25) and equation (2.26) only provides information regarding the specific initial and termination values of x and λ the loss from excluding these equations are considered minimal.

2.6 Summarized Roadmap of Background Research

The investigation of the issues in Section 1.2 is dependent on access to relevant and suitable scientific sources. This section gives a general summary of the theory used in the project and also points to adequate sources on the respective field.

As stated in Chapter 1 model based diagnosis refers to modelling a system and to build the diagnostic system based on that model [18]. This typically results in a number of equations that capture the main behaviour of the modelled system [14]. When applying structural methods however, as described in Chapter 1 and [13], only the model structure is relevant.

Of the equations describing the system, only certain equations can be used for diagnosis, the equations that form overdetermined equation systems. One way to compute this equation set is to use the Dulmage-Mendelsohn composition, which is described in [8]. However [8] also states that there can be smaller overdetermined subsets of this set known as minimal sets of overdetermined equations, MSOs. As these sets are minimal, it follows that they contain exactly 1 more equation than unknown variables. The paper [11] shows examples of algorithms which find these sets and compares them with respect to complexity.

While knowing the information provided by the MSOs is sufficient for generating all possible residuals of a system, additional information is needed in order to actually generate residuals. The paper [4] shows an implementation on this topic. The application in [4] resulted in residuals with mixed causality. The causality of residuals is described according to [8] by what type of calculations; differentiation, integration, that are necessary for achieving the residual. As stated earlier all MSOs consists of 1 more equation than the number of unknowns, removal of one equation, e_i from the set will result in an exactly determined equation system. This means that all of the unknown variables in the MSO are computable and thus computing them and substituting the unknown variables in e_i results in a residual-equation. However this substitution may result in demands for numerical integration, derivation or a combination of these, which results in integral, derivative or mixed causality. In the event that none of these options are required for the substitution, the causality is called algebraic.

The computation may include solving of Diffential Algebraic equations (DAEs), which is described in [14] as general equations involving variables and their derivatives. [14] also describes the index of a DAE as the number of times the DAE must be differentiated in order to achieve state space form. High-index DAEs are typically harder to solve according to [6, 22], although [22] described an algorithm for computing the largest low-index subpart of a high-index DAE. There are also examples of algorithms that converts high-index DAEs to low index directly, see [21]. If the problem is low-index then according to [6] an observer based residual can be constructed in the current observer-implementation. Construction of observer based residuals based on a non-linear differential equation is described in [12].

While causality and index describes the mathematics required to compute a diagnostic system, it doesn't describe the properties of the achieved diagnostic system itself. One such property is isolability, which according to [18], describes the ability to exclude other faults when in presence of a specific fault. More details on

isolability can be found in [16, 20], while [19] contained an algorithm for evaluating the property.

The resulting properties, such as the isolability of the diagnostic system, are dependent on the tests chosen to be included in the diagnostic system. There are certain limitations on test selection in order to satisfy demands for isolability, this topic is treated in [3, 17]. The paper [3] covered an efficient way of reducing the search-space by taking the realizability properties into account. According to [3] the method was also applied successfully to an industrial sized automotive engine system. The paper [17] on the other hand describes an algorithm for deriving MSOs. The article states that the algorithm has polynomial complexity and therefore is more suitable for industrial examples than other algorithms.

A necessary condition for having any isolability or detectability, i.e isolability from not having any fault; is redundancy. According to [18] redundancy means having more equations than unknowns, which basically is resulting in the overdetermined equation system discussed earlier. One way to affect the number of equations is placement of certain sensors, this topic was covered in [7]. In [7] it was also stated that adding new sensors enables transition of equations from the exactly determined part of the Dulmage-Mendelsohn decomposition to the overdetermined part. Furthermore, based on the model structure, minimal sets of sensor placements for full detectability can be computed and similar algorithm is also applicable for isolability according to [7].

When adding sensors however, it's important to consider the Signal-to-Noise-Ratio (SNR) itself as well as its size relative to the precision of the equations of the model. The topic of extracting as much information as possible from a set of sensors when in presence of noise is called sensor fusion. In sensor fusion one of the fundamental algorithms, `Weighted Least Squares`, provides a way to value the received information with respect to its certainty. This algorithm is described in [10].

3

Background

The background of the thesis is presented in this section. As stated in Chapter 1, the key concept of the diagnostic system computation is translation/parsing of Dymola Models into Matlab. In order to achieve this, an understanding of the Dymola language Modelica is needed, this is provided in Section 3.1. The translation process involves the XML-format, which is briefly covered in Section 3.2, and a translation program (parser), which is covered in Section 3.3. The parser was developed during the thesis but is founded on the work in [15]. The parser is applied to the CPC model, which is introduced in Section 3.4.1, the two tank model in Section 3.5.1 and the SECS model presented in Section 3.6. Given a correct translation of these models, the FDT can be applied in order to compute a diagnostic system, the FDT is covered in Section 3.7.

3.1 Modelica

The language used in Dymola is called Modelica. Modelica is a equation-oriented programming language, where basically a list of equation describes the behaviour of the model based on given parameters and variables, see Figure 3.1. Similar to many other languages, the left part of the declaration declares the type while the rest specifies the name. Also note that all variables and parameters must be declared before they are used [14]. The equations using these parameters and variables are specified under the line 'equation'. When adding variables, equations or parameters, the line must be terminated by a semicolon. Suggested initial values of variables can also be added in the declaration by adding $(start = value)$ to the declaration, where $value$ is the suggested value. Any declarations of parameters/variables and definitions of the equations must all be within the limits of the model, which is marked by 'model *model-name*' and 'end *model-name*'. For clarification see the example in Figure 3.1.

```
model Rocket
parameter Real losses=0.1;

Real velocity(start=0);

Modelica.Blocks.Interfaces.RealInput thrust;

equation
der(velocity) = thrust*(1-losses);

end Rocket;
```

Figure 3.1: *Modelica code example that explains variables, parameters and equations*

3.1.1 Re-usage of Code

According to [14] Modelica also includes model-libraries which enables the usage of model-objects as a sub-model of new models. By defining and saving the definition of *modelx* as in Section 3.1, the addition of the line *modelx modelx_1;* will add an *modelx* object with the name *modelx_1*. See the example in Figure 3.2 where *Rocket* objects called *Rocket_1* and *Rocket_2* are added. The partial prefix of the model indicates that the model can be expanded using inheritance, see Figure 3.3. Defining model-objects in this manner, having model-objects inside

```
partial model Rocket_fleet
Rocket Rocket_1;
Rocket Rocket_2;
end Rocket_fleet;
```

Figure 3.2: *Modelica code example that explains re-usage of models*

other models, results in a *hierarchical model*. The hierarchy in the hierarchal model is refering to the fact that the variable x from inside the modelx object is different the variable x being on the same level as the modelx object. These levels are here refereed to as the hierarchy and are also exemplified in the given example, as the Rocket_fleet model contains Rocket objects.

Connectors

One of the greater benefits of Modelica is the possibility to reuse older models in different applications. This is achieved by defining objects of different models and connecting them. In order to transfer data between model objects, an element called connector must be used. Connectors enables connection of different models by specifying an interface, which then can be used in connect-statements, see the example in Figure 3.3.

Inheritance

Modelica also includes the possibility of inheritance. By defining a model as a partial model and using the command extend, all the source-code from inside the limits of the model in Section 3.1 is pasted at the location of *extend model-name;*. Both connectors and inheritance is treated in the example of Figure 3.3.

```
model complete_fleet
extend Rocket_fleet;

Modelica.Blocks.Interfaces.RealOutput order_1=1;
Modelica.Blocks.Interfaces.RealOutput order_2=-1;

equation
connect(Rocket_1.thrust,order_1);
connect(Rocket_2.thrust,order_2);

end complete_fleet;
```

Figure 3.3: *Modelica code example that explains re-usage of models using inheritance and connectors*

3.1.2 Algorithms

As mentioned earlier, models are described by equations, where the left hand side of the equality sign is equal to the right hand side of the equality sign. The order of operands doesn't matter given that equality is a symmetric relation, $a = b \iff b = a$. That symmetry can be avoided by using the assignment operator, :=, instead. This and similar operations, where the purpose is to specify what computations to be done, rather than declaring a model, is written in a section called algorithm. Algorithms typically also contains traditional loops such as for and while.

3.2 eXtensible Markup Language, XML

An XML document simply stores information marked up in a text by using tags. Tags are defined by < *name* > where the name-part of the tag is describing the information content in the XML-element, which is defined by < *name* > *content* < */name* >. Below this section an clarifying example XML document is given. The first two lines of the file only states that the given file is an XML file and provides a given parser with validation file called 'persons.dtd'. The rest of lines describes the content, which was the main focus of this project.

```
<?xml version="1.0"?>
<!DOCTYPE persons SYSTEM "persons.dtd">
<persons>
<person job="programmer">
<name> John Doe </name>
<email>
grigore@none.ro
</email>
</person>
...
<person job="manager">
<comment>Classified<\comment>
</person>
</persons>
```

Similarly to the example above, other types of information can be stored in the XML format. One type is model information from a Modelica model which can be extracted from Dymola in XML. Information-content from the model is stored in an XML-tree, which can be translated into any programming language using a designated parser.

3.2.1 Hierarchy

When processing data from an XML-file, it's important to consider where in the structure the data has been found. This information of location of data is called Hierarchy in this thesis, as it follows from the similarity to hierarchial models in section 3.1. In this example the importance of hierarchy is clearly shown, as the information stored in email is useless unless you also got access to the information of who the email belongs too, which is stored in the hierarchy leading to the specific email element.

3.2.2 XML-generation in Dymola

A Dymola model can be generated as a file with the format XML [5]. For example, the commando `Advanced.OutputNestedExpandedModelAsXML:` enables generation of XML-file where the hierarchy of the model is maintained. Please note, that according to the developer of Dymola, Dassault, the XML-generation of Dymola is under development and might change in future releases. The Dymola version used in this thesis is 2018 alpha, see Table 1.1.

3.3 Parser

A basic parser was implemented by K. Lockowandt see [15]. The parser enabled conversion of models from the XML-format to .mat-format, which is fundamental to the method of this thesis, see Item 1 in Section 4. More specifically, the file formed in the .mat-format defines a *Diagnosis Model* object, which the FDT can be applied to, see Section 3.7.

The parser was developed further during this thesis, which enables handling of

matrix-variables, if-equations and for-loops, as well as handling of the case of having the same fault in multiple equations. For more information on the parser, see Appendix A or [15, chapter 5, p 17].

3.4 Cabin Pressure Control

Given that an air-plane such as Gripen cannot operate without a 'functioning' pilot, certain systems must be operable in order to ensure that the pilot remains in this state. One of these systems is Cabin Pressure Control (CPC), shown in Figure 3.4. The CPC controls the cabin pressure and consists of a Cabin Pressure Controller which is connected to two discharge valves [23]. The discharge valves controls the mass flow rate between the cabin and the ambient pressure by changing its position. The desired position is achieved by comparing a provided servo pressure with the cabin pressure, this comparison enables the discharge valves to change their position and therefore realize the shift in cabin pressure. The Cabin Pressure Controller provides the servo pressure to the valves and therefore controls the shifting of cabin pressure, which explains the name of the component. It is worth noticing that the connection between the controller and the discharge valves is mechanical [24], see the discharge valves in Figure 3.4.

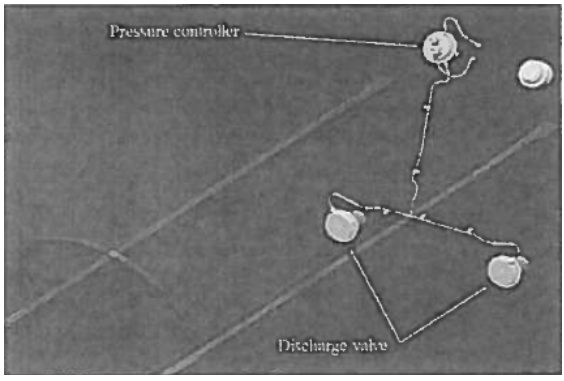

Figure 3.4: *A system overview of the Cabin Pressure Control*

3.4.1 Cabin Pressure Control model

A physical model of the CPC in Gripen E/F has been developed by Saab and implemented in Dymola using components from "Modelica Standard Library" [24]. The model of the discharge valve describes the mass flow rate from the cabin to the ambient as a function of the pressure difference, the valve loss characteristics, and the opening area. The opening area is determined by a balance of the forces acting on a rolling diaphragm. By dividing a part of the internal volume into two chambers, P1 and P2, which are connected to this diaphragm using pistons, variation of the position/lift of the valve can be achieved by the pressure provided to P1 and P2. Both P1 and P2 produces forces that effects the position of the valve.

These forces also opposes each other, which means the resulting position of the valve is related to the relation of the pressures of both chambers. By connecting P1 to the servo pressure, and P2 to the cabin pressure, the position of the valve can be controlled in a desired manner [24]. An overview of the entire CPC model is seen in figure 3.5.

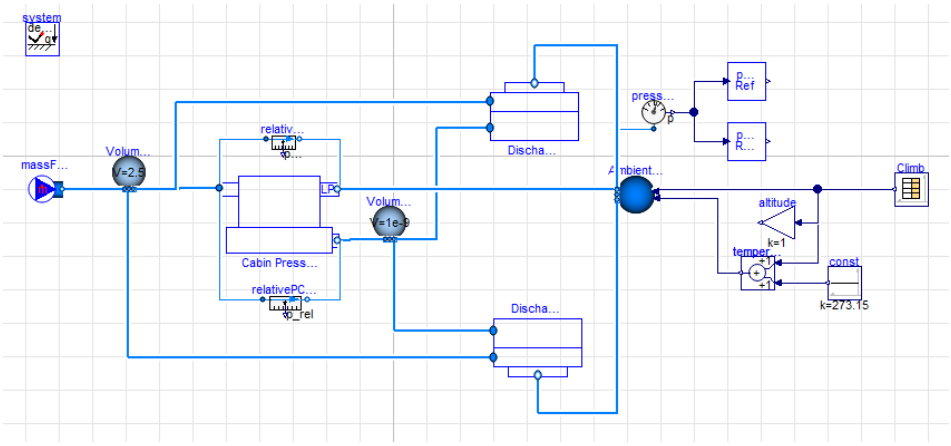

Figure 3.5: *Overview of the CPC model*

In Figure 3.5 the cabin is represented as the volume in the lefthand side of the model. The pressure of this volume is then regulated with respect to the pressure in the ambient volume in the opposite side of the model by the connection through the discharge valves. In order to achieve a valid representation of the discharge valves, the diaphragm is modelled as a spring and damper in parallel. This is connected to a mass with movement restrictions adapted to the possible displacement of the real valve.

3.5 Two Tank System

A simple model is necessary in order to enable testing of the developed algorithms as well as providing additional support for any conclusions made regarding diagnostic system generation. For these purposes, the two tank system, illustrated in Figure 3.6 is introduced. The system includes two tanks with orifices at the bottom of each tank, a pump, and sensors that measures the water levels of the tanks; x_1 and x_2.

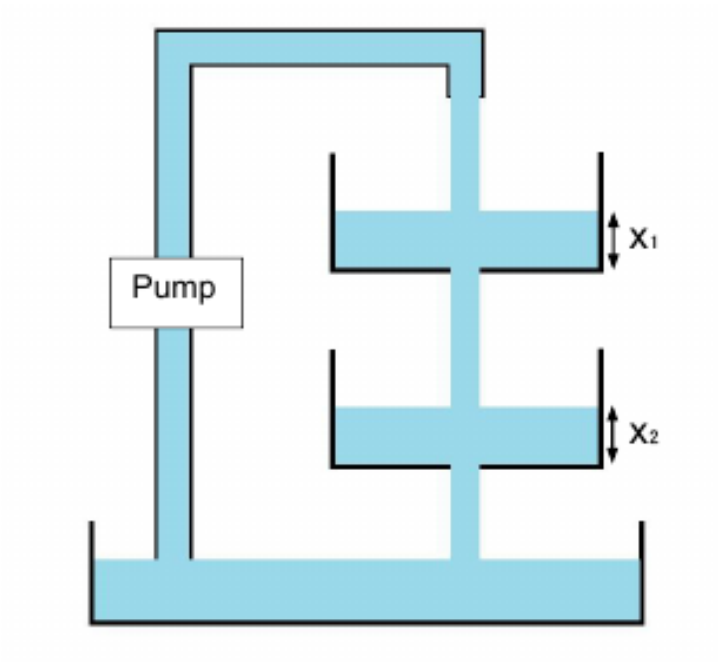

Figure 3.6: *Schematics of the coupled two-tank system. Source [15]*

In addition to the sensors measuring x_1 and x_2, there are also flow measuring sensors, see Table 3.1.

Table 3.1: *Initial sensors in the two-tank system.*

y_1	Water level in tank 1.
y_2	Water level in tank 2.
y_3	Water flow between tank 1 and 2.
y_4	Water flow out of tank 2.

3.5.1 Two Tank Model

The model of the Two Tank System is computed by using the physical relation known as Bernoulli's principle, which relates the outflow from an orifice at the bottom of a tank to the current water level. Assuming that the cross section area of the tank is contant, Bernoulli's principle states the following

$$v(t) = \sqrt{2gh(t)}, \tag{3.1}$$

where the water level is denoted by h, the water flow speed by v and the gravity is described by g. Furthermore the relation between the outflow from the tank $q(t)$ and the water flow speed is given by

$$q(t) = av(t), \tag{3.2}$$

where a denotes the area of the orifice through which the water flows out of the tank. From the assumption regarding constant base area, it is clear that the volume of the water contained in the tank is described by $Ah(t)$ where A is the cross sectional area of the tank. The rate of which the volume changes over time, i.e the time derivative, is

$$A\dot{h}(t) = u(t) - q(t), \tag{3.3}$$

i.e the flow into the tank ($u(t)$) subtracted by the flow out of the tank. Equation (3.1)-(3.3) gives an explicit expression of the water level in the tank:

$$\dot{h}(t) = \frac{1}{A}u(t) - \frac{a\sqrt{2g}}{A}\sqrt{h(t)}. \tag{3.4}$$

Given that there are two water levels to monitor in the given system, one state for each water level is introduced. These are denoted x_1 and x_2 for simplicity. From the previous calculation, the derivatives of these states can be explicitly expressed and therefore the computation of the state space model is complete. The complete state space model is

$$\dot{x}_1(t) = \frac{1}{A_1}u(t) - \frac{a_1\sqrt{2g}}{A_1}\sqrt{x_1(t)} \tag{3.5}$$

$$\dot{x}_2(t) = \frac{a_1\sqrt{2g}}{A_1}\sqrt{x_1(t)} - \frac{a_2\sqrt{2g}}{A_2}\sqrt{x_2(t)}. \tag{3.6}$$

$$y_1 = x_1 \tag{3.7}$$

$$y_2 = x_2 \tag{3.8}$$

$$y_3 = a_1\sqrt{2gx_1(t)} \tag{3.9}$$

$$y_4 = a_2\sqrt{2gx_2(t)} \tag{3.10}$$

where equation (3.5) and (3.6) describes the dynamics of the entire system, while the measurements are described by equation (3.7) to (3.10).

The model description from equation (3.5) to (3.10) can now be expanded by introducing fault signals, i.e. variables that represents physical faults, such as blockage or leakage for instance. In this application the faults described in Table 3.2 are introduced.

Table 3.2: *Faults introduced in the two-tank system.*

$f_{Clogging1}$	Blockage between Tank 1 and Tank 2.
$f_{Actuator}$	Discrepancy in the Pump.
$f_{Leakage2}$	Leakage between the water-flow sensor of Tank 1 and Tank 2.
$f_{WaterLevel2}$	Measurement error in the water-level sensor of Tank 2.
f_{Flow1}	Measurement error in the water-flow sensor of Tank 1.
$f_{Leakage3}$	Leakage between Tank 2 and the water-flow sensor of Tank 2.

By introducing the faults of Table 3.2 in the model described by equation (3.5) to (3.10), the complete model is achieved;

$$\dot{x}_1(t) = \frac{1}{A_1}u(t) + f_{Actuator} - \frac{a_1\sqrt{2g}}{A_1}(1 - f_{Clogging1})\sqrt{x_1(t)} \tag{3.11}$$

$$\dot{x}_2(t) = \frac{a_1\sqrt{2g}}{A_1}(1 - f_{Clogging1})(1 - f_{Leakage2})\sqrt{x_1(t)} - \frac{a_2\sqrt{2g}}{A_2}\sqrt{x_2(t)} \tag{3.12}$$

$$y_1 = x_1 \tag{3.13}$$

$$y_2 = x_2 + f_{WaterLevel2} \tag{3.14}$$

$$y_3 = a_1\sqrt{2gx_1(t)}(1 - f_{Clogging1}) + f_{Flow1} \tag{3.15}$$

$$y_4 = a_2\sqrt{2gx_2(t)}(1 - f_{Leakage3}) \tag{3.16}$$

Given that $f_{Clogging1}$ exists in multiple equations, the assumption from 2.13 is not fulfilled. There are however, tricks to bypass this problem, which is covered in Section 3.7.

3.6 Secondary Enviromental Control System

The Secondary Enviromental Control System (SECS) in Saab 39 Gripen fulfills the function of supplying electronics and radar with cooling air. Hot air from the engine is processed in the system, consisting of two coolers in series with a turbo element that connects the two coolers, see Figure 3.7. This model was also used in a parallel thesis, see [15], and the description of the model from that thesis is given in the next section for simplicity.

3.6.1 Model of the Secondary Environmental Control System

To model the air supply and cooling system, i.e the Environmental Control System (ECS) of the air-plane Saab 39 Gripen, Saab uses the modelling language Modelica. The ECS can be divided into a Primary (PECS) system, a Secondary (SECS) system and a Liquid Loop (LL), allowing each sub-system to be modelled separately. For different purposes more or less detailed models of the system is necessary. Therefore a detailed high fidelity (HFM) and a reduced low fidelity model (LFM) was developed. The HFM is used mainly for testing performance requirements, while the LFM is mostly used in simulators. This thesis focuses mainly on the SECS LFM.

Model Description

In Figure 3.7 the main part of the SECS-model is depicted. Hot and pressurised air is taken from the engine and cooled down. The cold air is used to cool down the liquid in the LL, which in turn is used to cool equipment, not depicted in the schematics. Three types of sensors are used to monitor the system, namely ten temperature sensors, four pressure sensors, and one accelerometer. The high amount of sensors monitoring the system makes the model useful for model-based fault diagnosis.

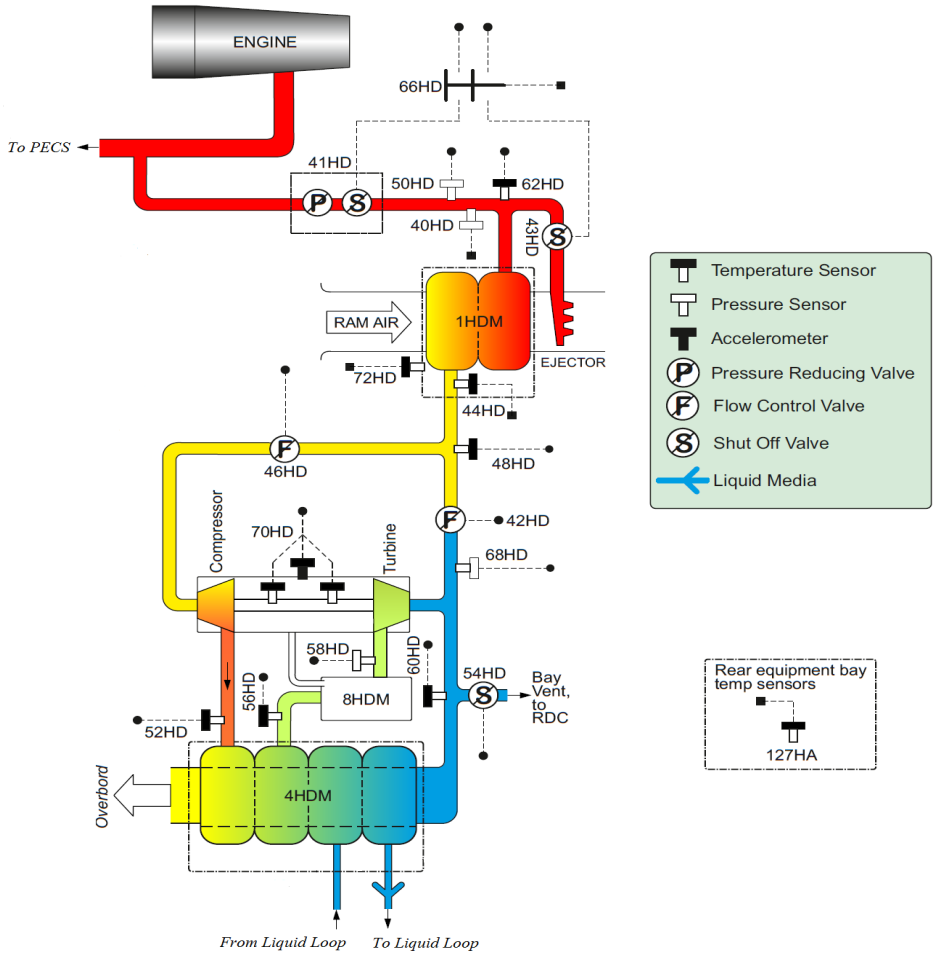

Figure 3.7: *Schematic of the Secondary Environmental Control System.*

3.6.2 Addition of Monitored Faults

Based on a request from Saab the faults of Table 3.3 are implemented into the model in order to achieve a hint of the current diagnosis potential in the system.

Table 3.3: *Faults introduced in the SECS*

$f_{mFlowEjector}$	Deviating mass-flow through the ejector
$f_{mFlowPrecooler}$	Deviating mass-flow through the precooler
$f_{etaPreCooler}$	Deviating degree of efficiency in the precooler.
$f_{mFlowPack}$	Deviating mass-flow through the cooling pack
$f_{etaTurb}$	Deviating degree of efficiency in the turbine

The faults presented in Table 3.3 are marked in Figure 3.8, as the model is confidential, a more detailed explanation of what specific equations that are effected of the fault cannot be given.

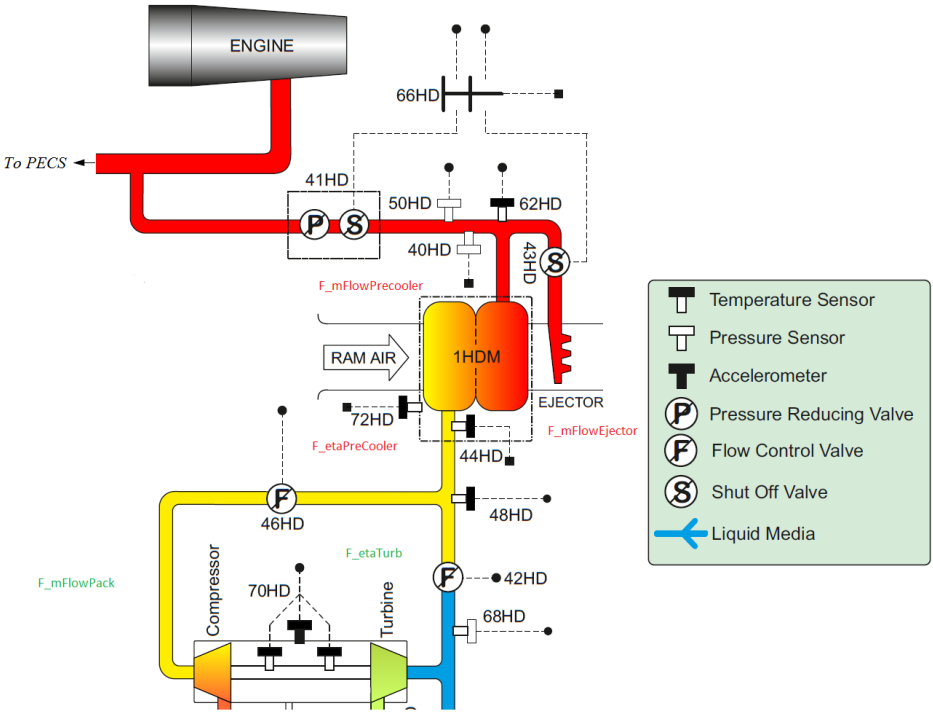

Figure 3.8: *The SECS model with faults of Table 3.3 implemented*

3.7 Fault Diagnosis Toolbox

The Fault Diagnosis Toolbox (FDT) is implemented in Matlab and enables computation of a diagnostic system based on a model of the diagnosed system. The FDT uses structural analysis to decide which equations that should be used in each residual. In the current version (2016-12-22) each fault must exist in only one equation as addressed in [6]. Given that the model description of the Two Tank System (3.11) - (3.16) does not fulfill this, either the chosen model or the chosen methodology (FDT) must be modified. In this thesis, the trick from 2.13 is used, which means that each of the faults that are present in many equations are substituted with a new unknown variable. This would result in a loss of redundancy if it was not for the fact that an additional equation also is introduced for each of these faults. These equations state that the substitute-variables are equal to the faults that they substitute. For the Two Tank System the adequate change can be achieved by adding the following equation

$$f_{Clogging1} = fault_0 \tag{3.17}$$

where $fault_0$ is the new fault signal, and $f_{Clogging1}$ is just a conveniently named variable.

By adding specifications that are redundant from a mathematical point of view, but necessary from a structural viewpoint; namely $\dot{x}_i(t) = \frac{dx_i(t)}{dt}$, the model of the Two Tank System can be written in FDT compatible form. Having the model in this form

$$\dot{x}_1(t) = \frac{1}{A_1}u(t) + f_{Actuator} - \frac{a_1\sqrt{2g}}{A_1}(1 - f_{Clogging1})\sqrt{x_1(t)} \tag{3.18}$$

$$\dot{x}_2(t) = \frac{a_1\sqrt{2g}}{A_1}(1 - f_{Clogging1})(1 - f_{Leakage2})\sqrt{x_1(t)} - \frac{a_2\sqrt{2g}}{A_2}\sqrt{x_2(t)} \tag{3.19}$$

$$y_1 = x_1 \tag{3.20}$$

$$y_2 = x_2 + f_{WaterLevel2} \tag{3.21}$$

$$y_3 = a_1\sqrt{2gx_1(t)}(1 - f_{Clogging1}) + f_{Flow1} \tag{3.22}$$

$$y_4 = a_2\sqrt{2gx_2(t)}(1 - f_{Leakage3}) \tag{3.23}$$

$$f_{Clogging1} = fault_0 \tag{3.24}$$

$$\dot{x}_1(t) = \frac{dx_1(t)}{dt} \tag{3.25}$$

$$\dot{x}_2(t) = \frac{dx_2(t)}{dt} \tag{3.26}$$

enables clarification regarding syntax and functions of the FDT, which will aid the explanation of the implemented algorithms.

A model-object for the model described by (3.18) to (3.26) will now be specified in Matlab using the FDT. The FDT must have access to the information whether

a variable is part of the unknown variables, known variables, or fault variables, which corresponds to the variable being part of $\boldsymbol{X}$, $\boldsymbol{Z}$, $\boldsymbol{F}$ respectively. For this model, this specification corresponds to the Matlab code provided in Figure 3.9, assuming that the prefix *d* is used to indicate a derivative.

```
modelDef.x={dx1,x1,dx2,x2,f_{Clogging1}};
modelDef.f={fault_0,f_{Actuator},f_{Leakage2},f_{WaterLevel2},...
f_{Flow1},f_{Leakage3}};
modelDef.z={u,y_1,y_2,y_3,y_4};
```

Figure 3.9: *Short example of definitions used for the FDT*

The parameters and model equations are entered in a similar manner, see [6]. The FDT supports two kind of models, *symbolic* and *structural*, where a symbolic model contains a complete model, while a structural model only contains the structure of a model. Given that the entire Two Tank System model is available, both structural and symbolic model objects can be created for this model using the FDT. For more information regarding the methods available for the created object see [6, p. 4] or the matlab commando *methods DiagnosisModel*.

4

Method

This section treats the methodology used in the project. In order to distinguish between the methods that were pre-existent to the thesis, and those that were invented during the project, any method mentioned is considered invented unless stated otherwise. The general methodology can be concluded in the following points, see the following subsections for further information.

1. Convert the model from Dymola to Matlab.
2. Analyze the model using the FDT.
   What can be said about the model index, MSOs, causality properties?
3. Extract the desired parts, that is the MSOs that will result in a diagnostic system that has sufficient detectability and isolability.
4. Analyze which type of residual, that results in desired behaviour for the current application; observer, sequential or other type.
5. Generate residuals based on the selected MSO-sets.
6. Convert residuals from Matlab to Dymola and compare the results of the residuals.

4.1 Processing of Model using Fault Diagnosis Toolbox

The idea is to convert the model from .mo to .mat format by using a parser developed in Python. Once the model is converted into matlab, the FDT can be used to compute the MSOs. These MSOs are analyzed and sorted such that the chosen subset of the MSOs results in residuals with sufficient isolability and as high

causality as possible. High causality refers to algebraic being prefered before Integral which in turn is prefered before mixed causality. Derivative causality is considered the lowest based on the experience that differentiating of noisy signals often results in poor accuracy, and therefore should be avioded.

The MSOs chosen by the algorithm are automatically converted into sequential residuals. By allowing the FDT to do its test selection on a sorted subset of the MSOs, the isolability and causality of the generated residuals can be guaranteed simultaneously. The figure 4.1 below describes the general algorithm for generation of a diagnostic system based on a FDT-model object, a set of MSOs, and a desired isolability. Apart from the node *Generate tests*, which corresponds to the FDT method `TestSelection`, the nodes of Figure 4.1 represent algorithms implemented in this thesis. The MSO-computation is described in Section 4.1.1, the sorting of the MSOs is described in Section 4.1.2, and the residual generation is described by Section 4.1.3. The resulting residuals generated by this algorithm are then converted into .mo-format, which enables simulation in Dymola.

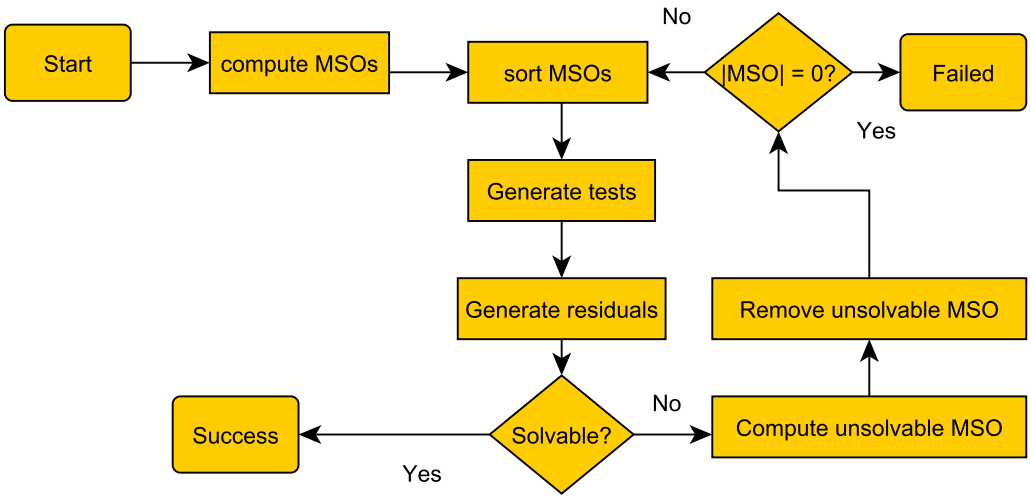

Figure 4.1: *General view of the diagnostic system generation*

4.1.1 MSO-computation

Once the description of the model has been converted into Matlab, the FDT can be used to compute a suitable diagnostic system. As mentioned earlier, only the overdetermined part of the model can be used to make residuals. This section describes the computation of minimal overdetermined sets of equations, (MSOs).

The FDT supports computation of all MSOs by using the member-function MSO, see [6]. However, as stated in [6, 17] the complexity of MSO-computation is often exponential with regards to the degree of redundancy in the model. Therefore, a

new algorithm that computes a reduced number of MSOs was developed during the thesis.

The algorithm used the method `Minimal Test Equation Support`, MTES, from the FDT to extract overdetermined subsystems of the original system. Additionally, sensors that did not contribute to the potential detectability or isolability according to the FDT were removed, which together with extraction resulted in reduced redundancy. By applying the MSO-method on these subsystems, a subset of the MSOs, which results in the same potential isolability and detectability as the sets generated by the MSO-method, can be computed with a lower time complexity.

4.1.2 MSO-sorting

As motivated in the introduction to Section 4.1 desirable properties of the generated residuals are high causality as well as sufficient resulting isolability. Based upon this a sorting algorithm was developed. This sorting algorithm works in two steps. First it divides the MSOs into subsets based on the best achievable causality in each MSO. Then it checks whether the subset of MSOs with best causality alone will result in sufficient isolability. If that is the case then the current subset is selected, if not then the subset is expanded by the subset with second best causality and the process is repeated until either sufficient isolability is achieved or all MSOs are included. The included MSOs are used for residual generation which is covered in the following sections.

4.1.3 Sequential residuals

Sequential residuals are generated using the method `seqresgen` from the FDT. By removing one equation from the MSO, the number of independent equations and unknowns are identical in the remaining set, which is a necessary but not sufficient condition for the equation system to be solvable. As it is not sufficient, the validity of the result generated by the solver is tested in the algorithm. It is also worth noticing that the residuals created by the FDTs *seqresgen* method assumes that the derivatives are constant between its state estimates [6], similar to the Euler-forward method.

Removal of MSOs

As stated earlier, the number of independent equations and unknowns being equal is not a sufficient condition for the system to be (uniquely) solvable. Therefore the algorithm is developed in such a way that if Matlabs solver `solve` (which is used in the `seqresgen` method) fails to find a solution to the equation system, the MSO that was used as input to the method is discarded. In the event that this results in the remaining set of MSOs being empty, the algorithm has failed, else the sorting and generating process is repeated, see figure 4.1.

4.1.4 Observer based residuals

Given that the method `ObserverResGen` from the FDT can only generate observers based on low index problems, another method must be used for problems of high index. As stated in Section 2.4.1 the method described by (2.18), for computation of the D-coefficient ($F(t)$) in the feedback used in the observer, see first subsection of this section. The P-coefficient ($G(t)$) is computed by using Lanner-hed's theorem, that will be presented in Section 5.1, in combination with PMP. A summary on the computation of the P-coefficient is provided in the second subsection of this section.

Computation of F

From Theorem 1 given in Section 2.4.1, a basic algorithm for computing $F(t)$ has been implemented. The algorithm rewrites the set of equations of an MSO to the form described in (2.18). Once this step is complete, the symbolic toolbox in Matlab enables calculation of the null space of the gradient of h by solving the following equation;

$$h_x(x_{1_i}, x_2) \cdot \begin{bmatrix} x_{1_i} \\ x_2 \end{bmatrix} = \vec{0}\,, \; x_{1_i} \neq 0 \tag{4.1}$$

where x_1 represent the states, x_{1_i} represents a specific state, and x_2 represent the algebraic variables. Based on equation (4.1) it can be determined what states x_{1_i} that are part of the null space;

$$\begin{aligned} A &= \{\, i : \textit{equation } (4.1) \textit{ fulfilled for } x_{1_i} \,\} \\ V &= \cup x_{1_i} \; \forall i \in A \\ W &= x_1 \setminus V. \end{aligned}$$

Theorem 1 now states that any $F(t) : \mathrm{Im}\ F(t) = W$ results in a low-index observer, which completed the computation of the necessary demands on $F(t)$. Note that these demands only resulted in limitation regarding the dimension of the coefficient. For simplicity, only ones and zeros were used as elements in $F(t)$ in this thesis. The flow chart of the algorithm is concluded below.

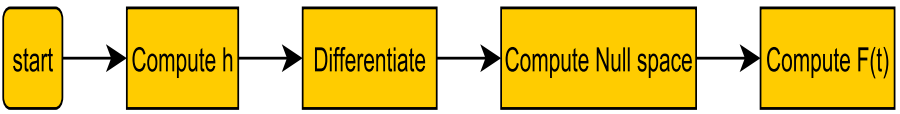

Figure 4.2: *Flow chart, computation of F(t)*

The algorithm in Figure 4.2 has similar limitations as the FDT `seqresgen` method, given that the Matlab methods `solve` and `jacobian` is used in the computation of V, see (4.1).

Computation of G

When Calculating G a different approach is taken. The equations of the chosen MSO are complemented with measurement error r which provides an exactly determined equation system which is solvable in Dymola. However, as G is considered to be unknown at this point, at least one additional equation per state must be added in order to achieve solvability. Instead of calculating G through linearization of 2.18 as in [12], an approach based on Pontryagin's Minimum Principle (PMP) [9], was used in this thesis.

PMP is used to compute G_i, where i represents the index of each observed state. By utilizing the theory described in Section 5.1, it is shown that the following equation

$$G_i = \frac{-2f_i(\hat{x}_1, \hat{x}_2, z, t) - F_i(t)\dot{r}}{r} \tag{4.2}$$

describes the G_i that results in the integral of $(F(t)\dot{r} + G(t)r)^2$ being minimal over the chosen simulation interval. The details of the computation of G_i is found in Section 5.1.1, while Section 5.1.2 covers the same application with a slightly modified objective function.

4.1.5 High-index residual implementations

When generating observer based residuals, there is currently no support for high-index MSOs in the FDT according to [6] . However, given that the equations are written in the semi-explicit form as in (2.17), the method described in [12] can be used to attempt index reduction. The methodology in Section 4.1.4 can therefore be used to provide observer solutions to high-index problems. The FDT method `seqresgen` is not limited to low-index problems in the same way, and therefore there is no need to adapt the generation method of sequential residuals to high-index problems.

Given that the sequential residuals can be generated for both high and low index MSOs, these are ideal to use when comparing the performance of the current Matlab methods of the FDT and the new Dymola methods. This is therefore of great importance to the investigation of Research Question 5.

4.2 Conversion from Matlab to Dymola

Once residuals with desired properties have been generated the last step in the new method is to convert the corresponding MSOs back to Dymola. This section covers this translation in more detail than previous sections. For clarity, the resulting Modelica code of a residual based on (3.21) and (3.23) from the two tank system, hereby referred to as the *Dymola example*, is included below.

```
model sequential_2
// Parameters
parameter Real twoTank_withfaults__a2 = 3.0000e-02,twoTank_withfaults__g = 9.8100e+00;
// Variables
Real twoTank_withfaults__x2(start=6);
// Measured signals
Modelica.Blocks.Interfaces.RealInput twoTank_withfaults__y2,twoTank_withfaults__y4;
//output
Modelica.Blocks.Interfaces.RealOutput r;
//Equations
equation
twoTank_withfaults__y4 = 2^(1/2)*twoTank_withfaults__a2*(twoTank_withfaults__g*twoTank_withfaults__x2)^(1/2);
twoTank_withfaults__y2 = twoTank_withfaults__x2+r;
end sequential_2;
```

The variables and the parameters are handled in the following fashion; the equations of the chosen MSO is searched for members of the models unknowns, X as well as its parameters. If one of these members are found in the MSO, a corresponding variable or parameter are declared as a Real parameter or Real variable. In the *Dymola example*, the parameters correspond to $RealtwoTank\_withfaults\_\_a2$ and $twoTank\_withfaults\_\_g$, while $twoTank\_withfaults\_\_x2$ represent the only variable of the system.

Given that Dymola could not handle overdetermined systems, an additional unknown variable was added in order to avoid singularity. This was achieved by declaring a new Real output named r and adding this into the existing equation system. By letting r affect only one equation in an additive manner, great similarities with the FDT `seqresgen` method is achieved, as this equation then corresponds to the *resEq*-argument of the method, see [6]. The introduction of the residual variable r therefore fulfilled two purposes. It enabled the achievement of an exactly determined system, which is mathematically necessary, but it also captured the information whether the equations of the MSO were fulfilled or not. As stated in Section 2.14 certain observations z must be provided in order to compute the value of the residual. These are declared as inputs in the model object, see $twoTank\_withfaults\_\_y2$ and $twoTank\_withfaults\_\_y4$ of the *Dymola example*.

By gathering the translated and modified MSO into a new Dymola model, with input z and output r, a representation of the residual in Dymola was achieved. In order to realize the system, known z must be provided to the model. This is accomplished by using tables that contains these values and are connected to the created model, see Figure 4.3. It is possible to connect the measurements of a simulation directly into the residual object. However, this is not treated in this thesis as the adaptive step-length in Dymola (see Table 1.1) would bias the comparison between the results in Matlab and Dymola, see Research Question 5.

The construction described in Figure 4.3 enables evaluation of multiple residuals at once, and has access to derivative-approximations of the provided measurements, which are provided by the combiTimeTables (see [5]). The combiTimetables are then connected to the translated residuals, which will have the form of the Dymola example and are represented by the blue boxes to the right in Figure 4.3. In this thesis, the measurements in the combiTimeTables are generated through a simulation of the provided model, although real measurements can be

inserted and used in the method in an identical manner.

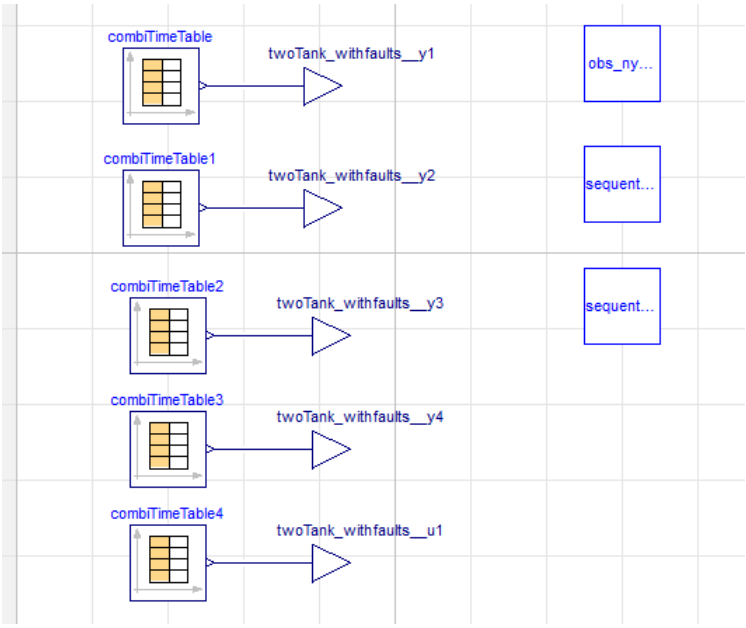

Figure 4.3: *Residual Evaluation in Dymola for Two Tank System*

4.3 Method Discussion

The current choice of method is a trade-off between applying general algorithms and being finished within the time-limit of the project. General algorithms enables testing of the variety of cases that appears when using multiple models, but it also typically consumes more working hours than more model-specific solutions. In order to verify that the algorithm works for the general case and not just a specific model, a variety of models of different complexity are used during the testing of the algorithm. Applications on simpler models also added certainty to the results regarding the more general research questions, such as the comparison between the current FDT implementations (Matlab) and the implementations in Dymola, see Research Question 5. If only one model would be tested, the risk of the results being biased would be much higher. Also, by testing the implementations on multiple models the robustness of the implemented algorithms could be secured. This section will evaluate the advantages and disadvantages of a few of the design-choices that were made during the thesis.

4.3.1 Structural Methods

While structural methods enables diagnostic analysis without unreasonable complexity there is one serious drawback when using these methods. This drawback

lies in the fact that the unknown variable x being present in equation e_i is a necessary but not sufficient requirement for computing and extracting x from e_i. The result of this being that what is viewed as an exactly determined system from a structural viewpoint might not be uniquely solvable from a symbolic point of view as mentioned in Section 4.1.3. For example, this situation occurs for all general linear systems $Ax = b$ where b is a column vector of length n and A is a n times n matrix with linearly dependent rows. The implemented method however means that potential information is lost, as sets of equations are discarded. This clearly presents a conflict between accuracy and complexity, as evaluating if x really is extractable from e_i increases the complexity, and therefore inhibits the main purpose of using structural methods in the first place.

4.3.2 Causality Sorting

The Causality sorting algorithm is based on the assumption that numerical differentiation of noisy signals often results in poor accuracy, lower Signal-to-Noise Ratio (SNR) and typically requires lowpass-filtering of the signal. By minimizing the number of numerical differentiations needed in the implementation, the accuracy-loss is avoided.

4.3.3 Observer Generation

The method chosen for generation of F enables potential index reduction for high-index problems, if the problems are formulated in the semi-explicit form. However, given that PMP is used to calculated the propertional term G, and PMP only provides necessary and not sufficient conditions for optimality, there are potential flaws in the algorithm regarding if a found optimum is global. However one of the benefits of the PMP implementation is that it is less dependent on a chosen operating point compared to other methods, such as the method described in [12] and the method that is currently implemented in FDT, see the FDT method `ObserverResGen` in [6]. By formulating an optimal control problem, the chosen G is a result of the combination of multiple operating points, which are selected by Dymolas solver (Dassl) based upon the available measurement data. If an implementation requires desired behaviour such as stability in multiple operating points, the PMP method provides a way to allow each operating point to contribute in a more even manner compared to linearization, which only takes one operating point into account.

Another benefit of the PMP apporach used in this thesis is the properties that follows from one of the chosen objective functions $\int_{t_i}^{t_f} (F(t)\dot{r} + G(t)r)^2 dt$. These properties follows from the fact that this objective function reaches its minimum value at $(F(t)\dot{r} + G(t)r)^2 = 0 \leftrightarrow F(t)\dot{r} + G(t)r = 0$, which means that the chosen observer will regulate r in the following manner,

$$F_i(t) = 0 \implies r \to 0 \tag{4.3}$$

$$F_i(t) \neq 0 \implies F_i(t)\dot{r} + G_i(t)r \to 0 \tag{4.4}$$

where $F_i(t)$ is assumed being a predetermined value.

Equation (4.3) describes the typical desired property of an observer based residual, namely that the residual variable will be pushed towards zero. The properties that follow from equation (4.4) on the other hand states that the behaviour of r will be pushed towards the behaviour of the linear system described by $F_i(t)\dot{r} + G_i(t)r = 0$. Given that $F_i(t)$ can be set arbitrary (zero excluded), this means that the pole of the linear system can be placed arbitrary as well.

4.3.4 Conversion to Dymola

By converting equations to Dymola, high-index DAE problems can be solved [5]. According to [5] Dymola enables the usage of a variety of numerical integration methods. Although as mentioned in Table 1.1, only the standard Dassl-solver was investigated in this thesis. By looking up the constant eps in the Modelica constant library (Dymola 2018 Alpha) and the corresponding coefficient in Matlab (Matlab R2016b) the results in Table 4.1 was found.

Table 4.1: *Comparison of machine epsilons Matab/Dymola.*

Language	Value of eps
Matlab	$2.2204 \cdot 10^{-16}$
Dymola	$1 \cdot 10^{-15}$

5
Results

This section presents the achieved results of the thesis. The thesis has treated several models, including the Two Tank Model from Section 3.5.1, the Cabin Pressure Control (CPC) from Section 3.4.1, and the Secondary Enviromental Control System (SECS) from Section 3.5.1. These models were used in different ways throughout the thesis, the SECS and Two Tank Model were converted successfully to Matlab and therefore used to answer the questions that required application of the FDT. The CPC was only used to investigate whether the parser method had the potential of being applicable to Dymola models in general.

During the project a few theoretical achievements were also made, as optimal control theory was used to determine the observer gain in a observer based residual. This result is summarized in Section 5.1.

5.1 Theory

The theoretical results of this thesis is an addition of the results of E. Frisk and J. Åslund in [12]. A summary is given here for convenience. The theory treats computation of observers based on a model of semi-explicit form;

$$\begin{aligned} \dot{x}_1 &= f(x_1, x_2, z, t) \\ 0 &= h(x_1, x_2, z, t) \end{aligned} \tag{5.1}$$

where x_1 represents states, x_2 represents other algebraic variables, while z represents known signals and time is represented by t. By letting the estimation error r affect the state-estimation $\hat{x}_1$ in the following manner

$$\begin{aligned} \dot{\hat{x}}_1 &= f(\hat{x}_1, \hat{x}_2, z, t) + F(t)\dot{r} + G(t)r \\ 0 &= h(\hat{x}_1, \hat{x}_2, z, t, r) \end{aligned} \tag{5.2}$$

both low-index and asymptotic stability can be guaranteed under certain conditions according to [12].

The conditions for achieving low-index is provided by Theorem 1 in Section2.4.1. In short that theorem states which conclusions that can be drawn based on the design of the parameter $F(t)$ given that $f(\hat{x}_1, \hat{x}_2, z, t)$ and $h(\hat{x}_1, \hat{x}_2, z, t, r)$ fulfills certain requirements. If these requirements are fulfilled, the problem described in equation (5.2) (see also Example 5.2) can be guaranteed to have low index given that the coefficient $F(t)$ fulfills the demand that its image covers the span of W. Where W is defined by the states x_{1_i} that under no circumstance is part of the null space of the gradient of h; h_x, see equation (2.20).

Theorem 5.1 (Lannerhed's theorem). *Let E be an MSO set of equations in the form 5.1 that has the sub-sets $\boldsymbol{X}'$ and $\boldsymbol{Z}'$ in the set of unknowns $\boldsymbol{X}$ and in the set of known signals $\boldsymbol{Z}$ respectively. Furthermore $\boldsymbol{X}'$ and $\boldsymbol{Z}'$ fulfill*

$$\boldsymbol{X}' = \{x \in \boldsymbol{X} \ : \ x \text{ exists in } E\}$$
$$\boldsymbol{Z}' = \{z \in \boldsymbol{Z} \ : \ z \text{ exists in } E\}$$

Let e_k represent the state equation $\dot{x}_{1_k} = f_k(\ldots)$. Then if the following prerequisites are fulfilled

1. *$\exists i : x_{1_i} \in W$ and the equations $E \backslash \{e_i\}$ describes an injective mapping $\boldsymbol{X}' \rightarrow \boldsymbol{Z}'$.*

2. *Based on E an observer has been designed that*

   is based on the model in Equation 5.2 and

   fulfills Theorem 1.

then the set of equations describing the observer can be expressed as a state space model.

Proof: From equation (2.20) it is clear that the image of W is the set of the states that are not part of the nullspace of h_x. Given that it is only these states that have derivatives that are effected by $\dot{r}$, the fact is that those corresponding states x_{1_i} can be computed using a different equation than the one containing $\dot{r}$. This fact follows from $h_{x_{1_i}} \neq 0 \ \forall x_2$ and Prerequisite 1. With no demands regarding resulting causality, this means that x_{1_i} and its derivative can be considered known in the equation which contains the time-derivative of x_{1_i}, e_i. Given that this equation must have $F_i \neq 0$ according to prerequisite 2 it follows that $\dot{r}$ can be explicitly expressed by this equation. The expression can then be used to gauss-eliminate all other cases of $F_j \neq 0 : j \neq i$, and the problem can therefore be rewritten in state-space form. ■

Lannerheds Theorem will now be clarified by an application on an MSO in the form of equation (5.1), see Example 5.2.

Example 5.2

Let

$$\dot{x}_{1_1} = x_{1_2} \tag{5.3}$$
$$\dot{x}_{1_2} = g(x_{1_1}) \tag{5.4}$$
$$0 = x_{1_2} - y_2 \tag{5.5}$$

be an example system in the form of (5.1). For this system it is clear that

$$x_1 = \begin{bmatrix} x_{1_1} \\ x_{1_2} \end{bmatrix}$$
$$x_2 = \emptyset$$
$$h = x_{1_2} - y_2$$

Now, let an observer of the system in the form of equation (5.2) be expressed as

$$\dot{\hat{x}}_{1_1} = \hat{x}_{1_2} + F_1\dot{r} + G_1 r \tag{5.6}$$
$$\dot{\hat{x}}_{1_2} = g(\hat{x}_{1_1}) + F_2\dot{r} + G_2 r \tag{5.7}$$
$$0 = \hat{x}_{1_2} - y_2 + r \tag{5.8}$$

Given that x_{1_2} and not x_{1_1} is present in the algebraic constraint h, it is clear that

$$h_x = h_{x_1} = \begin{bmatrix} 0 & 1 \end{bmatrix} \rightarrow V = \{x_{1_1}\}, W = \{x_{1_2}\}$$

As the linear span (spa) of W can be expressed as

$$W = spa(\begin{bmatrix} 0 \\ 1 \end{bmatrix})$$

it is clear that letting $F = \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \propto \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ results in compliance with Theorem 1, and as the observer is designed according to equation (5.2) the second prerequisite is fulfilled. Now, as the removal of the second state equation (5.4) (The equation that has $F_i \neq 0$) still results in expressibility of the unknown variables, the injectivity can be verified. This is easily seen in this example as $\hat{x}_{1_2}$ is part of a linear function in equation (5.8) and an uniquely defined $\hat{x}_{1_2}$ results in an uniquely defined $\hat{x}_{1_1}$ in equation (5.6), given that an initial state estimate is provided. Therefore, under this mild assumption, the first prerequisite is fulfilled. Lannerhed's Theorem now states that the model of equation (5.6),(5.7), and (5.8) can be expressed in state-space form. Insertion of $F = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ results in

$$\dot{\hat{x}}_{1_1} = \hat{x}_{1_2} + G_1 r \tag{5.9}$$
$$\dot{\hat{x}}_{1_2} = g(\hat{x}_{1_1}) + \dot{r} + G_2 r \tag{5.10}$$
$$0 = \hat{x}_{1_2} - y_2 + r \tag{5.11}$$
$$(5.11) \implies \hat{x}_{1_2} = y_2 - r \tag{5.12}$$

By using equation (5.12) $\hat{x}_{1_2}$ can be eliminated, which results in

$$\dot{\hat{x}}_{1_1} = y_2 + (G_1 - 1)r \tag{5.13}$$

$$\dot{r} = \frac{\dot{y}_2 - g(\hat{x}_{1_1}) - G_2 r}{2} \tag{5.14}$$

The equations (5.13) and (5.14) expresses a state space model, which means that Lannerhed's Theorem has been verified for this example. It is worth noticing that equation (5.14) could be used for gauss-elimination of other cases of $\dot{r}$, although this is not necessary in this example.

With the model written in state space form, an optimal control problem can be formulated in order to compute the observer gain $G(t)$.

Corollary 5.3. *Assume that a set of observer equations fulfils Theorem 5.1 and that a Hamiltonian that is convex with respect to observer gain G has been formed. Then the optimal G can be computed using Pontryagins Minimum Principle.*

Proof: The evidence of PMP providing necessary requirements for an optimum is given in [9]. The fact that the Hamiltonian is convex with respect to the observer gain G results in the local minimum being global according to the fundamental properties of a convex function. ■

5.1.1 Application of Theory: Optimization towards Stability

While Corollary 5.3 provides the theory needed for observer design, it leaves up to the designer to choose among all the possible objective functions that leads to a convex Hamiltonian. In this specific case, the square of the addition made to the differential equation by the measurement error is used as f_0, see equation (2.21). The optimal control problem can therefore be formulated in the following way;

$$\min_{G(t)} . \int_{t_i}^{t_f} (F(t)\dot{r} + G(t)r)^2 dt \quad subject\ to \begin{cases} \dot{\hat{x}}_1 = f(\hat{x}_1, \hat{x}_2, z, t) + F(t)\dot{r} + G(t)r \\ 0 = h(\hat{x}_1, \hat{x}_2, z, t, r) \end{cases} \tag{5.15}$$

The initial/end time of the integral and specific placement of r in h are design-parameters as well as the specific choice of $F(t)$, although the dimension-requirement of Theorem 1 must be fulfilled in order to guarantee low index. These requirements in combination with Theorem 5.1 and Corollary 5.3 enables the application of PMP to the problem. The pointwise minimization of the Hamiltonian, H, based on $G(t)$ provides the last necessary equation. In order to clarify the difference between description of the quantities and the restriction that must be true in the optimum, $*$ is used for notation of the optimum. The resulting equation

system is

$$A = F(t)\dot{r} + G(t)r \tag{5.16}$$

$$(2.22)\,(5.16) \implies H(\lambda, \hat{x}_1, \hat{x}_2, r, G(t)) = A^2 + \lambda(f(\hat{x}_1, \hat{x}_2, z, t) + A) \tag{5.17}$$

$$(2.23) \implies H^* = const \tag{5.18}$$

$$\frac{\partial H}{\partial G}(G(t)^*) = 0 \tag{5.19}$$

Based on equation (5.19) it follows that;

$$\frac{\partial H}{\partial G} = 2r(F(t)\dot{r} + G(t)^* r) + \lambda r = 0 \iff G(t)^* = \frac{-\frac{\lambda}{2} - F(t)\dot{r}}{r}, r \neq 0 \tag{5.20}$$

The fact that nothing can be said about G when $r = 0$ does not constrain the solution, as it is clear from equation (5.16) that G only has effect if $r \neq 0$. It is also logical that this case is ignored, as $r = 0$ is equivalent to all the equations in the chosen MSO are fulfilled, which means that there is no measurement error to generate a feedback from. Having r in the denominator can cause problems during the simulation, therefore simplification of equation (5.16) with the result of equation (5.20) might be necessary in order to cancel out r from the denominator of $G(t)^*$. Based upon this, it is easy to confirm that the optimization problem is convex through $\frac{\partial^2 H}{\partial G^2} = 2r^2 > 0,\ r \neq 0$ which confirms that any found candidate is a global minimum.

It has been shown that the control $G(t)^*$ is optimal to the given optimal control problem, and $G(t)^*$ has been expressed as a function of λ. Computation of λ through (5.18) provides the resulting optimal control;

$$(5.18) \iff \left| (5.16), (5.20) \iff A^* = -\frac{\lambda}{2} \right| \iff \tag{5.21}$$

$$\iff H = (-\frac{\lambda}{2})^2 + \lambda(f(\hat{x}_1, \hat{x}_2, z, t) - \frac{\lambda}{2}) = const \iff \tag{5.22}$$

$$\iff \frac{d(\lambda f(\hat{x}_1, \hat{x}_2, z, t) - \frac{\lambda^2}{4})}{dt} = 0 \iff \lambda f(\hat{x}_1, \hat{x}_2, z, t) - \frac{\lambda^2}{4} = C \iff \tag{5.23}$$

$$\iff (\lambda - 2f(\hat{x}_1, \hat{x}_2, z, t))^2 - 4(f(\hat{x}_1, \hat{x}_2, z, t))^2 + 4C = 0 \Leftarrow \tag{5.24}$$

$$\Leftarrow C = 0, \lambda_1 = 0, \lambda_2 = 4f(\hat{x}_1, \hat{x}_2, z, t) \tag{5.25}$$

The candidates in equation (5.25) are generated based on the assumption of the constant C being equal to zero. This assumption was motivated by the fact that $C \neq 0$ results in a square expression in the denominator of $\dot{\lambda}$, which with high certainty would disqualify the candidate as the adjoint equations would become much harder to fulfil for the general f, see equation (2.24).

As both candidates for λ resulted in the Hamiltonian becoming constant, other properties of the solution was considered in the final selection. $\lambda = \lambda_1 = 0$ would

result in most of the left-hand side of equation (5.17) being ignored. For the case that the corresponding D-coefficient, F_i is equal to zero this would result in $G_i = 0$ being an global minimum. As both G and F being equal to zero would result in state estimation that completely ignores the measurement error, that combination of parameter values should be avoided. Therefore that candidate is discarded and the following conclusion is drawn;

$$\lambda = \lambda_2 = 4f(\hat{x}_1, \hat{x}_2, z, t) \tag{5.26}$$

Equation (5.26) in combination with equation (5.20) provides the following G

$$(5.20), (5.26) \iff G^* = \frac{-2f(\hat{x}_1, \hat{x}_2, z, t) - F(t)\dot{r}}{r} \tag{5.27}$$

which results in optimal observer properties with respect to the chosen objective function.

Example of Application

By applying equation (5.27) to the model of Example 5.2, the optimal observer gain G^* can be computed.

Example 5.4

Based on Example 5.2 it is clear that

$$f_1(\hat{x}_1, \hat{x}_1, z, t) = \hat{x}_{1_2} \tag{5.28}$$

$$f_2(\hat{x}_1, \hat{x}_2, z, t) = g(\hat{x}_{1_1}) \tag{5.29}$$

$$F_1 = 0 \tag{5.30}$$

$$F_2 = 1 \tag{5.31}$$

Equation (5.27) now enables the computation of the optimal observer gain with respect to the objective function of equation (5.15);

$$G_1^* = \frac{-2\hat{x}_{1_2}}{r} \tag{5.32}$$

$$G_2^* = \frac{-2g(\hat{x}_{1_1}) - \dot{r}}{r} = \frac{-2g(\hat{x}_{1_1}) - (\frac{\dot{y}_2 - g(\hat{x}_{1_1}) - G_2^* r}{2})}{r} \rightarrow G_2^* = \frac{-3g(\hat{x}_{1_1}) - \dot{y}_2}{r} \tag{5.33}$$

5.1.2 Application of Theory: Optimization towards Convergence

This section describes the application of Corollary 5.3 to another objective function. By selecting the objective function in the following manner

$$\min_{G(t)} . \int_{t_i}^{t_f} (F(t)\dot{r} + G(t)r)^2 + W_r r^2 dt \quad subject\ to \begin{cases} \dot{\hat{x}}_1 = f(\hat{x}_1, \hat{x}_2, z, t) + F(t)\dot{r} + G(t)r \\ 0 = h(\hat{x}_1, \hat{x}_2, z, t, r) \end{cases} \tag{5.34}$$

where W is a non-negative weight-coefficient, the optimization should result in r converging to zero at a faster rate. It is easily noticed that $W_r = 0$ corresponds to the same objective function as in Equation 5.15.

Given that the derivative of the Hamiltonian H with respect to $G(t)$ has not changed, Equation 5.20 still provides a valid description of $G(t)^*$. Therefore, by replacing Equation 5.17 with

$$H(\lambda, \hat{x}_1, \hat{x}_2, r, G(t)) = A^2 + W_r r^2 + \lambda(f(\hat{x}_1, \hat{x}_2, z, t) + A) \tag{5.35}$$

the set of equations (5.35), (5.19),(5.18), and (5.16) describes the new system of equations to be solved. The system is not solved analytically in this thesis.

Example of Application

The necessary demands for the optimal observer gain G^* is stated in Example , however no analytical solution to the equation system is presented in this thesis.

Example 5.5

Based on Example 5.2 it is clear that

$$f_1(\hat{x}_1, \hat{x}_2, z, t) = \hat{x}_{1_2} \tag{5.36}$$
$$f_2(\hat{x}_1, \hat{x}_2, z, t) = g(\hat{x}_{1_1}) \tag{5.37}$$
$$F_1 = 0 \tag{5.38}$$
$$F_2 = 1 \tag{5.39}$$

Now, as the presence of $G(t)$ in the Hamiltonian H has not changed, the optimal observer gain G^* is still expressed by equation (5.20). However, the adjoint variable λ is not computed analytically for this case. The resulting equation system to be solved is;

$$\dot{\hat{x}}_{1_1} = \hat{x}_{1_2} + G_1^* r \tag{5.40}$$
$$\dot{\hat{x}}_{1_2} = g(\hat{x}_{1_1}) + \dot{r} + G_2^* r \tag{5.41}$$
$$0 = \hat{x}_{1_2} - y_2 + r \tag{5.42}$$
$$H^* = \begin{bmatrix} H_1^* \\ H_2^* \end{bmatrix} = \begin{bmatrix} (G_1^* r)^2 + W_r r^2 + \lambda_1(\hat{x}_{1_2} + G_1^* r) \\ (\dot{r} + G_2^* r)^2 + W_r r^2 + \lambda_2(g(\hat{x}_{1_1}) + \dot{r} + G_2^* r) \end{bmatrix} \tag{5.43}$$
$$\dot{H}^* = \vec{0} \tag{5.44}$$
$$G(t)^* = \begin{bmatrix} G_1^* \\ G_2^* \end{bmatrix} = \frac{1}{2r} \begin{bmatrix} -\lambda_1 \\ -\lambda_2 - 2\dot{r} \end{bmatrix} \tag{5.45}$$

5.2 Cabin Pressure Control

By using Dymola an XML-file was successfully generated from the CPC-model. When applying the Parser from Section 3.2 to this XML-file, it could be stated that the python parser `xml.etree.ElementTree` was successfully applied to the file, see Section A.1. Complete translation of the model information to Matlab was not achieved during the thesis, however, certain key problems were identified that are necessary to solve in order to achieve a translation that works for general Dymola models. One of these problems was the interpretation of advanced if-statements, see the following example from the CPC-model.

```
if energyDynamics == Dynamics.SteadyState then
0 = Hb_flow + Qb_flow + Wb_flow;
else
der(U) = Hb_flow + Qb_flow + Wb_flow;
end if;
```

In the example of code, only one of the two branches/equations above is active at the same time. Given that these equations contains different set of variables, the structure of the complete model will differ depending on if the condition is true or false. One way to bypass this problem is by performing multiple structural analyses, one for each possible constellation of the conditions being true or false. However, for the CPC model, which according to the performed investigation has over 200 if-statements in this form, this would result in over

$$2^{200} > 10^{60} \tag{5.46}$$

structures to investigate, given that each condition has the values true and false in its range. Some reduction of the number of if-statements was achieved by attempting to evaluate the conditions of each if-statement and only select the active branch during the translation to Matlab. Only 100 conditions were evaluable though, so additional methods or modification of the model is necessary, given that $2^{100} > 10^{30}$ different structures still cannot be investigated within a reasonable time-limit.

The benefits of only translating the model structure includes that no handling of all special functions are necessary. For example, the CPC model includes many thermodynamic descriptions, which of course, are not implemented in Matlab. If a model would be translated as a symbolic model, see Section 3.7, then a complete Matlab implementation of each of these functions would be necessary in order to apply the FDT. It can therefore be concluded that while the CPC-model is translatable to Matlab, there are certain obstacles that must be bypassed in order to achieve a translation of the model. This concludes the investigation of Research Question 1.

5.3 Two Tank Model

This section will present the results of applying the methods in Section 4 on the Two Tank Model from Section 3.5. Given that the Cabin Pressure Controller is only treated as a structural model in this thesis according to Section 1.3, other models must be used to verify the research questions that requires non-structural implementations of the FDT. The investigation of the difference between implementations of residuals in Dymola and Matlab is one clear research question with this requirement. All measurements are based on a simulation of the model with the initial states being set to 6 meters and the input seen in Figure 5.1. Note that no noise is used in this test, which means that differences can be linked directly to the computation methods as there is no stochastic element in the test.

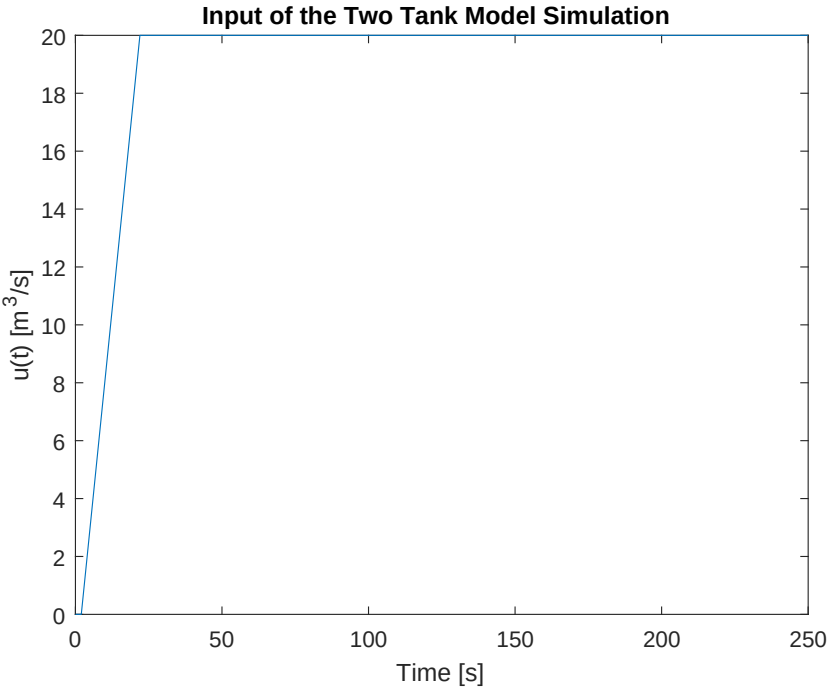

Figure 5.1: *The input used for the simulation of the Two Tank Model*

5.3.1 Comparison of Residual Evaluation, Matlab and Dymola

By combining the equations (3.18), (3.21), (3.22) and (3.25) a non-algebraic MSO is formed which is used for sequential residual generation by using the FDT. The equation (3.20) is chosen as the redundant part, which results in integral causality. As stated earlier, the ramp-input of Figure 5.1 is used in order to achieve measurements, which are used to evaluate the residual. Additional information on the simulation is seen in Table 1.1 and there was no faults or noise implemented

in the simulation. The data provided in Table 1.1 also provides the step-length of $\frac{250}{500} = 0.5$ seconds, which was used by the FDTs sequential residuals. The resulting samples of the measurements were stored and used for evaluation of both the Matlab and Dymola version of the residual. The result is seen in Figure 5.2.

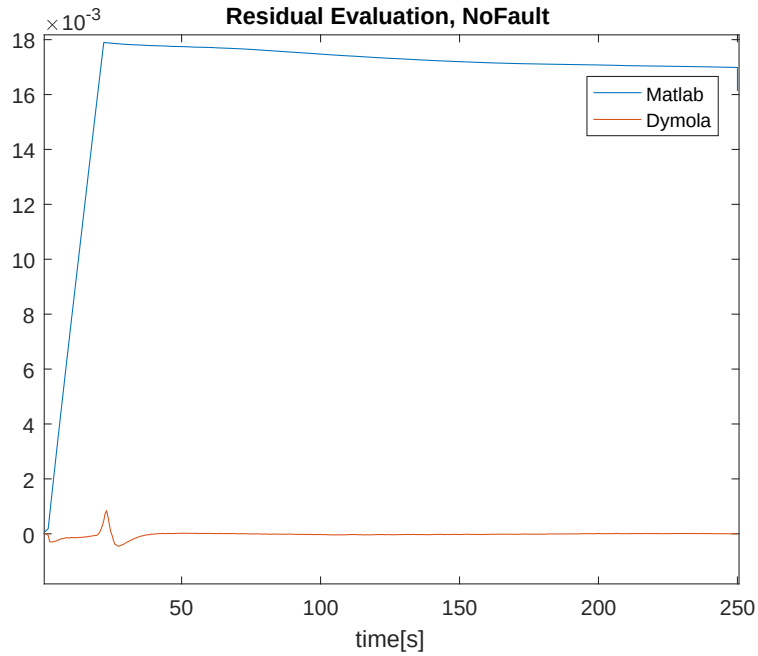

Figure 5.2: *Evaluation of a Non-algebraic Residual in Matlab and Dymola*

The equations (3.21) and (3.23) are also used to make a residual, where equation (3.23) is selected as the redundant part. The evaluation of this residual uses another set of measurements; white noise with a standard deviation of 0.03 and a step-function was added to equation (3.21). The step-function resembled $f_{WaterLevel2}$ and is seen in the last subplot of Figure 5.3.

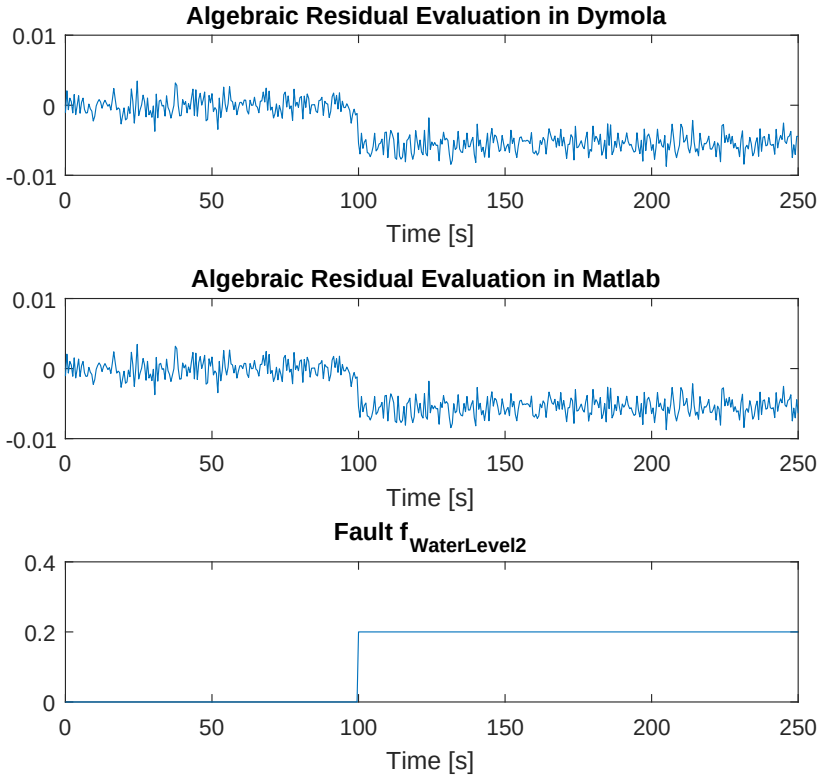

Figure 5.3: *Evaluation of an Algebraic Residual in Matlab and Dymola*

There are more than just sequential residuals implemented in this thesis. For instance, the original FDT observer method `ObserverResGen` is compared to the method described in Section 5.1.2. The equations that are used for the observer generation are equation (3.18),(3.19),(3.21),(3.22),(3.25) and (3.26), along with data from a simulation of the model of the system. The computation of F using the methods developed in the FDT resulted in $F_i = 1$ for the state-estimation that was based on equation (3.19). The value of F was a prerequisite to start the observer generation in Dymola. The results of the Dymola implementation from Section 5.1.2 with $W_r = 0.05$, is seen in the top subplot of Figure 5.4. The result of the original FDT method is seen in the middle subplot, while the last subplot shows the result of the original FDT method, but when using the initial state and gain from the Dymola observer. These two observers are hereby referred to as the *old FDT observer* and the *new FDT observer*.

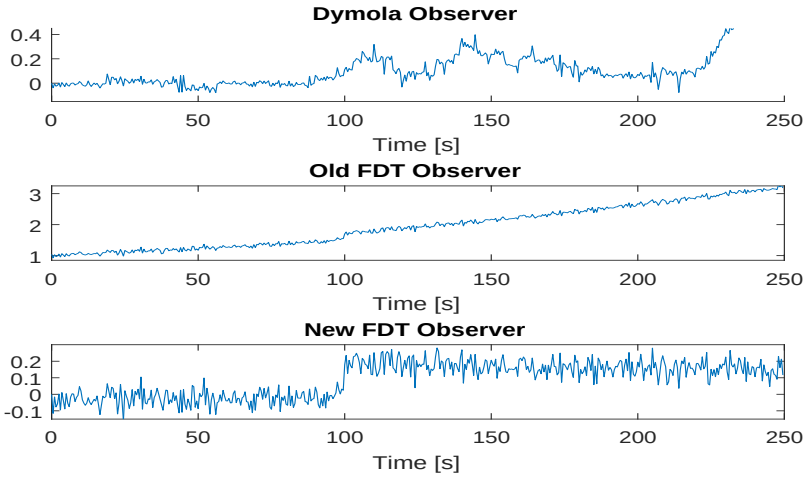

Figure 5.4: *Result of Observer residuals*

Summarized Conclusion

With the achieved results from the residuals implemented in Dymola and Matlab, the comparison of the two residual implementations can begin, which enables investigation of Research Question 5. Please notice that this comparison does not state whether Matlab or Dymola is a better implementation language in general, but rather what benefits and drawbacks the current methods in the FDT has compared to the new Dymola methods.

The residuals created by the `seqresgen` method in the FDT clearly results in poor estimates when the derivatives have a high rate of change, which corresponds to the system being in an excited state. This is seen in the first 20 seconds of Figure 5.2, which coincides with the period where the slope of u deviates from zero, see Figure 5.1. It is therefore reasonable to assume that the poor result in Matlab originates from the faulty assumption of the derivatives being constant between each state estimation, which follows from the Euler forward method that is used in the FDT, see [6]. This in combination with use of a constant step-length of 0.5 results in the poor accuracy. The corresponding Dymola result only has a measurable deviation from zero close to the 20 seconds mark, which corresponds to the point where the ramp input suddenly decelerates to a flat line. This deviation is however corrected quickly compared to the Matlab version and this indicates that the backward differentiation formulae method (DASSL) that is used in Dymola is more suited for non-algebraic residual-evaluation if the monitored system is in an excited state. The DASSL-solver also has a variable step-length, which contributes to the higher precision of the Dymola method.

The result of the algebraic residuals from Figure 5.3 however, indicates a very

small difference in performance. This is not unreasonable as differences that follows from machine epsilon (see Table 4.1) or the Dymola tolerance of 0.0001 is negligible compared to the effect of the noise in the measurements. One clear benefit of the Matlab version though, is that the `seqresgen` method in the FDT produces residuals that are evaluated sample by sample, which means that the method is more suited for real-time applications. Atleast when compared to the current Dymola implementation, that requires all the measurements to be present in tables, see Figure 4.3. On the other hand, given that the original model was implemented in Dymola, having the diagnostic system in Dymola as well enables simulation of the entire system; both original model of the system that is monitored and the diagnostic system. This is a clear benefit as it widens the usage of the FDT towards users that rely on Dymola models, which includes Saab, the client of this project.

The difference between the residual evaluation for the observers in Figure 5.4 can be partly explained by the fact that the initial values are only recommended values in Dymola [5], while they are final in the Matlab evaluation. This means that a poorly chosen initial value is easier to correct with sensor-data in Dymola compared to Matlab.

Apart from the interpretation of initial values it is harder to link the differences to a special cause, given that the observers are computed in such different ways. The Dymola observer has the benefit of using the available measurements for computing the optimal observer gain with respect to the chosen objective function, see equation (5.34). As the measurements are used for the gain estimate, there is no need for a predetermined operating point, which widens the usage of the method. Additionally, the Dymola observer is not limited by index of the used MSO, which is a great improvement over the old observer method. The sharpest within a decent time limit was seen in the *new FDT observer* though, even if the Dymola Observer reacted even more, but with over 100 seconds after the fault was implemented. This indicates that the Dymola observer is more suited for applications where a low risk of false alarms is needed, and timing is less crucial.

It is clear that the FDT benefits from the Dymola method as the performance of the observer drastically improves when the Dymola estimates of the initial state is used, as well as an observer gain that is generated using Lannerhed's Theorem and PMP instead of just a random one, see the last subplot of Figure 5.4.

Finally, the optimization done in the Dymola method can easily be compared to an LQ-regulator, as the integrand of the objective function $(F(t)\dot{r} + G(t)r)^2 + W_r r^2$ contains a combination of punishment on effort, which corresponds to the first square, and state, as r can be expressed as a state according to the proof of Lannerhed's Theorem. This partly explains the benefits of the Dymola observer compared to the *old FDT observer* seen in Figure 5.4. This concludes the investigation of Research Question 5.

5.4 SECS Model

The achieved results based on the SECS Model of low fidelity is presented in this section. The results consists of theoretical results such as the investigation of isolability (see Section 5.4.1) and complexity (see Section 5.4.2), but also an practical implementation of a diagnostic system is included (see Section 5.4.3).

5.4.1 Isolability analysis

The application of the `IsolabilityAnalysis` method from the FDT provided the following isolability for the SECS model, given the pretext described in Section 3.6. For clarification, the F_x notation is used in this section to describe system modes that contains the fault f_x.

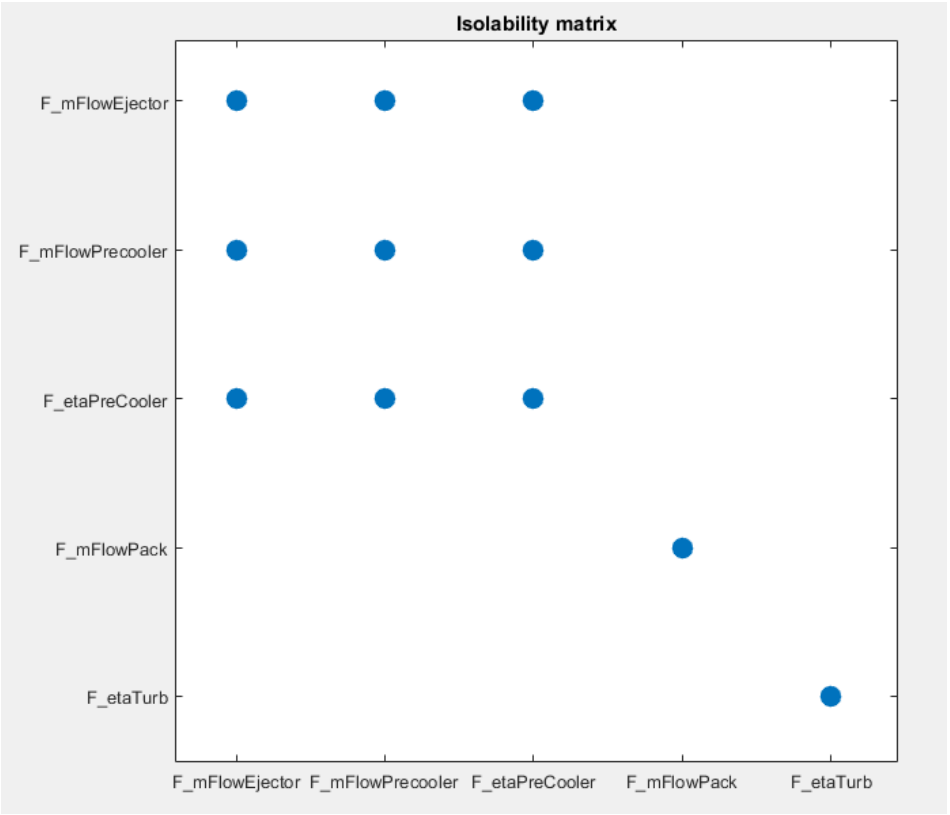

Figure 5.5: *The Isolability potential of a Diagnostic system, given the sensors provided in SECS and no causality restrictions*

From Figure 5.5 it is clear that the faults $F\_mFlowEjector$, $F\_mFlowPrecooler$ and $F\_etaPreCooler$ could not be isolated from each other with the current setup of sensors. The faults $F\_mFlowPack$ and $F\_etaTurb$ were however fully isolable

from the rest of the faults. This result appeared to remain consistent despite causality-restrictions, which was concluded by using the same method with causality inputs. This result is seen in Figure 5.6.

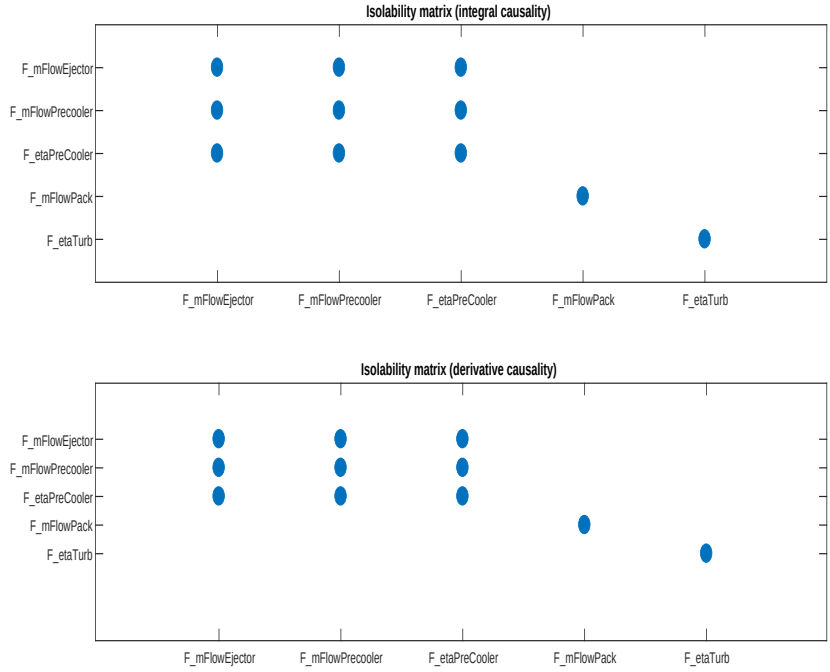

Figure 5.6: *The Isolability potential of a Diagnostic system, given the sensors provided in SECS and causality restrictions*

By applying the method `SensorPlacementIsolability` of the FDT on the model it can be established that there are 16 minimal combinations of additional sensors that would result in full isolablility for the modelled faults. Also, each of these combination consisted of 2 sensors. One of these combinations consisted of sensors measuring the massflows through the Ejector and the Precooler. The resulting isolability analysis after these sensors were added is seen in Figure 5.7. It is worth mentioning that all these conclusions and results regarding isolability are based on the SECS low fidelity model. Therefore it cannot be guaranteed that the same results are applicable to the real system, or even a model of higher fidelity. With this said, the investigation of Research Question 6 is concluded.

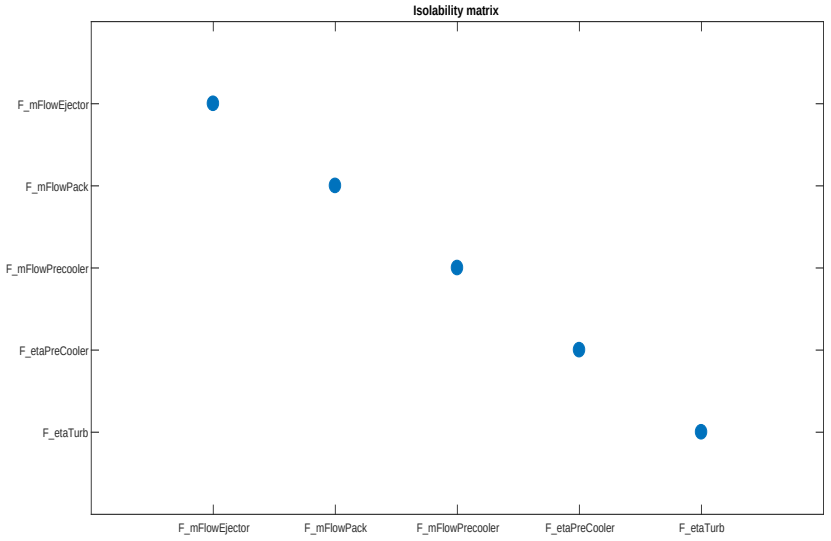

Figure 5.7: *The Isolability potential of a Diagnostic system, after adding additional sensors to the sensors provided in SECS and no causality restrictions*

5.4.2 MSO computation and Complexity

The isolability analysis only provides the theoretical limits of the generated diagnostic system. In order to achieve practical results, residuals must be generated and evaluated. In this thesis the residual generation was based on MSOs. When computing MSOs, the method in Section 4.1.1 was used. The method was applied for sub-models of SECS with degree of redundancy 7 to 10 and the time was measured for each application. As earlier studies in the field indicated that most MSO-generating algorithms was of exponential complexity [11] , the following assumption was made;

$$T \approx C_1 e^{C_2 n} \tag{5.47}$$

where T is the execution time, n is the degree of redundancy and C_1,C_2 are unknown coefficients. Based on the measurements it could be stated that;

$$C_1 \approx 1.61 \cdot 10^{-4} \tag{5.48}$$

$$C_2 \approx 1.52 \tag{5.49}$$

resulted in the best fit in the least-squares sense to the acquired measurements. The measurements and the resulting fitted curve from Equation 5.47 is seen in Figure 5.8.

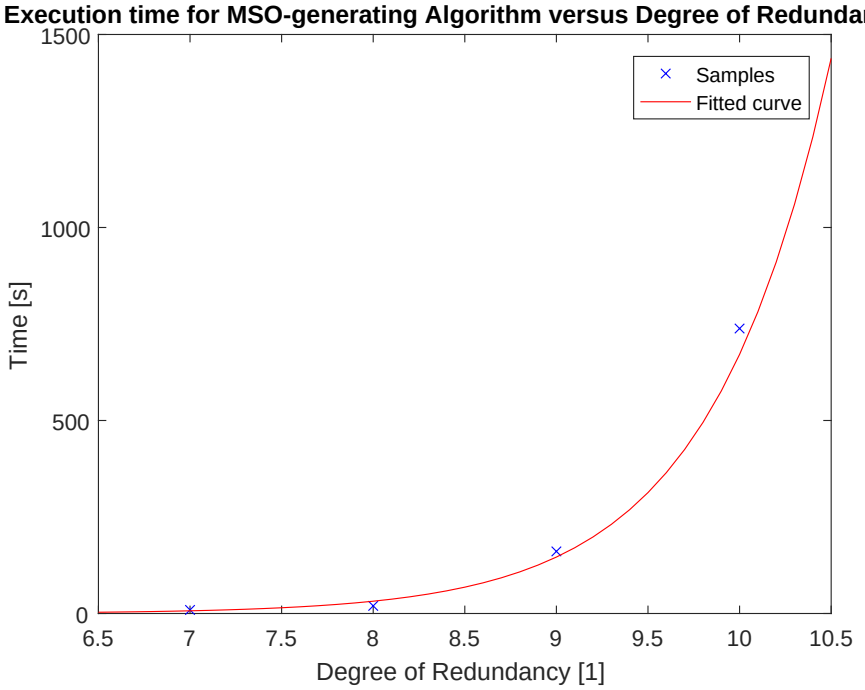

Figure 5.8: *Resulting Execution time versus Degree of Redundancy*

Summarized Conclusion

The result seen in Figure 5.8 indicates that the MSO-generating method of this thesis is of exponential complexity. The resulting coefficient C_2 also supports this, as the following Maclaurin-series is achieved for Equation 5.47;

$$C_1 e^{C_2 n} = C_1 \sum_{k=0}^{\infty} \frac{(C_2 n)^k}{k!} \tag{5.50}$$

from which it is clear that $C_2 > 1$ results in slower decay for the kth-term for large k compared to $C_2 < 1$. The result of $C_2 = 1,52$ therefore supports the hypothesis that the complexity is exponential or at least that it is not probable for a polynomial of limited order to provide as accurate description of the complexity. The time-complexity of the algorithm is therefore considered being exponential and this concludes the investigation of Research Question 2.

5.4.3 Diagnostic System Generation

While the MSO-algorithm from Section 4.1.1 was successful in generating MSOs for the SECS, most of the MSOs could not be used for sequential residual generation in the FDT. One of these sets, the *sensitive MSO*, was investigated as it had

sensitivity for both $f_{mFlowPack}$ and $f_{etaTurb}$. Given that the *sensitive MSO* had 10 if-statements, atleast 9 if-statements had to be part of the equation system to be solved, which resulted in a problem that could not be handled by the current FDT implementation. This set of equations is hereby refereed to as the *sensitive MSO*.

However, by using the method described in Section 4.2 and convert the *sensitive MSO* back to Dymola, the resulting equation system could be solved. The MSO-generation algoritm also found a couple of MSOs that had no sensitivity to any of the modelled faults. One of these, the *non-sensitive MSO*, was also chosen for residual evaluation. It is worth mentioning that the *non-sensitive MSO* had equations in common with the *sensitive MSO*. The result of these residuals based on a $\boldsymbol{Z}$ provided from a noise-free simulation of the system in faultfree mode is seen in Figure 5.9.

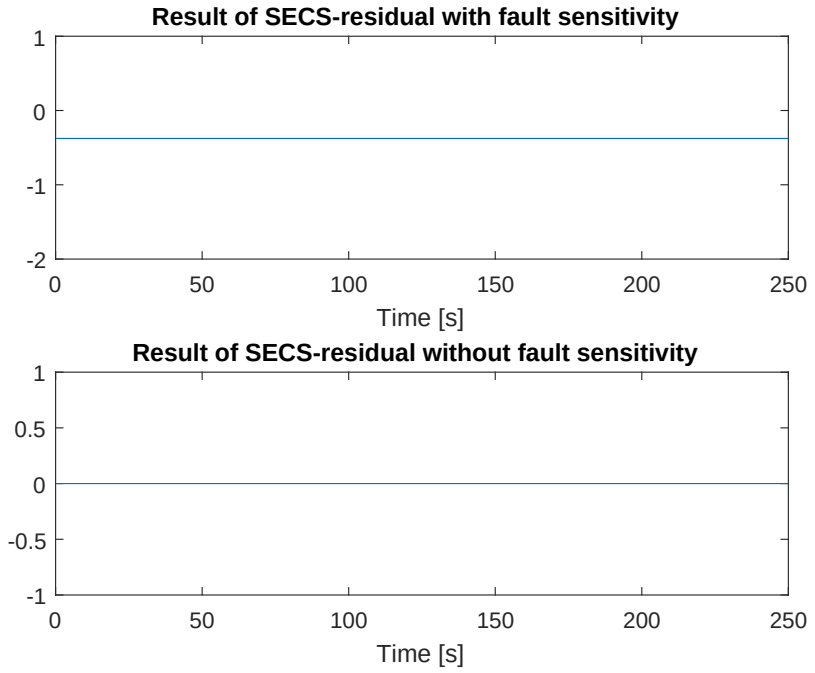

Figure 5.9: *Residual result of two Algebraic residuals based on SECS with no faults present*

When the fault $F\_mFlowPack$ was present, the *sensitive MSO* resulted in an unsolvable system, at a relative fault at 5 percent. The same effect was achieved when a non-monitored fault (F_nonmon) in the form of a biased temperature sensor was implemented. The result of the residual based on the *non-sensitive MSO* is seen in Figure 5.10.

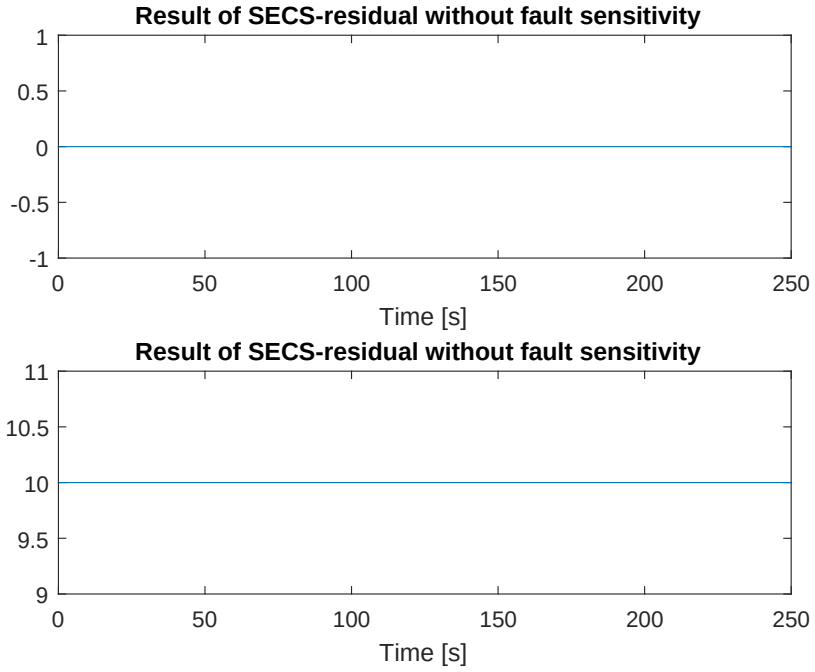

Figure 5.10: *Non-sensitve Residual response to F_mFlowPack and F_nonmon*

The resulting reactions of the residuals from the *sensitive MSO* and the *non-sensitive MSO* are summarized in Table 5.1 for all the tested system modes.

	sensitive MSO	*non-sensitive MSO*
NF	solvable	no reaction
F_mFlowPack	unsolvable	no reaction
F_nonmon	unsolvable	reaction

Table 5.1: *Resulting observations of residuals under different system conditions*

Summarized Conclusion

The observations of the chosen MSOs all differ from each other when the different system modes are tested, see Table 5.1. From Definition 1 it follows that the tested modes are isolable from one another. It can therefore be concluded that false alarms (See Research Question 7) can be avoided to some degree by utilizing non-sensitive MSOs similar to the *non-sensitive MSO* used in this thesis. The

point regarding the *non-sensitive MSO* having equations in common with the *sensitive MSO* is of great importance however, as the *non-sensitive MSO* then provides model-validation of some of the equations used in the generated residual. This is achieved by checking consistency in its sets of equations, which includes some equations from the *sensitive MSO*. By using both of these types of sets in the method, both diagnostics and model-validation can occur in parallel which is of great interest as it reduces computation time compared to approaching the fields in series and, as shown here, enables reduction of the risk of false alarms.

The observant reader might have reacted to the fact that the residual based on the *sensitive MSO* is not exactly equal to zero in Figure 5.9. This is due to the fact that the unit-information is lost in the translation process. In Matlab for example, pressure-variables are of the type *double* while in Dymola the units for pressure includes Bar, kPa, Pa and so on. This in combination with some of the parameters being hard coded results in trouble during the reconstruction of the model. For example; the pressure constant 95 results in very different model behaviour if it is interpreted as 95 $Pascal$ compared to 95 Bar. Therefore, given that there is no support for units in the current version of the FDT, great precaution is necessary when interpreting results from the residuals generated in Dymola. This conflict is minor however, given that the current FDT implementation could not generate any residual for the same set of equations, regardless if a fault was present or not. This contributes to the investigation of Research Question 5.

The Dymola implementation could solve the equation system provided by the *sensitive MSO* given that the measurements were taken from a simulation of the system without any implemented faults. The Matlab solver failed to generate a solution to the same equation system. Therefore, the results also contributes to the investigation of Research Question 5.

6

Conclusions and Future Works

This section provides the conclusions (see Section 6.1) of the thesis as well as suggestions on improvements that could be included in future applications. The topics of Model Translation (see Section 6.2) and Residual Generation (see Section 6.3) are treated.

6.1 Conclusions

This thesis describes computation of diagnostic systems based on models implemented in Dymola. Dymola is a program that uses the language Modelica. The Dymola models are translated to Matlab, where an application called Fault Diagnosis Toolbox, FDT is applied. The FDT has functionality for pinpointing minimal overdetermined sets of equations, MSOs. The algorithm for generating MSOs that was implemented in this thesis has exponential time complexity with regards to what level the system is overdetermined, also known as the degree of redundancy. The thesis also included tests of these MSOs through generation of residuals, which are functions that are equal to zero given that the system is fault-free. Residual generation in Dymola was added to the original methods of the FDT and during the thesis the results of the Dymola methods were compared to the original FDT methods. It could thereby be concluded that adding the Dymola methods to the FDT resulted in higher accuracy, as well as a new way to compute observer gain.

By applying the FDT to Saabs model of the Secondary Environmental Control System, SECS and the simpler Two Tank System, it could be validated that the computational properties of the developed methods in Dymola and Matlab differs and that it therefore exists benefits of adding the Dymola implementations to the current FDT methods. Furthermore, the investigation of the potential isola-

bility based on the current setup of sensors in SECS showed that full isolability is achievable by adding 2 mass flow sensors, and that the isolability was not limited by causality constrictions. One of the found MSOs was solvable in Dymola when given data from a faultfree simulation. However, when faults were implemented into the simulation, the resulting equation system of the residual became singular. By utilizing MSOs that had no reaction to any modelled faults, certain non-monitored faults could be isolated from the monitored ones and thereby reduce the risk of false alarms.

Some residuals were generated as observers, and a new method for constructing observers was found during the thesis by using the deduced Lannerheds theorem in combination with Pontryagin's Minimum Priniple. This method enabled evaluation of observer based residuals in Dymola without any selection of a specific operating point, as well as evaluation of observers based on high-index DAEs. The method also resulted in completely different behaviour of the estimation error compared to the method that was already implemented in the FDT. For example, one of the new observer-implementations achieved an estimation error that had much faster convergence towards zero when no faults were implemented in the monitored system, as well as having a sharper reaction to implemented faults.

6.2 Model Translation

In order to apply diagnosis to a general Dymola model, the parser from Section 3.3 must be developed. A reasonable first step on this topic is to achieve a translation of a general models structure. This would enable utilization of FDTs structural methods without any demands for model-specific functions being defined in Matlab. As seen in the thesis however, there are downsides of using structural methods, especially when these are used in combination with selection methods such as if, see Section 5.2. Therefore, development of the FDT is necessary in order to achieve robustness. This development should include excluding certain matchings, that is, the ability to perform structural methods, but still taking whether the variable is expressible from the current equation into account. If a variable is not expressible, it limits the usage of the equation, given that another matching must be found for this specific variable.

6.3 Residual Generation

The time-complexity of the MSO-generating algorithm is exponential with respect to the degree of redundancy. This means that methods which reduce the degree of redundancy, similar to the one implemented in this thesis are necessary in order to keep the time-complexity under control.

It has also been shown in this thesis that overdetermined systems without sensitivity to physical faults can be used for model-validation. In this thesis, a special case was found for the specific sensitive MSO (see Section 5.4.3) but this must be

generalized if it is to be applied on a general example. The minimal property of the MSO is not beneficial in this case, given that the goal of the over-determined system is not to diagnose, but rather validate as many equations as possible, while still remaining insensitive to the possible physical faults. Non-minimal PSO sets can also be of interest if they have fault sensitivity. It could be interesting to investigate whether the replacement of fault-signals with residual variables would result in sharper reactions (that is, r deviating more from zero) when in presence of those specific faults.

This thesis also provided a new method for observer generation, which could be improved in future projects. The method currently focuses a lot on stability, but a quick decay of estimation errors is typically also wanted when constructing residuals. Therefore, the optimal control problem used to generate G should be formulated in such a way, that it takes this desired property into account, preferably while maintaining the convexity of the current problem. The implementation in Section 5.1.2 has a higher weight on the decay of the estimation error, but cannot be guaranteed to result in a convex optimal control problem. The optimal control problem could also optimize with respect to the coefficient F, given that Theorem 1 only specifies the dimension and not specific values of the elements in F.

The implementation of residuals in Dymola does not only provide new ways of constructing observers, but it also opens a couple of possibilities for implementation of real-time diagnostic systems, as the C-code generated in Dymola can be applied to an integrated circuit. This enables a wide range of usage of the systems generated by the FDT and it is there an interesting topic for future projects.

Appendix

A

Parsing of Modelica Models

There are advantages of extracting models from Dymola to other platforms, such as Matlab. Instead of manually translating a model from Modelica to Matlab, which depending on the size of the model might be a tremendous task, it could be translated automatically. This thesis will explore one option on how the automatic translation can be achieved.

A.1 XML to Matlab Parser

In order to extract and translate the information contained in the ModelicaXML file software had to be developed. Therefore a program called XML to Matlab Parser (XMP) was implemented. The choice of programming language fell upon Python, which is suitable for a number of reasons. Python is foremost easy to use and a widely used language. The availability of the open-source XML-parser `xml.etree.ElementTree` (ET)[1] contributed as well, since it simplified the coding significantly. The XMP uses the ET to extract the tree structure as a Python object, on which operations is performed, to extract the information contained within each element.

A.1.1 Limitations

In the scope of this thesis only models which are translatable to valid ModelicaXML-code were considered. ModelicaXML-formatting is not standardised within the Modelica community and different programs built upon the Modelica language will have different standards. Therefore the structure will differ between different software applications of Modelica and for some might not be accessible at all. That is, only models translated by Dymola were taken into consideration. Since the time was limited not all cases that might occur in the Dymola-specific Mod-

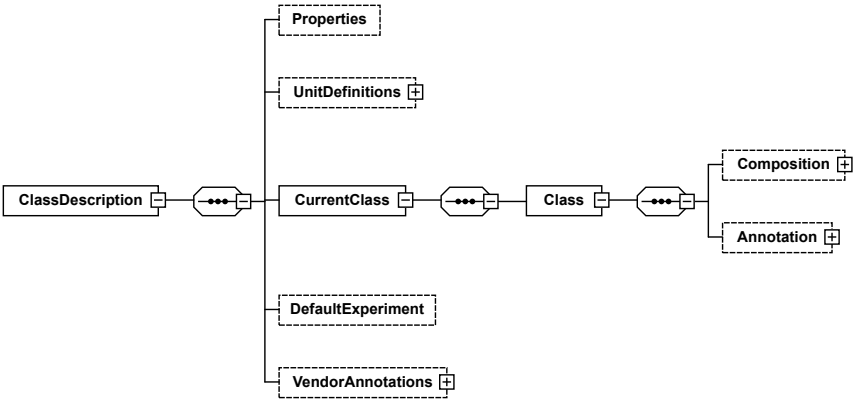

Figure A.1: *The root element of ModelicaXML.*

elicaXML standard were taken into consideration. Instead the XMP was based on the SECS-model as an initial guideline and further elaborated with a model of the cabin control pressure in Saab 39 Gripen. The limitations of the XMP will be mentioned throughout this chapter.

A.1.2 Structure and Content of Dymola-based ModelicaXML

The ET-parser creates a Python object called `ElementTree`. The tree consists of elements and can be visualised, as in Figure A.1, utilising the XSD file described in Section 4.4.1 in [15]. An element is depicted as a rectangle containing its tag. An optional element, i.e an element not necessarily needed to fulfil the ModelicaXML-standard, is visualised as a rectangle with a dashed line. A plus sign inside a square attached to the element means that the tree can be expanded further, i.e that the element can have child-elements. Throughout this chapter the element tag is written in **bold** and optional text strings and attributes shown in *italic*.

In Figure A.1 the root element, called **ClassDescription**, is depicted. Most of the information extracted by the XMP is contained in **Composition** which along with its children is depicted in Figure A.2. **Declaration** contains all symbols and components declared in the current component, i.e the **Class** element currently regarded, as shown in the bottom part of Figure A.3. In each of these sub-components the structure of the **Class** element is repeated, recreating the same nested structure as Modelica. As for the equations declared in the current component they are found in the element **Body**.

When defining models in the FDT the tree-structure is completely flattened, but the component hierarchy is kept in the names of variables to ensure unique variable-names and to enable the conversion from Matlab back to Modelica. This is achieved by always keeping track of which components have been entered and storing their names in a list, which is flattened before printing.

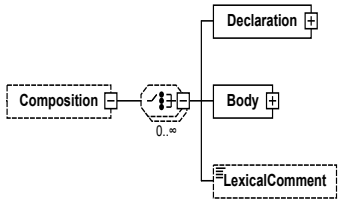

Figure A.2: *The element* ***Composition*** *with its two child elements.*

A.1.3 Symbols

In ModelicaXML variables, parameters and constants are all declared with **Variable** as seen in Figure A.3. What Dymola regards as variables will be referred to as symbols to avoid confusion. Parameters, constants, continuous and discrete variables are the type of symbols that may appear in a ModelicaXML file.

Variables, Parameters and Constants

To distinguish between parameters and variables, the variable key *variability* is used. This key is not present if the variable is continuous. For parameters the key equals *parameter*, for discrete variables *discrete* and for constants *constant*. However the latter is ignored since constants are inserted into the equations before the translation to ModelicaXML takes place. If the symbol is a parameter it will have an assigned expression which is found in **BindingEquation** and/or **StartExpression**. If the symbol is non-scalar the dimension is to be found in **Dimension**. Variables, independent of their causality are considered unknown and will be added to the set of unknown variables $\boldsymbol{X}$, unless they already are present in the set of known variables $\boldsymbol{Z}$.

Known Variables

In the field of fault diagnosis it is important to know, or to decide, which variables are considered known. Typically these are sensor measurements or states which are potentially measurable, and inputs. Which the known variables are has to be determined manually, but is often a straight forward task. These are contained in the set of known variables $\boldsymbol{Z}$.

Fault Variables

Fault variables need to be distinguished from known and unknown variables. If faults are implemented in the Modelica model, this can be achieved by using a name convention. If no name convention is used, or if it is inconsistently used, the faults can be sorted out from the rest manually afterwards and placed in the set of fault variables $\boldsymbol{F}$. If no faults are implemented in Modelica or if the user chooses to add more faults this has to be done manually.

The FDT requires that a fault variable only occurs in one equation, this based on the assumptions made in [8, page 5]. If faults are added manually, after the

parsing step, the user has to take this into consideration. But if fault variables are introduced directly into the model in Modelica, the XMP will handle this by considering the original fault f_{org} as an unknown variable. To avoid information loss a new equation $f_{org} = f_{new}$ is added, where f_{new} will be added to the set of fault variables instead.

A.1.4 Equations

Equations are contained in the element **Body**, containing the attribute *bodykind*, which states if the children are of type *equation*, *initial* or *algorithm*. In Modelica these are declared under the equation, initial equation or algorithm section respectively. In the FDT no differentiation is made between equations and algorithms and therefore both will be referred to as equations. In the case of initial equations these are valid only at the initialisation stage, and do not hold true at other time instances. Therefore the initial equations are disregarded, otherwise they might add false information. The element **Body** can contain different types of equations, which are displayed in Figure A.4. All body types are implemented except `While`. However the type **MultiReturningFunction** has been deemed irrelevant, for the current application of the XMP and is disregarded. The reason is that they do not contribute with any structural information. In the models regarded only `assert` statements are contained in **MultiReturningFunction**s, which are used to verify that the specified conditions are met at the simulation stage. These are typically used to ensure that a model operates within its limits of validity, e.g. that an absolute pressure never assumes negative values. Common equations are contained in a **SimpleEquation** element and consist of a left hand and right hand side marked with **LHS** and **RHS** respectively, as seen in Figure A.4. Each side can contain any **ExpressionType** elaborated below in Section A.1.5. Although often the left hand side only contains one variable, i.e a **ComponentReference**.

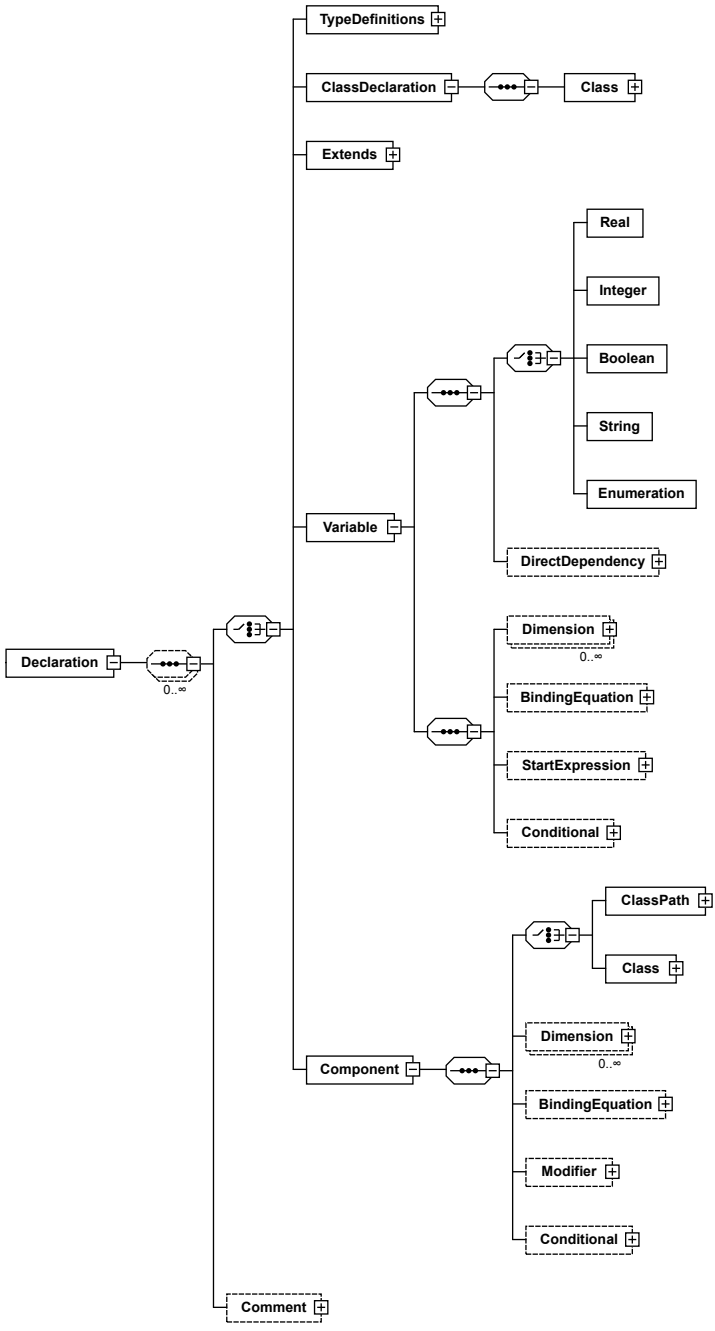

Figure A.3: *The element* ***Declaration*** *and its children with the child elements* ***Variable*** *and* ***Component*** *expanded.*

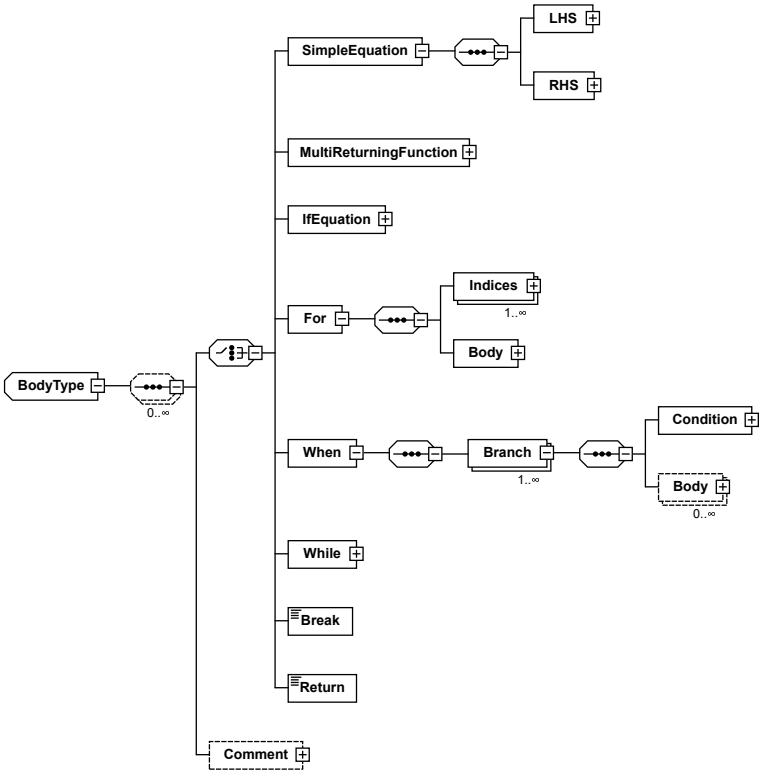

Figure A.4: *All Body types that can be present in ModelicaXML.*

A.1.5 Expressions

Expressions appear when parameters are assigned values and in equations. The expression types

- `Literal`
- `ComponentReference`
- `Unary`
- `Binary`
- `IfThenElse`
- `Range`
- `FunctionCall`
- `ForExpression`

are the types that can occur in ModelicaXML, which are all handled by the XMP.

Operators

An operator is referred to as **Binary** or **Unary**. The operands, or operand if it is a unary, are below the operator in hierarchy and can be any of the above mentioned expressions.

Function

Functions are tagged with **FunctionCall** and are handled differently depending on the function. The name of the function is to be found in the *ClassName* attribute of the child-element **FunctionPath**. The input arguments are contained in the element **Arguments** which has one or more children named **Argument** containing the expression.

Functions that are called in the same manner in Modelica as Matlab, e.g. trigonometric functions, are extracted as they are. Other functions, that do not appear in the same manner in Matlab, must be treated differently. Derivatives holds a special position amongst these since they are regarded as separate variables in the structural analysis. If a variable is differentiated in an equation the derivative is added to the list of variables and a new equation is added in the printing process. Other functions that do not have a Matlab equal are regarded as external functions and have to be defined in a separate m-file. An example is the `spliceFunction`, which performs a spline interpolation of two functions. This has to be done manually, but the functions found in the standard library are often well-documented within Dymola. Some Modelica-functions, for example `smooth` or `noEvent` do not describe the model but are used to facilitate the modelling process and are disregarded by the XMP.

If- and When-Statements

Besides functions if- and when-statements are also defined by external functions. If-statements are tagged with **IfThenElse**. They consist of a condition, a then-branch and an else-branch. If the condition is true, the if-branch is executed, otherwise the else-branch is executed. If-statements are predefined by a shell function in Matlab which takes the condition and the then- and else-branches as arguments. Note that the **ExpressionType** described here should not be confused with the **ifEquation** from Figure A.4. Although they serve more or less the same purpose they are expressed differently in ModelicaXML.

When-statements have the tag **When** and are of the type **Body**. They are therefore extracted in another fashion than if-statements, but are implemented in Matlab in a similar manner and will therefore be mentioned in this section. Each when-statement is declared with its own condition and only one branch which is executed if the conditions holds. Currently only when-statements where a single variable is assigned inside the statement is supported. The when-statements are bundled together as ifelse-statements without an else-statement in Matlab. The condition of each when-statement constitute the condition of a ifelse-statement except the first which is translated to the if-statement in the ifelse block. The branches inside the when-statements are translated to the branches of the ifelse-statement of the respective conditions.

References to Symbols

Symbols are declared inside **ComponentReference** which in turn contains one or more **Reference** instances depending on if the symbol is declared locally or outside the current component. As mentioned before the hierarchy is kept track of and is compared to the references to avoid printing component names twice or printing a false hierarchy in the final symbol string. The latter can occur if a symbol is referenced to inside a component but declared outside of it. Components, which are referenced to deeper down in the hierarchy, can be defined as inner/outer. They are declared both at the higher level as `inner`, and at the lower level as `outer`. An example for when the component typically is declared as inner/outer is a component that contributes with information of the environmental temperature and pressure of a system, or a component describing physical fields, e.g. an electrical field. If this is the case the symbols contained in these types of components will have different hierarchical structures depending on where they are declared, but they all refer to the same component. Such types of components do not have an attribute labelling them as inner/outer and the user therefore has to specify if such components are present to achieve the correct hierarchy.

A.2 Conclusion and Discussion

One of the main questions this thesis aims to answer is how Modelica models are portrayed in ModelicaXML. In this chapter the structure and content of ModelicaXML is presented. Implementation-wise the extraction has been solved mainly

with for-loops iterating over the tree-structure. Since the structure is repeated within components, recursive methods are used as well. Regarding the question on how Modelica should be translated to Matlab, the required syntax of the FDT is used as a base. Most equations in Modelica can be translated as is, since the languages resemble each other, but some paraphrases where made to fit the FDT, e.g. are derivatives handled differently.

Overall the XMP covers much of the Modelica language, but is not complete. To translate some of the remaining features, for example while-loops and universal when-statements, primarily more time is required. Some other parts of Modelica may offer greater challenges, e.g. translating all external functions, that can appear in Modelica, into Matlab. Depending on the desired level of autonomy, the implementation increases or decreases in complexity. Some limits of automation have already been detected, as the user for example has to define known variables, and inner/outer components. It is likely that more cases may appear. Furthermore, some solutions might work for the models regarded but perhaps not for other models, which were not explored. Additionally, the Modelica Association, responsible for setting standards in the Modelica community, needs to standardise the ModelicaXML-format, in order for further development to be justified. Still, the work presented in this thesis shows that it is indeed possible to extract and translate Modelica models into other languages.

Bibliography

[1] Elementtree overview. URL `http://effbot.org/zone/element-index.htm`. Cited on page 69.

[2] R.S. Bucy and P.D. Joseph. *Filtering for Stochastic Processes with Applications to Guidance*. John Wiley & Sons, first edition, 1968. Cited on page 13.

[3] E. Frisk C. Svärd, M. Nyberg. Realizability constrained selection of residual generators for fault diagnosis with an automotive engine application. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 43(6):1354–1369, 2013. Cited on page 18.

[4] M. Nyberg. C. Svärd. Residual generators for fault diagnosis using computation sequences with mixed causality applied to automotive systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 40(6):1310–1328, 2010. Cited on page 17.

[5] *Dynamic Modeling Laboratory User Manual*. Dymola, 2016. Cited on pages 22, 38, 41, and 55.

[6] D. Jung. E. Frisk, M. Krysander. *Fault Diagnosis Toolbox user manual -v0.12*, 0.12 edition, 2016. Cited on pages 2, 17, 30, 31, 34, 35, 37, 38, 40, and 54.

[7] M. Krysander. E. Frisk. Sensor placement for fault diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 38 (6):1398–1410, 2008. Cited on page 18.

[8] M. Krysander E. Frisk, A. Bregon et al. Diagnosability analysis considering causal interpretations for differential constraints. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 42(5):1216–1229, 2012. Cited on pages 9, 10, 11, 12, 17, and 71.

[9] L.S Pontryagin et al. *The Mathematical Theory of Optimal Processes*. Interscience, fourth edition, 1962. Cited on pages 16, 37, and 46.

[10] F. Gustafsson. *Statistical Sensor Fusion*. Studentlitteratur, second edition, 2015. Cited on pages 13 and 18.

[11] M. Krysander et al J. Armengol et al. Minimal structurally overdetermined sets for residual generation: A comparison of alternative approaches. *7th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, 42(8):1480–1485, 2009. Cited on pages 17 and 58.

[12] E. Frisk. J. Åslund. Observers for non-linear differential-algebraic systems. *Automatica*, 42(6):959–965, 2006. Cited on pages 2, 14, 15, 17, 37, 40, 43, and 44.

[13] M. Krysander. *Design and Analysis of Diagnosis Systems Using Structural Methods*. PhD thesis, Linköping univesity, 2006. Cited on pages 9 and 17.

[14] T. Glad. L. Ljung. *Modellbygge och simulering*. Studentlitteratur AB., second edition, 2011. Cited on pages 13, 17, 19, and 20.

[15] K. Lockowandt. Parsing and validation of modelica models utilising fault diagnosis. Master's thesis, Linköping univesity, 2017. Cited on pages 19, 22, 23, 25, 27, and 70.

[16] L. Mevel. M. Döhler. Fault isolation and quantification from gaussian residuals with application to structural damage quantification. *9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, 48 (21):640–645, 2015. Cited on page 18.

[17] J. Åslund. M. Krysander, M. Nyberg. An efficient algorithm for finding minimal overconstrained subsystems for model-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 38 (1):197–206, 2008. Cited on pages 18 and 34.

[18] E. Frisk M. Nyberg. *Model Based Diagnosis of Technical Processes*. M. Nyberg, E. Frisk, first edition, 2015. Cited on pages 2, 6, 7, 8, 11, 17, and 18.

[19] S. Park et al. Degree of fault isolability and active fault diagnosis for redundantly actuated vehicle system. *International Journal of Automotive Technology*, 17(6):1045–1053, 2016. Cited on page 18.

[20] R. Langari R. Sharifi. Isolability of faults in sensor fault diagnosis. *Mechanical systems and signal processing*, 25(7):2733–2744, 2011. Cited on page 18.

[21] G. Söderlind S. Mattson. Index reduction in differential algebraic equations using dummy derivatives. *SIAM Journal on Scientific Computing*, 14(3): 677–692, 1993. Cited on page 17.

[22] LR. Petzold. S. Raha. Constraint partitioning for structure in path-constrained dynamic optimization problems. *SIAM J. Scu. Comput.*, 22(6): 2051–2074, 2001. Cited on page 17.

[23] *Component Maintenance Manual Cabin Pressure Controller*. Saab Aeronotics, 2000. Cited on page 23.

[24] Viviam Lawrence Takase. Gripen E/F. Simulation Analysis Report for the Cabin Pressure Controller. report 12, Saab Aeronotics, April 2016. Cited on pages 23 and 24.

[25] Petter Ögren Ulf Jönsson, Claes Trygger. Optimal control - lecture notes. Cited on page 15.