Institutionen för systemteknik
Department of Electrical Engineering

Examensarbete

Design of Automated Generation of Residual Generators for Diagnosis of Dynamic Systems

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan vid Linköpings universitet
av

Isac Duhan

LiTH-ISY-EX--11/4440--SE

Linköping 2011

Linköpings universitet
TEKNISKA HÖGSKOLAN

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

Design of Automated Generation of Residual Generators for Diagnosis of Dynamic Systems

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan i Linköping
av

Isac Duhan

LiTH-ISY-EX--11/4440--SE

Handledare: **Daniel Eriksson**
ISY, Linköpings universitet

Examinator: **Mattias Krysander**
ISY, Linköpings universitet

Linköping, 26 October, 2011

	Avdelning, Institution Division, Department Division of Vehicular Systems Department of Electrical Engineering Linköpings universitet SE-581 83 Linköping, Sweden	**Datum** Date 2011-10-26

Språk Language ☐ Svenska/Swedish ☒ Engelska/English ☐ __________	**Rapporttyp** Report category ☐ Licentiatavhandling ☒ Examensarbete ☐ C-uppsats ☐ D-uppsats ☐ Övrig rapport ☐ __________	**ISBN** — **ISRN** LiTH-ISY-EX--11/4440--SE **Serietitel och serienummer** Title of series, numbering **ISSN** —

URL för elektronisk version

http://www.vehicular.isy.liu.se

http://www.ep.liu.se

Titel Title	Design av automatgenerering av residualgeneratorer för diagnos av dynamiska system Design of Automated Generation of Residual Generators for Diagnosis of Dynamic Systems
Författare Author	Isac Duhan

Sammanfattning
Abstract

Diagnosis and Supervision of technical systems is used to detect faults when they occur. To make a diagnosis, tests based on residuals can be used. Residuals are used to compare observations of the system with a model of the system, to detect inconsistencies.

There are often many different types of faults which affects the state of the system. These states are modeled as fault modes. The difference between fault modes are the presence of faults in the model. For each fault mode a different set of model equations is used to describe the behaviour of the real system. When doing fault diagnosis in real time it is good, and sometimes vital, to be able to change fault mode of the model, when a fault suddenly occurs in the real system. If multiple faults can occur the number of combinations of faults is often so big, even for relatively small systems, that residuals for all fault modes can not be prepared. To handle this problem, the residuals are to be generated when they are needed.

The main task in this thesis has been to investigate how residuals can be automatically generated, given a fault mode with a corresponding model. An algorithm has been developed and to verify the algorithm a model of a satellite power system, called ADAPT-Lite, has been used. The algorithm has been made in two versions. One is focusing on numerical calculations and the other is allowing algebraical calculations. A numerical algorithm is preferred in an automatized process because of generally shorter calculation times and the possibility to apply it to systems which can not be solved algebraically but the algebraical algorithm gives slightly more accurate results in some cases.

Nyckelord
Keywords Model Based Diagnosis, Dynamic Systems, Supervision

Abstract

Diagnosis and Supervision of technical systems is used to detect faults when they occur. To make a diagnosis, tests based on residuals can be used. Residuals are used to compare observations of the system with a model of the system, to detect inconsistencies.

There are often many different types of faults which affects the state of the system. These states are modeled as fault modes. The difference between fault modes are the presence of faults in the model. For each fault mode a different set of model equations is used to describe the behaviour of the real system. When doing fault diagnosis in real time it is good, and sometimes vital, to be able to change fault mode of the model, when a fault suddenly occurs in the real system. If multiple faults can occur the number of combinations of faults is often so big, even for relatively small systems, that residuals for all fault modes can not be prepared. To handle this problem, the residuals are to be generated when they are needed.

The main task in this thesis has been to investigate how residuals can be automatically generated, given a fault mode with a corresponding model. An algorithm has been developed and to verify the algorithm a model of a satellite power system, called ADAPT-Lite, has been used. The algorithm has been made in two versions. One is focusing on numerical calculations and the other is allowing algebraical calculations. A numerical algorithm is preferred in an automatized process because of generally shorter calculation times and the possibility to apply it to systems which can not be solved algebraically but the algebraical algorithm gives slightly more accurate results in some cases.

Sammanfattning

Diagnos och övervakning av tekniska system används för att upptäcka fel när de inträffar. För att ställa en diagnos kan tester baserade på residualer användas. Residualer används för att jämföra observationer av ett system med en model av system för att upptäcka inkonsistens.

Det finns ofta många typer av fel som påverkar ett systems tillstånd. Dessa tillstånd modelleras med olika felmoder. För varje felmod används olika uppsättningar av modellekvationer för att beskriva systemets beteende. När diagnoser ska ställas i realtid är det ofta bra och ibland avgörande att kunna byta felmod när ett fel plötsligt inträffar i systemet. Om multipelfel kan inträffa blir antalet kombinationer av fel ofta så stort att residualekvationerna för alla felmoder inte kan förberedas. Detta gäller även för relativt små system. För att hantera problemet bör residualerna kunna genereras vid den tidpunkt då de behövs.

Examensarbetets huvuduppgift handlar om att undersöka hur residualerna kan genereras automatiskt, givet en felmod och en modell. En algoritm har utvecklats och verifierats med en model av ett kraftsystem för en satellit, kallad ADAPT-Lite. Algoritmen har gjorts i två versioner. Den ena tillåts göra algebraiska beräkningar men den andra, i så stor utsträckning som möjligt, tillåts endast göra numeriska beräkningar. En numerisk algoritm föredras i en automatiserad process p.g.a. generellt sett kortare beräkningstid och dess egenskap att kunna lösa vissa problem som inte kan lösas algebraiskt. Den algebraiska algoritmen har dock visats sig ge aningen noggrannare resultat i många fall.

Acknowledgments

I would like to thank my supervisor, Ph.D. student Daniel Eriksson for the support and for all the help he has given me reviewing my report. I would like to thank my examinor, associate professor Mattias Krysander with whom I've had many technical and theoretical discussions. I would also like to thank associate professor Erik Frisk who has also given me much support with technical and theoretical questions.

Isac Duhan, Linköping October 2011

Contents

Chapter 1

Introduction

This work has been carried out at the division of Vehicular Systems, which is a part of the department of Electrical Engineering at Linköping University. The purpose has been to design and implement an algorithm to automatically generate tests, to detect possible faults in a system, based on theory of model based diagnosis.

In Section 1.1 there is an overview of model based diagnosis followed by an introductory explanation of automatic residual generation in Section 1.2. In Section 1.3 the ADAPT-Lite system, which will be used for the evaluation of the residual generator, is introduced. A problem description is formulated in Section 1.4. Finally an overview of the structure and content of the report is presented in Section 1.5.

1.1 Diagnosis and Supervision

Diagnosis and supervision of technical systems is in general about detecting and isolating faults occurring in the system. In many industrial systems it is important to have correct knowledge about the condition of the system and to be able to find and handle faults systematically. The reason for this can be many but often it is a safety and an economical issue, see e.g., [9]. When an industrial machine breaks down it can be dangerous for the operators working close to the machine. Using diagnosis to detect faults can prevent many hazardous situations. If left alone, a fault in a system can grow from minor to severe and cause additional faults. If a fault can be detected and corrected in an early stage, lots of money can be saved by preventing long production stops or more expensive repairs.

There are many different methods within the field of diagnosis. When the examined system can be described with mathematical models it may be possible to use model based diagnosis.

1.1.1 Model Based Diagnosis

In model based diagnosis a mathematical model, describing the observed system, is used to make a diagnosis. A diagnosis is, according to [9], a conclusion of which combinations of faults that can explain the process behavior. Faults can be discovered by detecting inconsistencies between the model and the real system. With an accurate model and well placed sensors it may sometimes be possible to not only say that a fault has occurred but also where in the system it is situated and what kind of fault it is.

One type of test to detect the inconsistencies between a model and a real system can be created by using a residual together with a criteria for when the residual show the inconsistencies. A residual can be written as

$$r(t) = f(y(t)) \tag{1.1}$$

where $r(t)$ is the residual, $y(t)$ are measurements and $f(y(t))$ is the residual generator, which is constructed out of model equations. When there are no model uncertainties in f and no measurement noise in $y(t)$, the following is true

$$r(t) = 0.$$

Example 1.1: Making residuals of a model

A model, where two sensors, $y_1(t)$ and $y_2(t)$, are measuring an unknown signal $x(t)$, is given as

$$\begin{aligned} y_1(t) &= x(t) \\ y_2(t) &= x(t). \end{aligned} \tag{1.2}$$

Using a variable substitution of $x(t)$ gives

$$y_1(t) - y_2(t) = 0 \tag{1.3}$$

which is a consistency relation. A residual generator $f(y(t))$ is made of the left hand side in (1.3) which is then written as

$$y_1(t) - y_2(t) = f(y(t)) \tag{1.4}$$

and the residual is the resulting signal when input is put into the residual generator as

$$r(t) = f(y(t)), \tag{1.5}$$

where $r(t)$ is the residual.

The model, which the residual generator f is made of, is seldom perfect and measured signals often contain noise. Therefore the residuals in most cases slightly deviate from zero even when the real system has no faults. Because of this the residuals are often filtered and thresholded. The thresholds are chosen to separate the deviation in the fault free case from the bigger deviation caused by faults in the system. In other words, the threshold makes the criteria for when there is an inconsistency between the model and the real system. An example of this can be seen in Figure 1.1, where a residual is affected by a fault after 50 seconds. Despite the signal being noisy, it is possible to detected the fault with a threshold, drawn with a dashed line in the figure. Understanding the use of thresholds is important for this thesis but no thresholds will be designed. The focus in this work is on generating residuals.

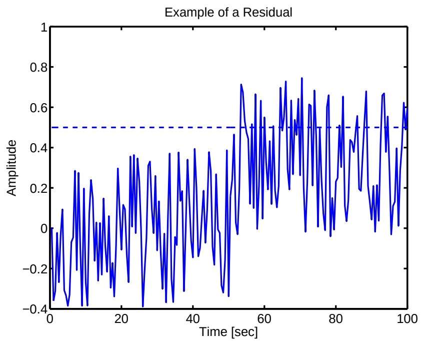


Figure 1.1. Example of a residual with a fault occurring at 50s. With a threshold at 0.5 on the y-axis the fault will be detected.

With information from several residuals it is often possible to make use of fault isolation algorithms which will say more precisely where in the system the faults have occured. Fault isolation is not in the scope of the thesis and will therefore not be further explained here but examples of fault isolation algorithms can be found in [3] and [8].

1.1.2 Fault Modes

To describe what state the system is in, in regard to what faults that are present in the system, the term fault mode is used, see [9]. The fault mode is used to tell which faults are present or if the system is free of faults. To be able to describe the behavior of the system when the system is in a certain state the equations which describe the corresponding fault mode are needed. Different fault modes are described by different sets of model equations. If one component in a system breaks it may however be that only the equations for that component changes in the set in the transition between the fault modes. The fault mode only says how the system is currently modelled, i.e., which equations are used. It does not necessarily tell the truth about what faults are actually present in the system.

With the nomenclature used in this thesis the fault free case is a fault mode. Only "component one" broken in the model is also a fault mode and only "component two" broken is another fault mode. A multiple fault of "component one" and "component two" broken in the model is yet another fault mode and so on.

Example 1.2: Fault Modes

The model equations

$$\begin{aligned} y(t) &= x(t) && [\text{ fault free mode }] \\ y(t) &= x(t) + k && [\text{ fault mode 1 }] \\ y(t) &= 0 && [\text{ fault mode 2 }] \end{aligned} \tag{1.6}$$

represent three different fault modes for a sensor y measureing a variable x. The first fault mode represents the fault free case, the second when there is a bias error present, and the third fault mode represents when the sensor is "dead". The three equations in (1.6) will never be used at the same time.

1.2 Automatic Generation of Residual Equations

Designing residuals for a diagnosis algorithm can be made by hand for small systems which have few measured signals. In bigger and more complex systems with a larger amount of measured signals, the number of possible residuals that can be made grows rapidly. To be able to get good fault isolation, that is to tell different faults apart when they are detected, more then a handful of the possible residuals might be needed for a large system. In this case it can be hard and time consuming to create all these by hand. Therefore to create residuals automatically would have several advantages. One is to save development time. Another is to avoid mistakes during development that are easy to make in a manual procedure, especially for large systems.

One important aspect is that even for relatively small systems, the number of possible residuals needed to detect and isolate faults in all possible fault modes is too large to be precalculated and prepared. Thus it is not only important but necessary to be able to automatically generate new residuals while the system is running and goes from one fault mode into another. This is especially true when there is a need to use different sets of model equations for different fault modes of the system. During simulation in this thesis it is however assumed that the fault mode is constant and does not do a transition into another fault mode. It is still however a background and motivation for generating the residual equations automatically.

1.3 ADAPT-Lite

During the past two years NASA has organized a diagnosis competition which in 2010 ran under the name 'Second International Diagnostic Competition (DXC'10), see [11]. In the competition there were three different entry categories. The competition category of interest for this thesis concerns the so called ADAPT-Lite system. What is given is the following:

Abbreviation	Component
BAT2	battery
INV2	inverter
FAN	fan
CB	circuit breaker
IT	current sensor
EY	relay
AC	load
ST, TE, ISH, ESH	position sensor

Table 1.1. The components and their abbreviations in the ADAPT-Lite system.

- A sketch of the system, shown in Figure 1.2, which shows how all the components are connected.
- All fault modes for each component.
- Measurement data from the sensors in the real system from a number of different fault scenarios.

What is not given are models for the components and model parameter values. The models have to be made and the parameters, e.g., resistance of resistors, have to be estimated with the use of the data series.

The ADAPT-Lite system is a suitable platform for evaluating a diagnosis algorithm and is therefore used for this in the thesis.

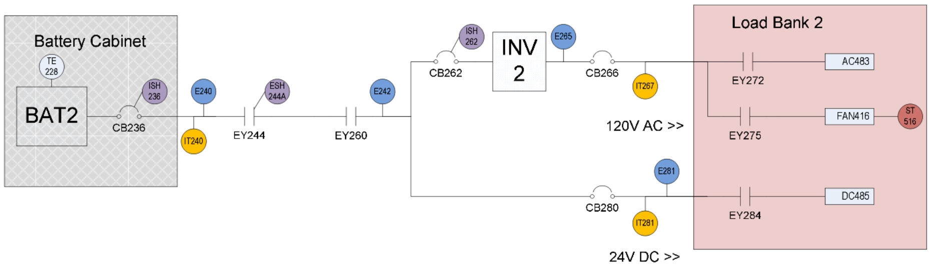


Figure 1.2. Schematic over the ADAPT-Lite satellite system. The figure comes from [11]. In Table 1.1 the abbreviations are explained.

1.4 Problem Formulation

The purpose of this thesis is to develop an algorithm which automatically generates tests from model equations to detect faults. This includes modeling of the system, developing an algorithm which automatically generates residuals and observer relations and creating a simulator to simulate the generated tests. The residuals are to be generated numerically. To evaluate the method it will be applied to the ADAPT-Lite system and the results from this will be analyzed.

The problem can be summed up in the following bullets:

- An algorithm to automatically generate residual generators from model equations shall be made.
- The algorithm shall handle equations that contain first order derivatives.
- If possible the algorithm shall operate completely numerical without assistance from algebraic solvers so that in the future an online implementation is made possible.
- The algorithm shall be evaluated on the ADAPT-Lite system.
- All the components in the ADAPT-Lite system that have not yet been modeled at the start of the thesis project need to be modeled.

The work does not need to consider the following:

- The algorithm does not have to work online in real time which also means that the simulation time can be long.
- To make or use a fault isolation algorithm is not in the scope of this thesis.

1.5 Thesis Outline

The thesis includes the following chapters:

Chapter 2, Theory The chapter presents a state space model and differential algebraic equations (DAE) and what a DAE's index is. It shows how numerical linearization can be made which is needed for the creation of a Kalman Filter observer. It is also discussed why a Kalman Filter is used when generating residuals and how it can be made.

Chapter 3, Modeling In this chapter it is presented how a bigger model can be built with smaller models of components. It is shown what equations are needed to be generated to complete the model and it also shows how unknown parameters can be estimated.

Chapter 4, Algorithm The chapter presents an algorithm that is used for generating the needed residuals. It shows how the model equations are restructured for the creation of the residual generators, how the Kalman Filter is made in the algorithm and how a simulation is done to estimate unmeasured signals in order to make residuals.

Chapter 5, Application on ADAPT-Lite In this chapter it is shown how the algorithm is applied to the system ADAPT-Lite for evaluation. The results of the simulations are shown in this chapter.

Chapter 6, Conclusions The chapter presents the conclusions from the evaluation of the application on the ADAPT-Lite system. It discusses the generation of the residual equations, the simulation method, algebraical shortcuts used for the numerical procedure, the estimation of the variances for the Kalman filter and approximations made in a linearization procedure.

Chapter 7, Future Work discusses future work that can be done that has not been addressed or completed in this master thesis. Topics addressed in this chapter are improved model, multiple faults, automatic parameter estimation, automatic estimation of λ, improved numerical simulation method and real time implementation.

Appendix A The appendix contain models of the ADAPT-Lite system.

Chapter 2

Theory

In this chapter the theory of mathematical models in the form of state space models and differential algebraic equations (DAE) are introducedv and an index of a DAE is described. This chapter also explains what a residual is and how observers can be created with a Kalman filter, then how to create a Kalman filter and how to do the required linearization is described. A technique for parameter estimation for model parameters is also described.

2.1 Models

There are two types of models which are used in this thesis, state space models, and models described by differential algebraic equations (DAE). These are presented in this section as well as theory for linearization of models.

2.1.1 State Space Models

A state space model

$$\begin{aligned} \dot{x} &= f(x, y) \\ r &= g(x, y), \end{aligned} \tag{2.1}$$

represents a dynamic system where $y = y(t)$ are the inputs, $r = r(t)$ are the outputs and $x = x(t)$ are internal states. A linear state space model is written as

$$\begin{aligned} \dot{x} &= Ax + By \\ r &= Cx + Dy, \end{aligned} \tag{2.2}$$

where the dimension of the matrices are

$$A \in \mathbb{R}^{n\times n},\ B \in \mathbb{R}^{n\times p},\ C \in \mathbb{R}^{q\times n},\ D \in \mathbb{R}^{q\times p}.$$

In this thesis, the non linear form in (2.1) is used in the simulation of the system and the linearized form in (2.2) is used when designing observers using a Kalman filter.

2.1.2 Differential Algebraic Equations

A system of differential algebraic equations (DAE), contains both dynamic and algebraic constraints, see [1]. A DAE is written as

$$g(\dot{x}, x, y, t) = 0$$

where x are internal states, y contain measured signals, and t is the time. Some DAE's can be written on a semi-explicit form

$$\begin{aligned} \dot{x} &= f(x, y, t) \\ 0 &= g(x, y, t). \end{aligned} \tag{2.3}$$

where the dynamic equations are separated from the algebraic constraints. The algebraic constraints make the DAE harder to solve and simulate than a system of Ordinary Differential Equations, ODE, which can be written as

$$\dot{x} = f(x, y, t),$$

and does not have algebraic constraints.

2.1.3 The Index of a DAE

To classify the complexity of a DAE an index is used. On page 36 in [2] it is said that *"From the point of view of the numerical solution, it is desirable for the DAE to have an index which is as small as possible."*

The following definition of the DAE index comes from [1] where the notation of z stands for both x and y:

Definition 2.1 *For general DAE systems $F(t, z, z') = 0$, the index along a solution $z(t)$ is the minimum number of differentiations of the system which would be required to solve for z' uniquely in terms of z and t (i.e., to define an ODE for z). Thus the index is defined in terms of the overdetermined system*

$$\begin{aligned} F(t, z, z') &= 0 \\ \frac{dF}{dt}(t, z, z', z'') &= 0 \\ &\vdots \\ \frac{d^pF}{dt^p}(t, z, z..., z^{p+1}) &= 0 \end{aligned}$$

to be the smallest integer p so that z' can be solved for in terms of z and t.

To follow Definition 2.1 may not always be the most practical way of determining the index. Another way to check what index a DAE has can be found in [2]. With the semi-explicit DAE in (2.3) it is possible to differentiate the algebraic constraint with respect to t which gives

$$\begin{aligned} \dot{x} &= f(x, y, t) \\ g_x(x, y, t)\dot{x} + g_y(x, y, t)\dot{y} &= -g_t(x, y, t). \end{aligned} \tag{2.4}$$

If g_y is non singular, the system has index 1 and is called an implicit ODE. If g_y is singular and it is possible with algebraic manipulation and coordinate changes to rewrite (2.4) on the form of (2.3) then one can continue with another differentiation step. The number of steps required to get a non singular g_y, in other words an implicit ODE, is the index of the DAE. Note that in this thesis, systems of index higher than one will not be handled so only one differentiation step is needed here.

2.1.4 Model Linearization

Given a dynamical system on the form $\dot{x} = g(x, y)$, where x and y are vectors and a linearization point (x_0, y_0), the system can be linearized as:

$$\begin{aligned}\dot{x} = g(x,y) = g(x_0 + \Delta x, y_0 + \Delta y) &= \\ &= g(x_0, y_0) + \underbrace{\frac{\partial g(x_0, y_0)}{\partial x}}_{= \text{A}} \Delta x + \underbrace{\frac{\partial g(x_0, y_0)}{\partial y}}_{= \text{B}} \Delta y + \mathcal{O}(\Delta x^2, \Delta y^2) \approx \\ &\approx g(x_0, y_0) + A\Delta x + B\Delta y,\end{aligned} \tag{2.5}$$

where the matrices A and B are defined as

$$\begin{aligned} A &= \frac{\partial g(x_0, y_0)}{\partial x} \\ B &= \frac{\partial g(x_0, y_0)}{\partial y}. \end{aligned} \tag{2.6}$$

In order to compute the linearization in (2.5) numerically, the elements in the matrices A and B need to be approximated. A and B are matrices with elements consisting of derivatives as shown in (2.6). One way to compute this is by using forward differentiation

$$\begin{aligned} A_{ij} &= \frac{g_i(x_0 + \varepsilon e_j, y_0) - g_i(x_0, y_0)}{\varepsilon} \\ B_{ij} &= \frac{g_i(x_0, y_0 + \varepsilon e_j) - g_i(x_0, y_0)}{\varepsilon}, \end{aligned}$$

where A_{ij} and B_{ij} is the element in the i:th row and the j:th column of matrix A and B respectively, ε is a small number and vector e_j is a zero vector except for the element j which equals one.

2.2 Nonlinear Least Square Estimation of Parameters

For the model to be of use, unknown parameters need to be estimated for the non linear system. This can be made using nonlinear least square estimation, see [13]. Assume that the parameters in the vector θ in

$$r = g(x, \theta)$$

are to be estimated. The variables x are known and r is a residual that ideally is zero and is therefore in the parameter estimation set to zero. The penalty function f_p is created as

$$f_p(\theta) = \sum_{i=1}^{N} (0 - g(x_i, \theta))^2,$$

where N is the number of data samples. The estimation $\hat{\theta}$ is calculated with

$$\hat{\theta} = \arg\min_{\theta} f_p(\theta).$$

The better the parameters are estimated the smaller the value of the penalty function. For the minimization procedure a simplex algorithm, see [7], can be used.

2.3 Diagnosis

The purpose of fault diagnosis is to get information about what condition the system is in and to discover when components break down. In a perfect world the systems would never break or have faults and there would, in general, be little sense in doing diagnosis. In reality however the systems does not work perfectly and the need for diagnostics can for instance be a safety or quality control precaution.

2.3.1 Residuals

Residuals are the results of residual generators $f(x, y)$, where the signals y are measured and x are states. The residual generators are made from a model of the real system, see Section 1.1.1. The model with the residuals can be written as

$$\begin{aligned} \dot{x} &= g(x, y) \\ r &= f(x, y). \end{aligned}$$

The states x are unknown so estimations $\hat{x}$ are used. A naive system is

$$\begin{aligned} \dot{\hat{x}} &= g(\hat{x}, y) \\ r &= f(\hat{x}, y). \end{aligned} \tag{2.7}$$

For r to be a valid residual it has to be stable which may not be the case in (2.7). To stabilise it, feedback is used

$$\begin{aligned} \dot{\hat{x}} &= g(\hat{x}, y) + h(r) \\ r &= f(\hat{x}, y). \end{aligned} \tag{2.8}$$

One way to design the feedback function h is to use a Kalman filter. In order to do that, (2.8) needs to be linearized as

$$\begin{aligned} \dot{x} &= Ax + By \\ r &= Cx + Dy. \end{aligned} \tag{2.9}$$

To linearize the system in (2.7) to get it on the form of (2.9), assumptions are made which are discussed in Section 4.2.1. The linearized system in (2.9) is only used to calculate h. For simulation, (2.8) is used.

2.3.2 Observers using Kalman Filters

To estimate non measured states in a state-space model the Kalman filter can be used, see [5]. The Kalman filter is an optimal linear filter that works in two alternating steps. First there is a time update where a first prediction of the upcoming estimate is made that is based on the previous value of the state. The second step is the measurement update. Here the information from the predicted estimate is fusioned with measurements to create a likely better estimate. Both the predicted values and the measured values have uncertainties or noise. The less noisy a measurement is the more weight is put on it in the fusion and the more it is going to show in the final estimate. The state space model with noise is written as

$$\begin{aligned} \dot{x} &= Ax + By + Gw \\ \tilde{r} &= Cx + Dy + v, \end{aligned} \tag{2.10}$$

where v and w are noise that are assumed independent and identically distributed. Residual $\tilde{r}$ is an ideal residual affected by true noise. The matrices

$$\begin{aligned} \mathrm{Cov}(w) &= Q \\ \mathrm{Cov}(v) &= R. \end{aligned} \tag{2.11}$$

are covariance matrices that will decide how much weight is put on the predictions relative the measurements of the states in the state estimation. When Q and R are unknown they can be seen as tuning parameters.

The following is the observer

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + By + K(-r) \\ r &= C\hat{x} + Dy, \end{aligned} \tag{2.12}$$

where K is the Kalman gain and $\hat{x}$ is the estimate of x.

If the noise vectors w and v are independent, the Kalman gain matrix K is written as

$$K = PC^T R^{-1}$$

where P is the symmetrical positive semidefinite solution to the equation

$$AP + PA^T - PC^T R^{-1}(PC^T)^T + GQG^T = 0,$$

see [4].

Chapter 3

Modeling

In model based diagnosis a model of the system that is to be diagnosed is needed. In this chapter some principles of how the model can be made are presented. First it is explained how a system can be modeled componentwise and then it is shown which equations that are needed to connect the components to build a larger model. There is also a note on the parameter estimation that is done for unknown parameters in the model.

3.1 Modeling of Components

The components of a system are here assumed to be modeled separately. The system can for example be an electrical circuit with the components consisting of batteries, resistors, relays, etc. The behavior of each component is described by a set of equations. Depending on the fault mode, different sets of equations are used. To identify which equations that describe the behavior of each fault mode, the equations are tagged. The tag represents the fault mode for which the equation is valid. Equations that are valid for all fault modes are untagged.

Example 3.1: Equations of a Real Load Component

A resistor in a DC circuit can be described by

$$V = (R + R_b)I \tag{3.1a}$$

$$V = V_p - V_n \tag{3.1b}$$

$$R_b = 0 \ [OK] \tag{3.1c}$$

$$\frac{dR_b}{dt} = 0 \ [OS], \tag{3.1d}$$

where I is the current through the resistor, V is the voltage over the resistor, V_p and V_n are the potentials on either side of the resistor, R is the nominal resistance and R_b is the resistance offset. The tag [OK] in (3.1c) marks that this equation is only valid in the fault free case. The tag [OS] in (3.1d) indicates that this equation is valid only in the case of a constant offset fault. The untagged equations (3.1a) and (3.1b) are always valid. If the current fault mode is fault free, [OK], the following equations are used

$$\begin{aligned} V &= (R + R_b)I \\ V &= V_p - V_n \\ R_b &= 0. \end{aligned}$$

3.2 Equation Set Generation

For each fault mode of the system, a different set of equations defines the system. A set of equations that describes a system is not only determined by the equations describing the behaviour of each component but also of equations connecting the component variables in different components. A set of equations includes the following:

- Component Equations: Equations for each component in one fault mode.
- Connection Equations: Equations for connecting components.

For electrical systems the Connection Equations will connect the currents and the potentials defined in the components using Kirchhoff's Current Law so that all the equations together define a circuit.

Example 3.2: Equation Generation of Real Load Components

Two fault free resistors are to be connected in a series. The component equations are

$$\begin{aligned} &\text{Resistor 1} \begin{cases} V_1 &= (R_1 + R_{b1})I_1 \\ V_1 &= V_{p1} - V_{n1} \\ R_{b1} &= 0 \end{cases} \\ &\text{Resistor 2} \begin{cases} V_2 &= (R_2 + R_{b2})I_2 \\ V_2 &= V_{p2} - V_{n2} \\ R_{b2} &= 0. \end{cases} \end{aligned} \tag{3.2}$$

To make the serial connection of these two components, equations that connect I_1 with I_2 and V_{n1} with V_{p2} are needed. The following equations are added to complete the equation system

$$\begin{aligned} V_{n1} &= V_{p2} \\ I_1 &= I_2. \end{aligned} \tag{3.3}$$

Together the equations (3.2) and (3.3) describe two serially connected resistors.

3.3 Parameter Estimation

When the equations have been generated and handled the unknown parameters can be estimated by using a nonlinear least square estimation described in Section 2.2. This method can be used to estimate many parameters at once, when, e.g., estimating resistances of many resistors.

Chapter 4

The Algorithm

This chapter describes the algorithm that is used every time the system goes into a new fault mode. In the thesis the fault mode is constant but the algorithm is made with transitions in mind. An overview of the algorithm can be seen in Figure 4.1.

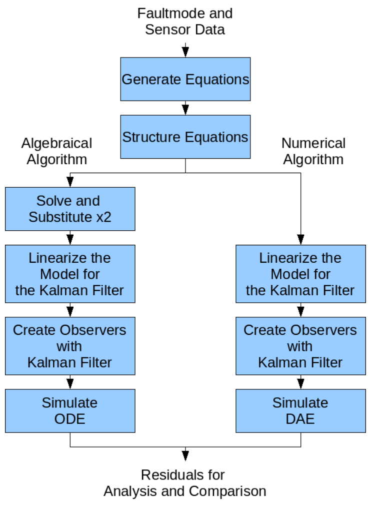

Faultmode and Sensor Data
Generate Equations
Structure Equations
Algebraical Algorithm
Numerical Algorithm
Solve and Substitute x2
Linearize the Model for the Kalman Filter
Create Observers with Kalman Filter
Simulate ODE
Linearize the Model for the Kalman Filter
Create Observers with Kalman Filter
Simulate DAE
Residuals for Analysis and Comparison

Figure 4.1. An overview of the algorithm.

The algorithm is divided into two parallel paths in the algorithm. The paths are called the algebraical algorithm and the numerical algorithm. Ideally all calculations are done numerically because not all problems can be solved algebraically. However the algebraical algorithm is easier to implement and better simulation results can be expected from this algorithm in comparison with the numerical algorithm. An algebracial algorithm comes at the cost of calculation efficiency and an algebraical algorithm can not solve all the mathematical problems a numerical algorithm can. Therefore the algebraical algorithm is used to evaluate the numerical algorithm. An important difference between the algorithms is that the algebraical algorithm simulates a system of ordinary differential equations, ODE, and the numerical algorithm simulates a system of differential-algebraic equations, DAE. The two algorithms start off the same and are splitted later on. The split is explained later in this chapter.

4.1 Equation Structuring

It is assumed that all the equations and conditions of the model of the system can be written as

$$\begin{aligned} &g_i(\dot{x}, x, y) = 0, \\ &g_{\text{cond}}(x, y), \end{aligned}$$

where x contains unknown variables and y contains measured signals, and g_{cond} contains relations of inequalities. The unknown x can be divided into x_1 and x_2 where x_1 are dynamic variables which occurs time differentiated in some relations, and x_2 are variables which do not occur differentiated in any relations. This gives

$$\begin{aligned} &g_i(\dot{x}_1, x_1, x_2, y) = 0 \\ &g_{\text{cond}}(x_1, x_2, y). \end{aligned}$$

The equations can be divided into dynamical and algebraical equations. It is assumed that $\dot{x}_1$ can be written explicitly which gives

$$\begin{aligned} &g_{\text{dyn}}(x_1, x_2, y) = \dot{x}_1 \\ &g_{\text{alg}}(x_1, x_2, y) = 0 \\ &g_{\text{cond}}(x_1, x_2, y). \end{aligned}$$

It is assumed that g_{alg} is overdetermined with respect to x_2. To be able to compute the non dynamic variables x_2, a subset that is exactly solvable of the equation system g_{alg} is needed. The system g_{alg} is therefore divided into g_{alg1} and g_{alg2}

$$\begin{aligned} &g_{\text{dyn}}(x_1, x_2, y) = \dot{x}_1 \\ &g_{\text{alg1}}(x_1, x_2, y) = 0 \\ &g_{\text{alg2}}(x_1, x_2, y) = 0 \\ &g_{\text{cond}}(x_1, x_2, y), \end{aligned} \tag{4.1}$$

where g_{alg1}, given x_1 and y, is an exactly solvable system and g_{alg2} contains all the rest of the equations in the overdetermined system g_{alg}. How this dividing process is done is shown further in Section 4.1.1. The equations in g_{alg2} and g_{cond} will be used for residuals.

4.1.1 About dividing g_{alg}

The system of equations

$$g_{\mathrm{alg}} = 0$$

is, in this thesis, assumed to be overdetermined with respect to x_2. The equations g_{alg} are split into g_{alg1} and g_{alg2} where g_{alg1} is exactly solvable with respect to x_2 and g_{alg2} is the rest of the relations that are used for residuals.

To find the equations for g_{alg1} a subset of the equations in $g_{\mathrm{alg}} = 0$ is chosen excluding one equation. The subset is then tested to see if it is underdetermined with respect to x_2. If the subset is underdetermined the equation is taken back into the set but if the subset is not underdetermined the excluded equation remains excluded. The steps are then repeated until all original equations in $g_{\mathrm{alg}} = 0$ have been test-excluded. After this the remaining subset is perfectly solvable with respect to x_2. The remaining subset are the equations of g_{alg1} and the excluded equations are the equations of g_{alg2}.

To test if a system of equations, $g_{\mathrm{sub}}(x_1, x_2, y)$, is underdetermined with respect to x_2, the system $g_{\mathrm{sub}}(x_1, x_2, y)$ is differentiated with respect to x_2

$$\frac{\partial g_{\mathrm{sub}}(x_1, x_2, y)}{\partial x_2} = M,$$

where x_2 is a vector and M is a matrix. Since the system is overdetermined initially, the column rank of M will be full at the start. If the column rank remains unchanged after a row has been removed, the corresponding system $g_{\mathrm{sub}} = 0$ of the changed matrix M is either still overdetermined or exactly solvable. When the column rank drops by one, the system is underdetermined and the previous corresponding $g_{\mathrm{sub}} = 0$ is then an exactly solvable system that has been searched for.

Note that in order to fully test the column rank of M, the rank should be computed for all operating points (x_1^*, x_2^*, y^*) to see if there exist singularities. The matrix M can lose its column rank for certain points causing singularity and this could result in failed simulations. An analysis considering if the matrix M becomes singular for certain operating points has not been performed.

This way of dealing with the dividing process of the equations is an algebraical method. Another non algebraical way of dealing with the dividing process of the equations is to use structural analysis. This has not been tested in the thesis but more information can be found in [6].

4.1.2 Using Algebraic Part-Solution

One of the subsystems can sometimes be solved algebraically as

$$g_{\mathrm{alg1}}(x_1, x_2, y) = 0 \implies x_2 = G_{\mathrm{alg1}}(x_1, y).$$

A variable substitution of x_2 in the equations gives

$$\begin{aligned} \tilde{g}_{\mathrm{dyn}}(x_1, y) &= \dot{x}_1 \\ \tilde{g}_{\mathrm{alg2}}(x_1, y) &= 0 \\ \tilde{g}_{\mathrm{cond}}(x_1, y) &= 0. \end{aligned} \tag{4.2}$$

Note that the dynamic equation in (4.2),

$$\dot{x}_1 = \hat{g}_{\text{dyn}}(x_1, y), \tag{4.3}$$

is an ODE.

4.1.3 Using Non-Algebraic Part-Solution

In order to numerically find the estimations in the simulation the steps in Section 4.1.2 are to be avoided because it calls for the use of algebraic solvers. In order to resolve x_2 the model has to be linearized first but this is done for the design of the Kalman filter and not for the actual simulation where the non linear model is used and x_2 is simulated. The model so far is seen in (4.1). The next step for the numerical algorithm is to linearize the system in order to make a Kalman filter.

4.2 Observers

In order to make an observer a Kalman filter is used. The model of the system is nonlinear and must therefore first be linearized before a Kalman filter can be made. This section deals with both the linearization and the design of the Kalman filter.

4.2.1 Linearization

In Section 4.1 the algorithm got splitted into two algorithms, the algebraical algorithm and the numerical algorithm. These algorithms handle the linearization similar but have some differences and are therefore divided into different subsections below. Both algorithms accomplishes a linear state space model

$$\begin{aligned} \Delta\dot{x}_1 &= A\Delta x_1 + B\Delta y \\ r &= C\Delta x_1 + D\Delta y \end{aligned} \tag{4.4}$$

so that the Kalman filter can be calculated.

The linearization of a function $g(x, y)$ with the linearization point (x_0, y_0) gives

$$g(x, y) \approx g(x_0, y_0) + M\Delta x + N\Delta y$$

where M and N are matrices. To get a linear state space model the constant term $g(x_0, y_0)$ must be small so that it can be neglected. This can be done by choosing a stationary point as the linearization point. When the model is correct, the states and residuals should then be small, which makes $g(x_0, y_0)$ small.

In this thesis however it has not been convenient to choose a stationary point because it has been very far from the operating point. An operating point has instead been chosen as the linearization point. The term $g(x_0, y_0)$ is still assumed small as

$$g(x, y) \approx \underbrace{g(x_0, y_0)}_{\approx 0} + M\Delta x + N\Delta y \approx M\Delta x + N\Delta y.$$

Note that this can be a very rough approximation. It is important to remember that the linearized system is only used for computing the Kalman filter. The linearized system is not used for simulation.

4.2.2 Linearization in the Algebraic Algorithm

The equations

$$\begin{aligned}\tilde{g}_{\text{dyn}}(x_1, y) &= \dot{x}_1\\ \tilde{g}_{\text{alg2}}(x_1, y) &= 0.\end{aligned}$$

extracted from (4.2) are linearized. This is done numerically, described in Section 2.1.4, and gives

$$\begin{aligned}\tilde{g}_{\text{dyn}}(x_1, y) &\approx \underbrace{\tilde{g}_{\text{dyn}}(x_0, y_0)}_{\approx 0} + A\Delta x_1 + B\Delta y\\ \tilde{g}_{\text{alg2}}(x_1, y) &\approx \underbrace{\tilde{g}_{\text{alg2}}(x_0, y_0)}_{\approx 0} + C\Delta x_1 + D\Delta y.\end{aligned}$$

The result is the system

$$\begin{aligned}\Delta\dot{x}_1 =& A\Delta x_1 + B\Delta y\\ r =& C\Delta x_1 + D\Delta y,\end{aligned} \tag{4.5}$$

which is in a state space form suitable for the design of the Kalman filter.

4.2.3 Linearization in the Numerical Algorithm

The equations

$$\begin{aligned}g_{\text{dyn}}(x_1, x_2, y) &= \dot{x}_1\\ g_{\text{alg1}}(x_1, x_2, y) &= 0\\ g_{\text{alg2}}(x_1, x_2, y) &= 0\end{aligned}$$

extracted from (4.1) are to be linearized. This is done numerically, described in Section 2.1.4, and gives

$$\begin{aligned}g_{\text{dyn}}(x, y) &\approx \underbrace{g_{\text{dyn}}(x_0, y_0)}_{\approx 0} + A_1\Delta x_1 + A_2\Delta x_2 + A_3\Delta y\\ g_{\text{alg1}}(x, y) &\approx \underbrace{g_{\text{alg1}}(x_0, y_0)}_{\approx 0} + B_1\Delta x_1 + B_2\Delta x_2 + B_3\Delta y\\ g_{\text{alg2}}(x, y) &\approx \underbrace{g_{\text{alg2}}(x_0, y_0)}_{\approx 0} + C_1\Delta x_1 + C_2\Delta x_2 + C_3\Delta y.\end{aligned}$$

The variable x_2 needs to be eliminated to make a state space model. Under the assumption that the model perfectly describes the reality then

$$g_{\text{alg1}} = 0.$$

It is possible to resolve Δx_2 and eliminate it from the expressions with

$$\Delta x_2 = -(B_2)^{\dagger}(B_1 \Delta x_1 + B_3 \Delta y),$$

where $\dagger$ denotes the pseudoinverse, $B_2^{\dagger} = (B_2^T B_2)^{-1} B_2^T$, see [10]. This gives

$$\begin{aligned} D_1 &= A_1 + A_2(B_2)^{\dagger}(-B_1) \\ D_2 &= A_3 + A_2(B_2)^{\dagger}(-B_3) \end{aligned}$$

$$\begin{aligned} E_1 &= C_1 + C_2(B_2)^{\dagger}(-B_1) \\ E_2 &= C_3 + C_2(B_2)^{\dagger}(-B_3) \end{aligned}$$

and the linearized system can be written as

$$\begin{aligned} \Delta \dot{x}_1 &= D_1 \Delta x_1 + D_2 \Delta y \\ r &= E_1 \Delta x_1 + E_2 \Delta y. \end{aligned}$$

which is the desired state space model.

4.2.4 Tuning the Kalman Filter

A Kalman filter is created as described in Section 2.3.2. A choice of the Q and R matrices needs to be done. The relation between these matrices is important and this can be tuned. The matrices are chosen as

$$\begin{aligned} Q &= \lambda \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix} \\ R &= \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix} \\ &Q \in \mathbb{R}^{n \times n} \ R \in \mathbb{R}^{q \times q}, \end{aligned} \tag{4.6}$$

where λ is a tuning parameter, n is the number of states and q is the number of residuals. The matrices in (4.6) are a rough approximation of the covariance matrices. A more accurate approximation would be to use

$$
\begin{aligned}
Q &= \lambda \begin{pmatrix} \mathrm{Var}(x_{1,1}) & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \mathrm{Var}(x_{1,n}) \end{pmatrix} \\
R &= \begin{pmatrix} \mathrm{Var}(r_1) & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \mathrm{Var}(r_q) \end{pmatrix},
\end{aligned} \tag{4.7}
$$

where $\mathrm{Var}(x_1)$ are the variances of the states and $\mathrm{Var}(r)$ are variances of the residuals. Note that the tuning parameter λ is used here as well. A problem with calculating $\mathrm{Var}(r)$ is that the residuals r are unknown until the system has been simulated. This means that another approximation method has to be used first. The approximation in (4.6) is chosen in this thesis but a comparison of the simulation results given with (4.6) and the simulation results given with (4.7) is made in Section 5.7.

4.3 Simulation

The simulation is done using the MATLAB solver *ode15s*. Documentation for this is found in [12]. This solver is a multistep solver. It is based on numerical differentiation formulas. It optionally uses the backward differentiation formulas. This solver can handle DAEs with index one and it also handles stiff equation systems. For information on DAE and index, see Section 2.1.2.

Important parameters that are needed for the simulation are the error tolerances; *RelTol* for the relative tolerance and *AbsTol* for the absolute tolerance with the same variable names as the MATLAB documentation.

The relative tolerance *RelTol* is the largest acceptable error relative the size of the state during a time step. If the relative error is exceeded the time step size is reduced.

The absolute tolerance *AbsTol* specifies the largest acceptable solver error as the value approaches zero. If the value is exceeded the time step is reduced.

4.3.1 Simulation in the Algebraical Algorithm

The algebraical algorithm is made as a reference to the numerical algorithm to better judge how well the numerical algorithm simulates.

The system to simulate is

$$
\begin{aligned}
\dot{\hat{x}}_1 &= \tilde{g}_{\mathrm{dyn}}(\hat{x}_1, y) - Kr \\
r &= \tilde{g}_{\mathrm{alg2}}(\hat{x}_1, y),
\end{aligned}
$$

which is an implicit ODE.

Simulation Initialization Point

A simulation initialization point needs to be approximated. The same point that is used as the linearization point is also used here as the simulation initialization point.

4.3.2 Simulation in the Numerical Algorithm

The system to simulate is

$$\begin{aligned} \dot{\hat{x}}_1 &= g_{\text{dyn}}(\hat{x}_1, \hat{x}_2, y) - Kr \\ 0 &= g_{\text{alg1}}(\hat{x}_1, \hat{x}_2, y) \\ r &= g_{\text{alg2}}(\hat{x}_1, \hat{x}_2, y), \end{aligned}$$

which is a DAE with the algebraic constraints in $g_{\text{alg1}}(\hat{x}_1, \hat{x}_2, y) = 0$.

Simulation Initialization Point

As with the algebraical algorithm, the linearization point is also used as a initialization point for the numerical algorithm. It can be worth noting that the points for the numerical algorithm have a larger dimension than the points for the algebraical algorithm, because of how x_2 is handled differently.

Chapter 5

Application on ADAPT-Lite

In this chapter the method presented in Chapter 4 is applied to the system ADAPT-Lite and the generated residuals are analysed. The mathematical models for the different components in ADAPT-Lite can be found in Appendix A.

5.1 ADAPT-Lite Overview

The ADAPT-Lite system is presented again in Figure 5.1. The abbreviations are explained in Table 1.1 which also gives an overview of the components present in the system. Some components exist in both a direct current, DC, version and an alternating current, AC, version. All components connected to the left of the inverter, INV2, in Figure 5.1 are DC components, and all components connected to the right are AC components.

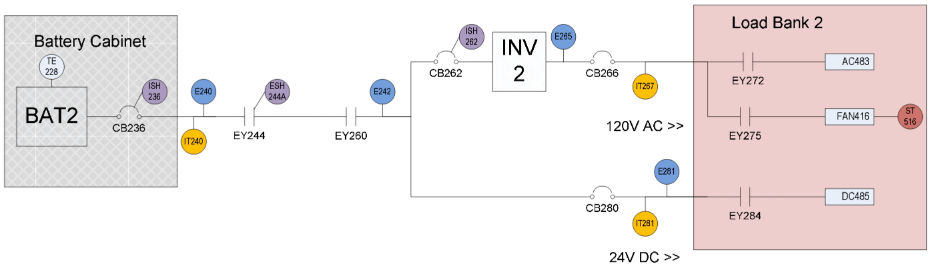


Figure 5.1. Schematic over the ADAPT-Lite satellite system. The picture comes from [11].

5.2 Starting Point of the Project

At the start of the application of the method on ADAPT-Lite, some work had already been done at the department. Models for most of the components al-

Parameter	Value	Unit
$Z_{\mathrm{r,fan}}$	6.3	Ω
$Z_{\mathrm{i,fan}}$	51.3	Ω
$k_{0,\mathrm{fan}}$	158	-
$k_{\mathrm{t,fan}}$	4.3	-
K_{bat}	3.7×10^{-3}	-
R_{bat}	1.6×10^{-5}	Ω
R_{244}	0.0027	Ω
R_{260}	0.0027	Ω
R_{266}	24.8	Ω
R_{272}	109.6	Ω
R_{275}	4.9	Ω
R_{280}	0.2086	Ω
R_{284}	2×10^{-16}	Ω
R_{483}	625	Ω
R_{485}	9.69	Ω
$P_{0,\mathrm{inv}}$	3.2×10^{2}	W

Table 5.1. The table shows values of the model parameters that have been estimated for the ADAPT-Lite system.

ready existed and a program in Mathematica, to build a model and to generate MATLAB files, already existed and worked. What the Mathematica program has been complemented with is a battery model, a fan model and the equation handling described in Section 4.1. The subsequent steps explained in Chapter 4 have been programmed in MATLAB. Functions for easier handling of the reading of the measured indata was already prepared for the MATLAB program but the rest has been done in this thesis.

5.3 Estimated Model Parameters

The values of the estimated model parameters are presented in Table 5.1. The values are presented for the interested reader but it is not necessary to analyze the values in order to understand the rest of the report.

5.4 Generated Equations

The number of equations that are generated varies depending on fault mode but in this project the number of equations are very similar between the fault modes. The number of equations generated for the fault mode NF, No Fault, is shown in Table 5.2. Two equations contain dynamics, which means there will be two states. Out of 103 algebraic equations 94 are sorted into g_{alg1} and 9 into g_{alg2}. There is also one conditional relation which is also used as a residual.

System of Equations	Number of Equations
$g_{\text{dyn}} = 0$	2
$g_{\text{alg1}} = 0$	94
$g_{\text{alg2}} = 0$	9
g_{cond}	1

Table 5.2. The number of equations in the fault mode NF, i.e., no fault.

5.4.1 Dynamic Equations

In the fault mode NF, No Fault, the dynamic equations, for which the observers are made, are

$$\frac{dV_{\text{bat}}}{dt} = K_{bat} I, \tag{5.1}$$

$$\frac{d\omega_{\text{fan}}}{dt} = k_t(k_0\sqrt{max(0, P)} - \omega_{\text{fan}}). \tag{5.2}$$

Equation (5.1) comes from the battery model and describes how the time derivative of the internal battery voltage is proportional with the current from the battery. More information on the battery model can be found in Appendix A.1.

Equation (5.2) comes from the fan model and models how the fan speed varies. Note that in the implementation a *max* function has been added in the square root expression as seen in (5.2) to avoid imaginary answers. More information on the fan model can be found in Appendix A.4.

5.4.2 The Residual Equations

The resulting residuals constructed out of $g_{\text{alg2}} = 0$ are

$$\begin{aligned}
r_1 &= i_{\text{EY275},Im} - i_{\text{FAN416},Im} \\
r_2 &= i_{\text{EY275},Re} - i_{\text{FAN416},Re} \\
r_3 &= I_{\text{DC485}} - I_{\text{EY284}} \\
r_4 &= V_{\text{EY260},n} - V_{\text{CB280},p} \\
r_5 &= I_{\text{EY260}} - I_{\text{EY244}} \\
r_6 &= I_{\text{CB236}} - I_{\text{BAT2}} \\
r_7 &= x_{24} \\
r_8 &= ESH244A - x_3 \\
r_9 &= ISH236 - x_2,
\end{aligned} \tag{5.3}$$

where subscript n respectively p denotes the components NPin and PPin, subscript Re respectively Im denotes the real and imaginary part of a complex value. I and V are DC currents and voltages or potentials and i and v are AC currents and voltages or potentials. The other variables are explained in Table 5.3. The location of the residuals in the system are visualized in Figure 5.2, which shows which parts of the system that are covered by residuals.

x_2	position of the circuit breaker CB236
x_3	position of the relay EY244
x_{24}	offset in the measurement of the fan speed
$ESH244A$	sensor measurement of x_3
$ISH236$	sensor measurement of x_2

Table 5.3. Explanations of variables in the residual equations.

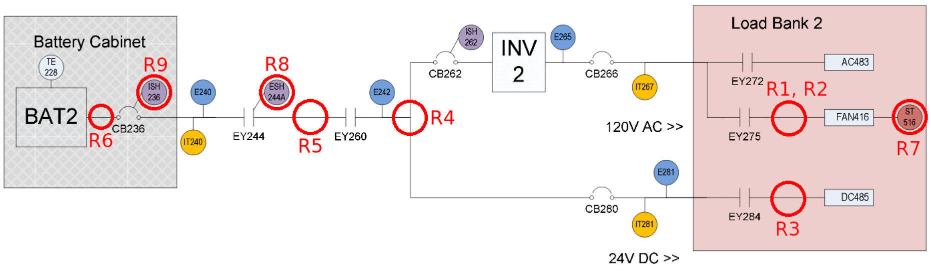


Figure 5.2. Schematic over the ADAPT-Lite satellite system with the location of the residuals visualized. The circles show which connections or components the residuals are centered around but they do however not show all components the residuals are dependent on.

5.4.3 The Conditional Relation

The single generated conditional relation is

$$(V_{p,Inverter} > 22) == (v_{n,Re,Inverter} > 120)$$

where, $==$, denotes a logical test to see if the left hand side equals the right hand side. The relation comes from the model of the inverter; see Appendix A.5 for more details about the model.

The relation is reformulated with the logical operator XOR, *exclusive or*, and made into a residual r_{cond},

$$r_{\text{cond}} = \text{XOR}\Big((V_{p,Inverter} > 22),\ (v_{n,Re,Inverter} > 120)\Big),$$

which means $r_{\text{cond}} = 0$ if both conditions in the operation *exclusive or* are true or false. If only one condition is true then $r_{\text{cond}} = 1$. The discrete answer $r_{\text{cond}} = 0$ means that it is possible, but not certain, that the inverter is OK and $r_{\text{cond}} = 1$ means that there is a fault.

5.4.4 The Kalman Gain

A choice of the tuning parameters λ described in Section 4.2.4 must be made in the design of the Kalman filter.

The Kalman gain matrices for the algebraical and the numerical algorithm with $\lambda = 10^{-1}$ has been computed in the fault free mode as

$$\begin{aligned} K_{\text{alg}} &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 320 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 310 & 0 & 0 \end{pmatrix} \\ K_{\text{num}} &= \begin{pmatrix} 0.067 & -0.050 & 0 & 0 & -0.28 & 0 & -0.13 & 0 & 0 \\ -3.8 & 2.8 & 0 & 0 & 16 & 0 & 21 & 0 & 0 \end{pmatrix}. \end{aligned} \tag{5.4}$$

The number of columns are the same as the number of residual equations $r = g_{\text{alg2}}$. The gain matrices do not look the same for all fault modes and data sets. What residuals are used in the feedback varies and is not only residual r_6 and r_7 as in (5.4). The magnitude of λ do affect the magnitude of the gain which but since it is here set to the same value, this is not the reason for the difference. One reason may be that the R matrices are slightly different. This is because g_{alg2} differs for the two algorithms. The linearization point also has a different number of coordinate elements in the two algorithms because x_2 is algebraically solved in one algorithm but not in the other. If using the R matrix from the algebraically algorithm in the numerical algorithm instead of the one calculated in that algorithm the values in the K matrix becomes much more similar between the two algorithms. This is however not done in the presented simulations in this report.

5.5 Algebraical Shortcuts for Numerical Algorithm

Two shortcuts in the numerical algorithm, which include algebraical solving, have been used even though it would have been preferred to only use numerical solving for this algorithm. The shortcuts used are explained in this section.

5.5.1 Linearization Point and Initial Simulation Point

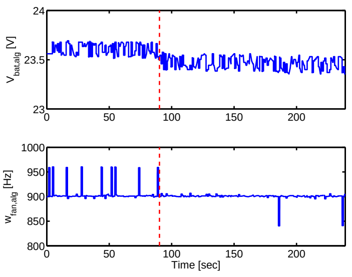
The calculation of the linearization point is done the same for the algebracial and the numerical algorithm. This computation invovles using algebraical steps. For the numerical algorithm this becomes an algebraical shortcut. The linearization point is an operating point and the point is also used as an initial point for the simulation.

To see the effect of this shortcut, the point is also approximated with a vector with the value of 1 in every element position. In Figure 5.3 the two methods are compared by plotting the simulated states for both algorithms, both with the two different linearization points. It is possible to note that the curve shapes are slightly different but also that the magnitudes are comparable and rather similar.

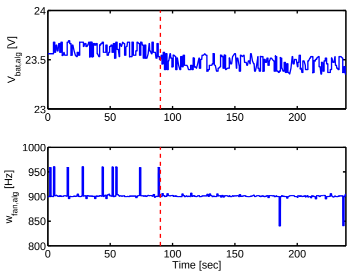
5.5.2 Reformulation of the Algebraic Constraints

The numerical algorithm only seem to simulate when the algebraic constraints of the DAE are reformulated with algebraical calculations. Nominally the algebraic constraints of the DAE are formulated as

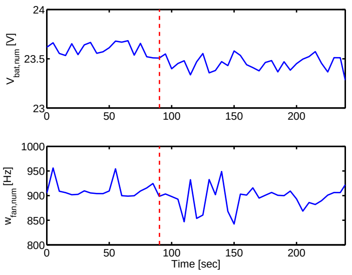
$$0 = g_{\text{alg1}}(x_1, x_2, y).$$



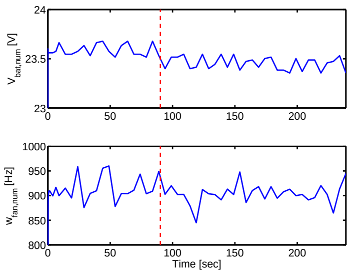
(a) states, algebraical algorithm, lin. point made of operating point



(b) states, numerical algorithm, lin. point made of ”ones”



(c) states, numerical algorithm, lin. point made of operating point



(d) states, numerical algorithm, lin. point made of ”ones”

Figure 5.3. The algebraical and the numerical algorithm is simulated with a linearization point and simulation start point consisting, in (a) and (c) of an operating point, and in (b) and (d) of a vector made from only ones. Note that the jagged shape of state ω in (a) and (b) are caused by discretizations in the sensor measurements. The states are simulated in fault mode NF, with fault f_1 introduced at 90.2 sec, dashed line. Fault modes and faults are explained more in Section 5.6.2 but are not very important for now. The curves with the different linearization points differs slightly for state ω with the numerical alorithm but the main observation is that the linearization point has not mattered very much.

This is modified by algebraically solving g_{alg1} with respect to x_2 as

$$g_{\text{alg1}}(x_1, x_2, y) = 0 \implies x_2 = G_{\text{alg1}}(x_1, y).$$

Instead of substituting x_2 in all the other model equations, as is done for the algebraical algorithm, new relations are created,

$$0 = x_2 - G_{\text{alg1}}(x_1, y),$$

which will be the new algebraic constraints used instead of the nominal ones. Note that the system is basically the same but with a new form. Why this makes a difference for the solver used for the simulation has not been successfully analysed in the thesis but it has been a necessary step to get any results from the numerical algorithm.

5.6 Simulation

In this section the simulation tolerances are set, the fault modes and faults are defined and simulations of different fault modes in combinations with faults are simulated.

5.6.1 Simulation Tolerances

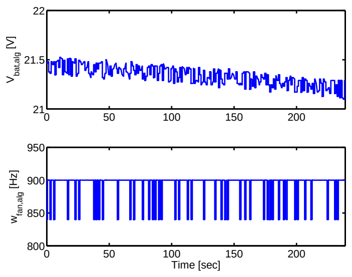
The simulation parameters, $RelTol$ for the relative tolerance and $AbsTol$ for the absolute tolerance, must be set for the numerical algorithm. The algebraical algorithm, is not as sensitive to the choice of the parameters as the numerical algorithm and the default parameters can be used. If the parameters for the numerical algorithm are set too large, the precision of the simulation will be low. If the tolerances are set to small the simulation may not converge or have to abort earlier than supposed. Values of different sizes have been tested and manually iterated until good values have been found. How good the parameter values are can be evaluated with the simulated plots of the states. In MATLAB the default values are $RelTol = 10^{-3}$ and $AbsTol = 10^{-6}$. These do however not work very well. In Figure 5.4 two sets of parameters are tested. In Figure 5.4(b) the simulation is aborted before simulation completion because the tolerances can not be met when the parameters are set too small. The simulation in Figure 5.4(c) look more reasonable and the parameters, $RelTol = 10^{1}$ and $AbsTol = 10^{0}$, are the simulation tolerance parameters used in all the subsequent simulations in this chapter in the numerical algorithm.

5.6.2 Defining the Fault Modes and the Faults

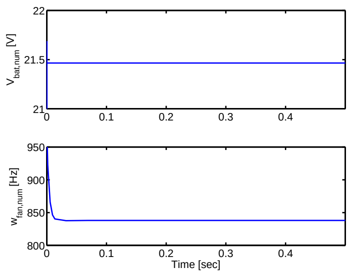
In this thesis only single faults are studied. This means that each fault mode, except *no fault*, has one fault that corresponds to the fault mode. The faults and fault modes are presented in Table 5.4. The magnitudes of the faults are unknown. It is important to remember that the fault mode concerns what model equations that are used and that the actual fault concerns what is happening to the real system which may show in the measured signals.

Fault Mode	Fault	Description
NF	-	No fault
F_1	f_1	Resistance offset, in AC483.
F_2	f_2	Resistance offset in DC485
F_3	f_3	Drift in voltage sensor E242

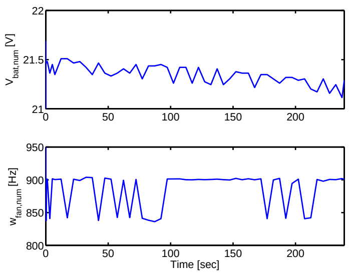
Table 5.4. The table defines the fault modes and their corresponding faults, as well as a brief description of the faults.



(a) algebraical algorithm



(b) $RelTol = 10^{-1}$, $AbsTol = 10^{-1}$



(c) $RelTol = 10^{1}$, $AbsTol = 10^{0}$

Figure 5.4. Simulation of the states testing the tolerance parameters $RelTol$ and $AbsTol$ for the numerical algorithm. The algebraical algorithm, in (a), is used as a reference. In (b) the parameters are too strict and the simulation is therefore unintentionally aborted at around 0.5 sec because the conditions can not be met. In (c) the simulation resembles that of the reference (a) and the parameters can therefore be accepted.

5.6.3 Fault Mode NF without Faults

The system is tested with fault mode NF and with no fault occurring. The simulation can be seen in Figure 5.5. The two algorithms look qualitatively rather similar. Both algorithms show a slight offset in residual r_2.

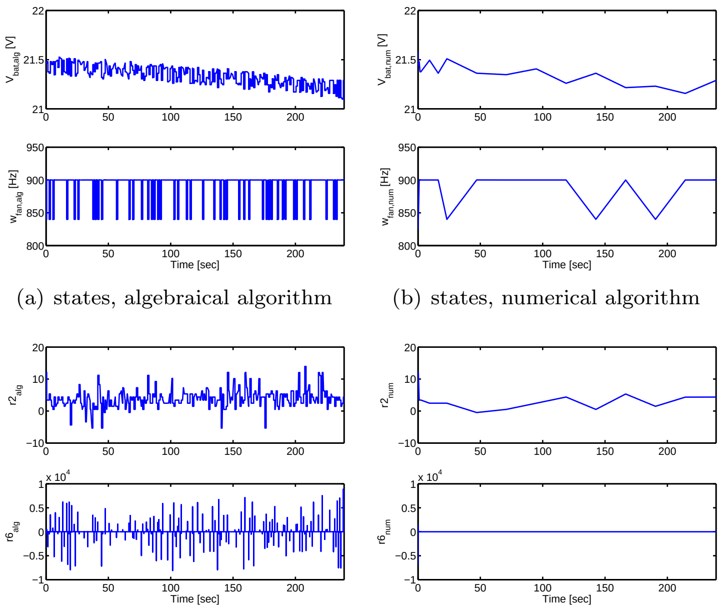
5.6.4 Fault Mode NF with Fault f_1

The system is tested with fault mode NF in combination with the fault f_1 defined in Table 5.4. The simulated states and two residuals can be seen in Figure 5.6

The fault can be seen in the state V_{bat}. The residual r_1 shows the fault quite clearly. In Figure 5.2 it can be seen that residual r_1 is one of the residual closest to the faulty component AC483. Residual r_2, which is also close and whose residual is not shown here, does however not show the fault clearly.

5.6.5 Fault Mode F_1 with Fault f_1

The variables in x_1 can not be algebraically solved in Mathematica for fault mode F_1. The linearization point is therefore calculated with an "ad hoc" method for



(a) states, algebraical algorithm (b) states, numerical algorithm

(c) residuals, algebraical algorithm (d) residuals, numerical algorithm

Figure 5.5. The states and two residuals are simulated in fault mode NF, with no fault introduced in the system. The two algorithms look similar but numerical algorithm has a longer step length in the simulation. In (a) and (b) the battery voltage V_{bat} decreases over time which is expected. The residual r_2 in (c) and (d) have a slight offset from zero which is not ideal to have.

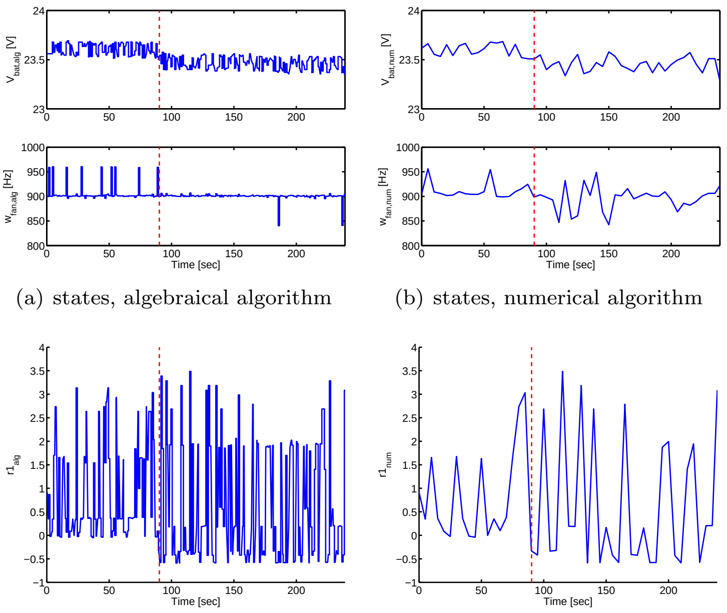
both the algebraical and the numerical algorithm. As the linearization point a vector with the value of one in every element is used. Plots from the simulation are shown in Figure 5.7. The fault f_1 is not compensated in residual r_1 in Figure 5.7(b).

5.6.6 Fault Mode NF with Fault f_2

The fault f_2 is a fault in the component DC485. The residual that should and does show the fault the most clearly is residual r_3. This is expected because residual r_3 checks the currents going from component EYE284 to component DC485. The simulation is plotted in Figure 5.8. In the plot the states are not very strongly affected by the fault but the residual r_3 shows very clearly when the fault is occurring. It would be very easy to detect this fault with thresholding with both the algebraical and the numerical algorithm.

5.6.7 Fault Mode F_2 with Fault f_2

The system is simulated in fault mode F_2 with the related fault f_2 occurring. f_2 is an offset fault of the resistance in component DC485 and the fault mode F_2 shall compensate for this kind of fault. The simulation is shown in Figure 5.9. The fault



(a) states, algebraical algorithm (b) states, numerical algorithm

(c) residual, algebraical algorithm (d) residual, numerical algorithm

Figure 5.6. The states and residual r_1 are simulated in fault mode NF, with fault f_1 introduced at 90.2 sec, dashed line. The state V_{bat} shows clear signs of the fault, especially with the algebraical algorithm, in (a). The residual r_1 in (c) and (d) shows a change in amplitude when the fault comes.

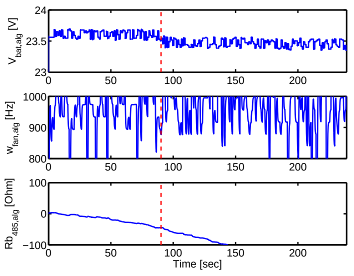
is compensated by assigning a state to the fault variable R_b which can be seen in (a) and (b) in the figure. In (c) and (d) the residual r_3, which is very sensitive to the fault, is presented. Both the algorithms react to the fault in residual r_3 but with the algebraical algorithm this is compensated very quickly and only a narrow spike is shown. With the numerical algorithm the compensation for the fault is much slower. The difference in the results of the algorithms is also seen clearly in the state R_b in (a) and (b) in the same figure.

5.6.8 Fault Mode NF with Fault f_3

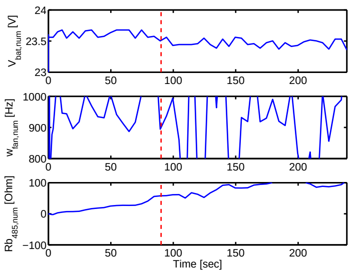
The system is simulated in fault mode NF, with fault f_3 which is a sensor fault. The simulations in Figure 5.10 show that the fault is visible for both algorithms.

5.6.9 Fault Mode F_3 with fault f_3

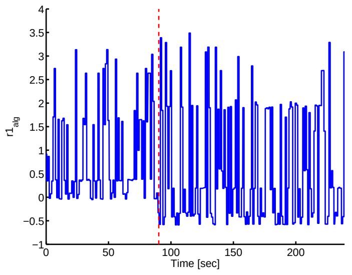
As with fault mode F_1, with fault mode F_3, x_1 can not be algebraically solved. An "ad hoc" linearization point is created with ones as coordinates. Plots from the simulation are shown in Figure 5.11 and in 5.12. It is curious that in Figure 5.11 the numerical algorithm compensates for the fault better than the algebraical



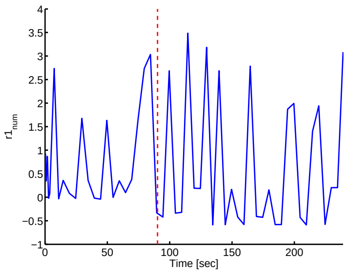
(a) states, algebraical algorithm



(b) states, numerical algorithm



(c) residual, algebraical algorithm



(d) residual, numerical algorithm

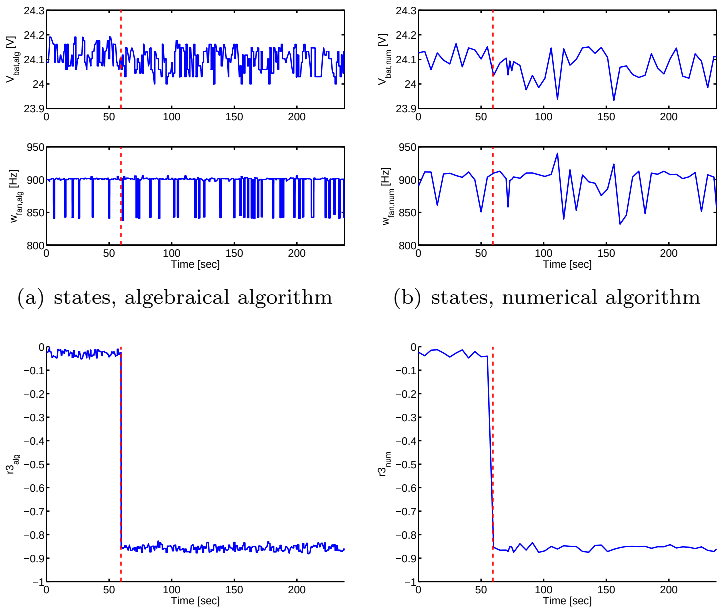
Figure 5.7. The states and residual r_1 are simulated in fault mode $F1$, with fault f_1 introduced at 90.2 sec, dashed line. The state ω suffers from the choice λ, but the choice is a compromise of overall getting good states. The state V_{bat} shows signs of the fault in (a) and (b). Why the state R_b behaves differently in (a) and (b) is unknown. The residual r_1 in (c) and (d) shows a change in amplitude when the fault comes which shows that the fault has not been fully compensated.

algorithm. Why this happens is unknown. The fault is however not compensated in all residuals, which is shown in Figure 5.12.

5.7 Variance Estimations for the Kalman Filter

When the Kalman Filter is created, the variance estimations of the states and the residuals are made and used in the matrices Q and R. In the other sections of this chapter the rough estimation, by using absolute values of the signals together with tuning of the parameter λ, has been used. In this section this method is compared with a more accurate estimation of the variances. In this method the variances are estimated by calculating the variances from simulated states and residuals. The matrices Q and R are however still dependent on the tuning parameter λ. The reason why this method has not been used previously, is that simulations are required and a first estimation or guess is needed anyway.

In Figure 5.13 and Figure 5.14, a comparisons between the presented estimation methods is made. A variance estimation of the states and residuals has been calculated from simulations of the algebraical algorithm. In general, the plots



(a) states, algebraical algorithm (b) states, numerical algorithm

(c) residual, algebraical algorithm (d) residual, numerical algorithm

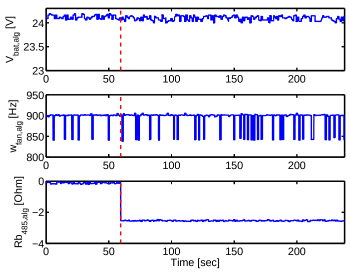
Figure 5.8. The states and two residuals simulated in fault mode NF, with fault f_2 introduced at 59.5 sec, dashed line. The states are relatively unaffected but the residual r_3 shows the fault very clearly in both algorithms in (c) and (d). It is possible in (d) to see that the reaction on the fault seem to happen just before the the fault time. This is caused by that the fault occures between two model evaluations.

from the two methods look similar. Some differences may be caused by different values of λ. The states from the numerical algorithm with the more accurate variance estimation in Figure 5.13(d) need different values of λ and the chosen value is a compromise which is why the state ω_{fan} has a big variance.

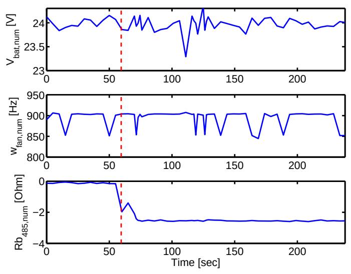
It is interesting to note that the Kalman filters with the more accurate variance estimation

$$
\begin{aligned}
K_{\text{alg}} &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 65 & 0 & 0 \end{pmatrix} \\
K_{\text{num}} &= \begin{pmatrix} 0.034 & -0.0011 & 0 & 0 & 0 & 0 & -0.002 & 0 & 0 \\ -18.194 & 0.618 & 0 & 0 & 0.110 & 0 & 1.170 & 0 & 0 \end{pmatrix}
\end{aligned} \tag{5.5}
$$

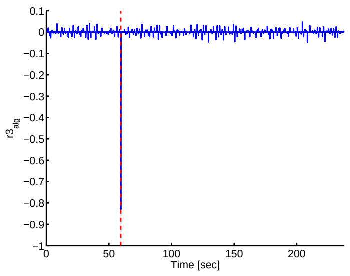
which is in (5.5) shown for the fault free fault mode, have some similarities to that of the of the Kalman filter shown in (5.4), with regards to what residuals are used in the feedback.



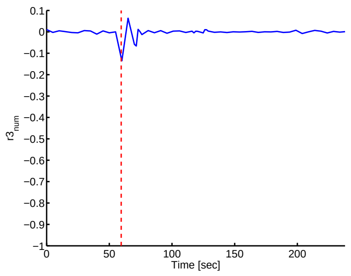
(a) states, algebraical algorithm



(b) states, numerical algorithm



(c) residuals, algebraical algorithm



(d) residuals, numerical algorithm

Figure 5.9. The states and two residuals simulated in fault mode F_2, with fault f_2 introduced at 59.5 sec, dashed line. The offset variable R_b is now assigned a state and the fault can be compensated. This is done successfully in both algorithms as can be seen in the state R_b in (a) and (b) and on the residual in (c) and (d). Note the very narrow spike in (c) at the time of the fault. In (d) the spike is shallower and broader.

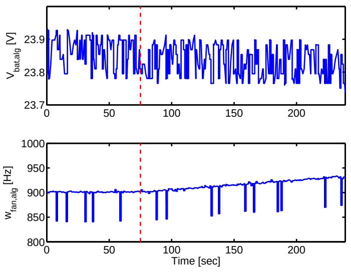
5.8 The Numerical Algorithm's Sensitivity to λ

In the simulations the parameter λ has been adjusted for the numerical algorithm to give simulation results similar to those of the the algebraical algorithm. In many cases it seems vital to make careful adjustment of λ to not get very distorted signals in the simulation. This is shown in Figure 5.15. Examples of values that have been used for the numerical algorithm are presented in Table 5.5 where the spread of the values is noteworthy. The algebraical algorithm is not as sensitive to λ and $\lambda = 10^5$ has been used as a standard value for this algorithm with the exceptions $\lambda = 10^{-4}$ for fault mode $F1$ and $\lambda = 10^{10}$ for fault mode $F3$.

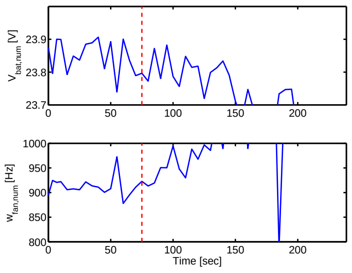
5.9 About Approximation in the Linearization

In the linearization in Section 4.2.1 an approximation is made as

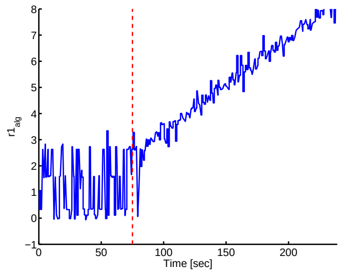
$$g(x, y) \approx \underbrace{g(x_0, y_0)}_{\approx 0} + M\Delta x + N\Delta y \approx M\Delta x + N\Delta y$$



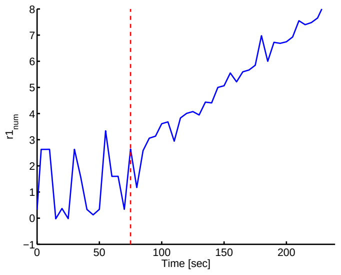
(a) states, algebraical algorithm



(b) states, numerical algorithm



(c) residuals, algebraical algorithm



(d) residuals, numerical algorithm

Figure 5.10. The states and residual r_3 simulated in fault mode NF, with fault f_3 introduced at 75 sec, dashed line. The states are fairly unaffected but the residual r_3, which is a residual for a sensor, clearly shows the expected drift in (c) and (d).

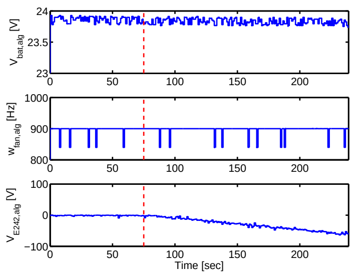
where the term $g(x_0, y_0)$ is assumed insignificant and is neglected. To see if this assumption is well made, a test quotient l is made as

$$l = \frac{\|g(x, y) - g(x_0, y_0)\|}{\|g(x, y)\|} \tag{5.6}$$

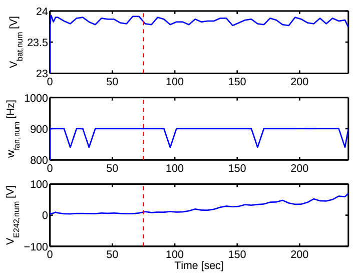
Which compares the signals from approximated system with the nonlinear system. When $g(x_0, y_0)$ is insignificant, the quotient is close to one. There is a function $g(x, y)$ for every state and residual with which the quotient l can be made. Histograms of l from a few selected states and residuals l from a simulation of fault mode NF without faults in the system are plotted in Figure 5.16. Many of the samples are somewhat close to one for the quotient l but there are deviations with values many times bigger than one.

In Figure 5.17 the quotient l is plotted over time. The signals do not show any obvious pattern but look rather stochastic.

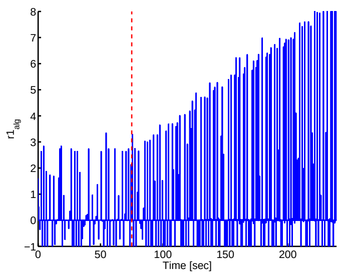
It seems that the decision to neglect the term $g(x_0, y_0)$ works well in many time steps in the simulation but not in all.



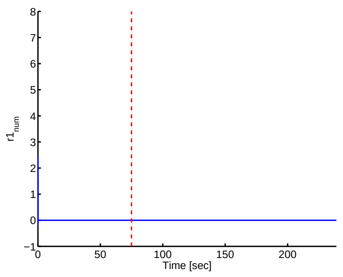
(a) states, algebraical algorithm



(b) states, numerical algorithm

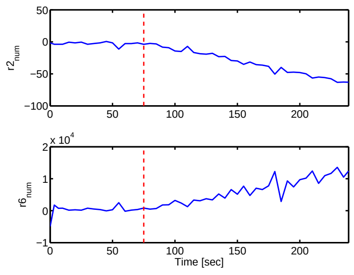


(c) residual, algebraical algorithm

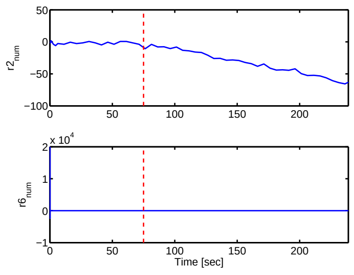


(d) residual, numerical algorithm

Figure 5.11. The states and residual r_3 simulated in fault mode $F3$, with fault f_3 introduced at 75 sec, dashed line. The third state in (a) and (b) should compensate for the fault so that the residuals in (b) and (d), are constant. The numerical algorithm compensates for the fault. The algebraical algorithm tries to compensate for the fault but fails.

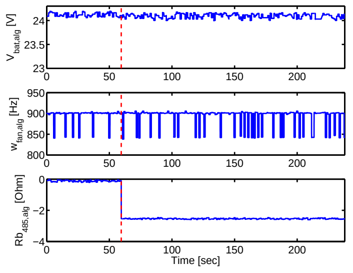


(a) residual, numerical algorithm, fault mode NF

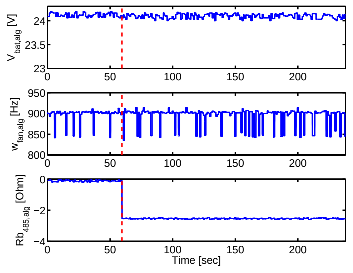


(b) residual, numerical algorithm, fault mode F_3

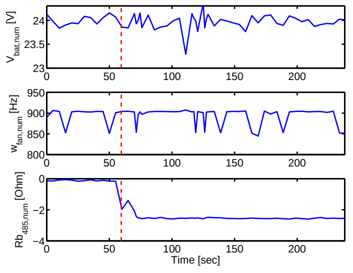
Figure 5.12. A residual in fault mode NF in (a) and fault mode F_3 in (b) with fault f_3 introduced at 75 sec. Only the numerical algorithm is shown here but the algebraical algorithm gives very similar results. The drift in residual r_2 is more or less the same despite the fault mode difference. In residual r_6 however, the fault is compensated with fault mode F_3 and is not detectable.



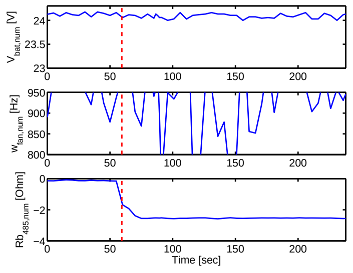
(a) states, algebraical algorithm



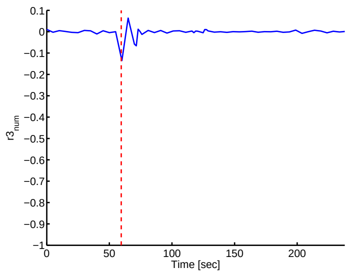
(b) states, algebraical algorithm, with the more accurate variance estimation



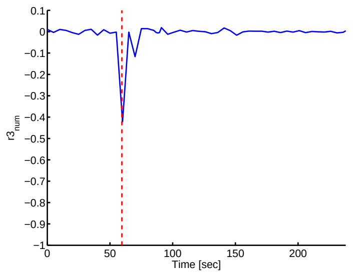
(c) states, numerical algorithm



(d) states, numerical algorithm, with the more accurate variance estimation

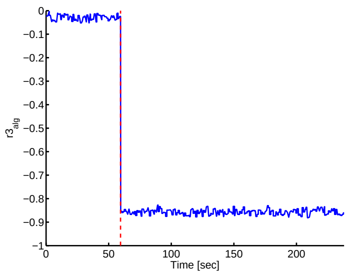


(e) residual r_3, numerical algorithm

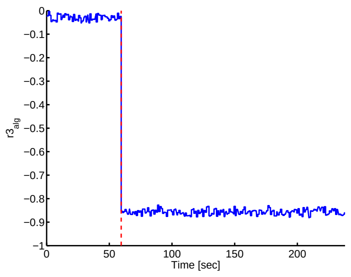


(f) residual r_3, numerical algorithm, with the more accurate variance estimation

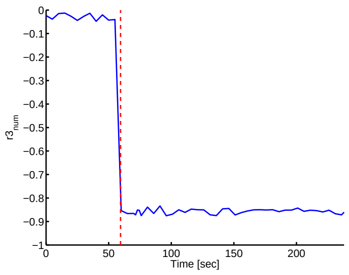
Figure 5.13. The states and a residual in fault mode F_2 with fault f_2 introduced at 59.5 sec. In (b), (d) and (f) the more accurate estimations of the variations of the states and residuals have been used. Note that the different variance estimation methods overall makes a small difference here. The ω_{fan} in (d) however shows a big variance caused by the choice of λ. The states need different values of λ and the chosen value in the simulation in (d) and (f) is a compromise where the states V_{bat} and and R_b have been prioritized. In the residuals (e) and (f) there is a noticeable difference where the more accurate estimation of the variance shows a more accurate result. In both simulations the effect from the fault is much smaller than it was without the fault compensation given by the fault mode.



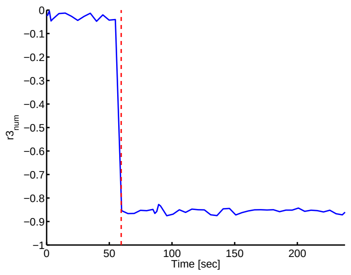
(a) residual r_3, algebraical algorithm



(b) residual r_3, algebraical algorithm, with the more accurate variance estimation



(c) residual r_3, numerical algorithm

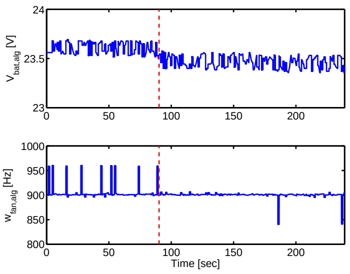


(d) residual r_3, numerical algorithm, with the more accurate variance estimation

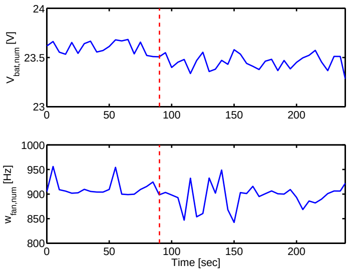
Figure 5.14. The residual r_3 in fault mode NF with fault f_2 introduced at 59.5 sec. In (b) and (d) the more accurate estimations of the variations of the states and residuals have been used. Note that the variance estimation does not make a big difference here.

Fault Mode	Fault	λ-Value
NF	-	10^{-1}
NF	f_1	10^{-1}
NF	f_2	10^{-1}
NF	f_3	10^{-2}
F_1	f_1	10^{-4}
F_2	f_2	10^{1}
F_3	f_3	10^{10}

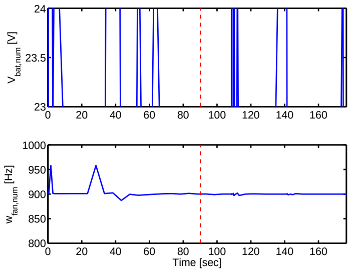
Table 5.5. The table shows the values of λ for simulations of combinations of fault modes and faults for the numerical algorithm. Note that the choice of λ also depends on the measurement series of the signals. The values should only be seen as examples of values used. It is interesting to note the spread.



(a) algebraical algorithm

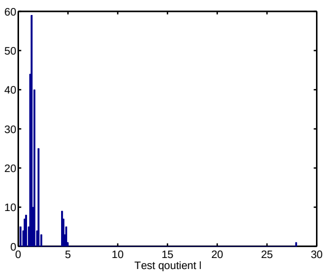


(b) numerical algorithm, $\lambda = 10^{-1}$

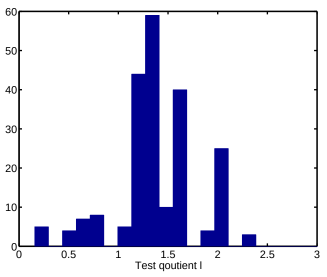


(c) numerical algorithm, $\lambda = 10^{4}$

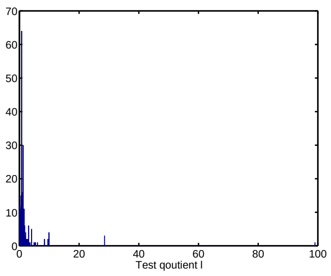
Figure 5.15. The plot shows simulated states for fault mode NF and with fault f_1 introduced at 90.2 sec. The influence of λ to the numerical algorithm is tested in (b) and (c) with the algebraical algorithm in (a) as a reference. Plot (b) looks similar to (a) but in (c) with a higher λ, the state V_{bat} is very distorted.



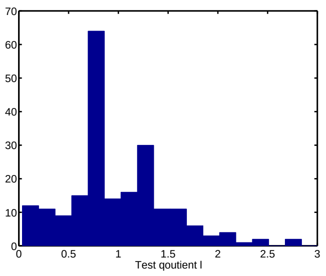
(a) state ω_{fan}



(b) state ω_{fan} zoomed in

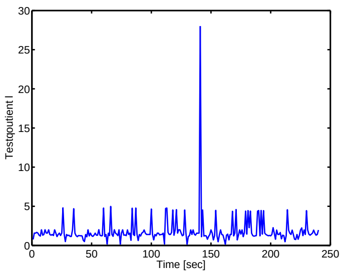


(c) residual r_3

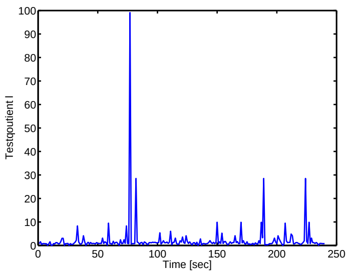


(d) residual r_3 zoomed in

Figure 5.16. Histograms of test quotient l from selected states and residuals. In both (c) and (d) the quotient is not clearly centered around the value one, but it is not far away.



(a) state ω_{fan}



(b) residual r_3

Figure 5.17. Time plots of test quotient l from selected states and residuals. The signals seem stochastic.

Chapter 6

Conclusions

From the results, presented in the previous chapter, a couple of conclusions can be drawn which are presented in this chapter.

6.1 Generation of Residual Equations

The generation of the residuals works as it is supposed to in many cases. There are problems with generating the equations for certain fault modes when x_1 can not be solved algebraically. The algorithms still work if the linearization points can be estimated without the solved x_1, as is the case in Section 5.6.5 and in Section 5.6.9.

When the equations in g_{alg} are sorted into g_{alg1} and g_{alg2} there is a certain randomness of what equations go where. The randomness is somewhat dependent on the order of which the equations are sorted in g_{alg}. The user has no real control over this in the current system. This do however affect where in the system the residuals will be located. When faults are to be isolated it can be important to have control over the choice of which equations are sorted into g_{alg1} and which are sorted into g_{alg2}. This has however not been focused upon in the project but is important to take note of.

6.2 Simulation Method

Both the algebraical and the numerical algorithm have worked quite satisfactory and some of the tested faults have been clearly visible for the bare eye. The algebraical algorithm shows the faults slightly more clearly than the numerical algorithm but both algorithms work pretty well at the simulation stage.

As shown in Section 5.8 the simulations with the numerical algorithm are in some cases quite sensitive to the parameter λ which may need to be retuned for each fault mode and/or in-data set. This can be a hindrance for the automatization of a diagnosis system because it needs to be tuned manually unless there is a way to automatize the tuning.

6.3 Algebraical Shortcuts

The algebraical shortcuts for the numerical algorithm makes the algorithm not completely numerical and it is reasonable to believe that such an implementation is slower in calculation time. There are two shortcuts that have been taken which are discussed separately below.

6.3.1 Linearization and Simulation Initialization Point

The algebraical shortcut used with the numerical algorithm, to find the linearization and simulation start point, described in Section 5.5.1, did affect the curves of the simulated states. It did however not dramatically affect the magnitude and the curve shapes had similarities. By using knowledge about the real system and the expected magnitudes of the signals it should be possible to make a system to automatically find good linearization points and simulation start points without any algebraical shortcuts when the system is running.

6.3.2 Reformulation of the Algebraic Constraints

The second algebraical shortcut is the reformulation of algebraical constraints. This problem might be hard to overcome. No other solution can be presented here but other programs or solvers for simulating DAEs may be options to try.

6.4 Estimation of Variances for the Kalman Filter

The rough estimation of the variances that has been used in the design of the Kalman filter seems to have worked. In the comparison with the more accurate but less automatic way of estimating the variances, in Section 5.7, the rougher method did well. Different estimations do however give slightly different results for the numerical algorithm and an improved estimation may still be important for achieving higher accuracy in the simulations.

6.5 Approximation in the Linearization

The approximation that is made in the algorithm in Section 4.2.1 has been evaluated in Section 5.9. The evaluation shows that the approximation is not unfounded but also that it is possible that it can be problematic for the simulations. To draw a more precise conclusion about the consequences of the approximation would need a deeper analyzation of the problem than has been made in the thesis.

Chapter 7

Future Work

During the course of the work new questions have arisen and some old question have not been of priority, which are never the less of interest. These are left as future work which are questions that can be analyzed in new projects.

7.1 Improved Model

How good the fault detection is depends on how good the models of the system are. The work in this master thesis has not tried to get as good residuals as possible but rather to get residuals that are good enough to see if the method of automatic residual generation works. It would however be very interesting to see how much better the residuals can get with more work put into the modeling of the ADAPT-Lite system.

7.2 Multiple Faults

Only single faults have been tested. It would be interesting to see what happens when there are multiple faults occurring and when fault modes for these multiple faults are used. It would be interesting to see how well the faults are compensated for by the fault modes, and too see what happens with the detectability of new faults when the system is already in a fault mode for multiple faults. It would be especially interesting to see how the detectability of faults is affected when decoupling dynamic faults. It is possible to speculate that there may be a risk that the states that compensates for a dynamic fault also compensates for other faults unintentionally.

7.3 Automatic Model Parameter Estimation

In this master thesis the process for estimating model parameters has been done manually. It would be of great interest if this could be fully automated so that the residual equations or the model equations, the unknown parameters and sample data work as in-parameters in a program that automatically calculates the unknown parameters. This would save a lot of time when the analyzed system is changed.

7.4 Automatic Estimation of λ

The tuning parameter λ, used for the design of the matrices Q and R in the Kalman filter design, has been tuned manually while looking at the resulting plots. The spread in the λ values for different simulations has been big. This is a hindrance for an automatic process. It would be very good if an estimation of λ could be made automatic.

7.5 Improved Numerical Simulation Method

The numerical simulations of the system have given results but are there other simulation tools that can simulate more efficient and better? The simulations have been run in MATLAB but maybe another programming language or other algorithms could be better suited for efficient simulations of DAE's.

7.6 Real Time Implementation

In this thesis the results have only been simulated offline with pre-collected measurement data from sensors. To do an implementation, that runs fully online in real time, of the diagnostic program, demands for additional work to be done. Other programming languages, for instance C, may be needed. It will also need logic for handling the online switching between fault modes which must also be executed in real time.

Bibliography

[1] Uri N. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations.* SIAM, Philadelphia, 1998.

[2] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* Elsevier Science Publishers, New York, 1989.

[3] J. de Kleer, Alan K. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56:197–222, 1992.

[4] T. Glad and L. Ljung. *Reglerteori: flervariabla och olinjära metoder.* Studentlitteratur, 2003.

[5] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[6] Mattias Krysander, Jan Åslund, and Mattias Nyberg. An efficient algorithm for finding minimal over-constrained sub-systems for model-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 38(1), 2008.

[7] Jeffrey C. Lagarias, James A. Reeds, Margaret H. Wright, and Paul E. Wright. Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9:112–147, 1998.

[8] Mattias Nyberg. A fault isolation algorithm for the case of multiple faults and multiple fault types. In *Proceedings of IFAC Safeprocess'06*, Beijing, China, 2006.

[9] Mattias Nyberg and Erik Frisk. *Model Based Diagnosis of Technical Processes.*

[10] R. Penrose. A generalized inverse for matrices. *Proceedings of the Cambridge Philosophical Society*, 51:406–413, 1955.

[11] PHM Society. second international diagnostic competition, January 2011. `https://www.phmsociety.org/competition/dxc/10`.

[12] The Mathworks, Natick, MA. Matlab V7.7 (R2008b), 2008.

[13] Chien-Fu Wu. Asymptotic theory of nonlinear least squares estimation. *The Annals of Statistics*, 9(3), 1981.

Appendix A

Component Models of ADAPT-Lite

In this chapter the component models used for the ADAPT-Lite circuit are shown.

To save space the definitions of currents, voltages and potentials that are used through out this chapter are defined in Table A.1.

I	DC current through the component
V	voltage over the component
V_p	potential at the PPin
V_n	potential at the NPin
$i_{p,Re}$	real part of current through the PPin
$i_{p,Im}$	imaginary part of current through the PPin
$i_{n,Re}$	real part of the current though the NPin
$i_{n,Im}$	imaginary part of the current though NPin
v_{Re}	real part of the voltage over the component
v_{Im}	imaginary part of the voltage over the component
$v_{p,Re}$	real part of the potential at the PPin
$v_{p,Im}$	imaginary part of the potential at the PPin
$v_{n,Re}$	Real part of the potential at the NPin
$v_{n,Re}$	imaginary potential at the NPin

Table A.1. Definition of currents, voltages and potentials of the component models.

A.1 Battery

The battery is modelled as

$$\begin{aligned} \frac{dV_{\text{bat,int}}}{dt} &= K_{\text{bat}} I, \\ V_n &= V_{\text{int}} + R_{\text{bat}} I, \end{aligned} \tag{A.1}$$

where $V_{bat,int}$ is the internal voltage, R_{bat} is the internal resistance and K_{bat} is a proportionality constant. The connection points and constant parameters can be seen in Table A.2.

PPin	-
NPin	I, V_n
constants	K_{bat}, R_{bat}

Table A.2. The table shows connection points and the constant parameters of the battery.

A.2 Circuit Breaker and Relay

The circuit breaker and the relay are modelled as

$$\begin{aligned} I &= 0;\ [T, SO], \\ V &= RI, \\ V_p &= V_n + V, \end{aligned} \tag{A.2}$$

where R is the resistance. The connection points and constant parameters can be seen in Table A.3.

PPin	I, V_p
NPin	I, V_n
constants	R

Table A.3. The table shows the connection points and the constant parameters of the circuit breaker and the relay.

A.3 Complex Circuit Breaker and Complex Relay

The complex circuit breaker and the complex relay is modelled as

$$\begin{aligned} i_{Re} &= 0;\ [T, SO] \\ i_{Im} &= 0;\ [T, SO] \\ v_{Re} &= Ri_{Re}, \\ v_{Im} &= Ri_{Im}, \\ v_{Re} &= v_{p,Re} - v_{n,Re}, \\ v_{Im} &= v_{p,Im} - v_{n,Im}, \end{aligned} \tag{A.3}$$

where R is the resistance through the component. The connection points and constant parameters can be seen in Table A.4.

PPin	$i_{p,Re}$, $i_{p,Im}$, $v_{p,Re}$, $v_{p,Im}$
NPin	$i_{n,Re}$, $i_{n,Im}$, $v_{n,Re}$, $v_{n,Im}$
constants	R

Table A.4. The table shows the connection points and constant parameters of the complex circuit breaker and the complex relay.

A.4 Fan

The fan is modelled as

$$\begin{aligned} v_{Re} &= z_{Re}i_{Re} - z_{Im}i_{Im};\ [OK], \\ v_{Im} &= z_{Im}i_{Re} + z_{Re}i_{Im};\ [OK], \\ \frac{d\omega}{dt} &= k_t(k_0\sqrt{P} - \omega);\ [OK], \\ P &= v_{Re}i_{Re} + v_{Im}i_{Im}, \\ v_{Re} &= v_{p,Re} - 0, \\ v_{Im} &= v_{p,Im} - 0, \end{aligned} \tag{A.4}$$

where z_{Re} and z_{Im} are the real and imaginary parts of the resistance, P is the effect used, ω is the fan speed and k_0 and k_t are constants. The connection points and constant parameters can be seen in Table A.5.

PPin	i_{Re}, i_{Im}, $V_{p,Re}$, $V_{p,Im}$
NPin	-
constants	z_{Re}, z_{Im}, k_0, k_t

Table A.5. The table shows the connection points and the constant parameters of the fan.

A.5 Inverter

The inverter is modelled as

$$\begin{aligned} (V_p > 22) &= (v_{n,Re} > 120);\ [OK], \\ v_{n,Re} i_{n,Re} &= V_p I_p - P_0;\ [OK], \\ V_p &> 22;\ [FO], \\ v_{n,Re} &< 120;\ [FO], \\ v_{n,Im} &= 0. \end{aligned} \tag{A.5}$$

where I_p is the DC current through the PPin and P_0 is the effect used in the inverter. The connection points and constant parameters can be seen in Table A.6.

PPin	I_p, V_p
NPin	$i_{n,Re}$, $i_{n,Im}$, $v_{n,Re}$, $v_{n,Im}$
constants	P_0

Table A.6. The table shows the connection points and the constant parameters of the inverter.

A.6 Real Load

The real load is modelled as

$$\begin{aligned} R_b &= 0;\ [OK], \\ V &= (R + R_b)I, \\ V_p &= V_n + V, \\ V_n &= 0, \end{aligned} \tag{A.6}$$

where R is the nominal resistance, R_b is an offset resistance, Note that $V_n = 0$ because this component does not connect the NPin in the ADAPT-Lite circuit. The connection points and constant parameters can be seen in Table A.7.

PPin	V_p, I
NPin	-
constants	R

Table A.7. The table shows the connection points and the constant parameters of the real load.

A.7 Real AC Load

The real AC load is modelled as

$$\begin{aligned} R_b &= 0;\ [OK], \\ \frac{dR_b}{dt} &= 0;\ [OS], \\ v_{Re} &= (R + R_b)i_{Re}, \\ v_{Im} &= (R + R_b)i_{Im}, \\ v_{Re} &= v_{p,Re} - 0, \\ v_{Im} &= v_{p,Im} - 0, \end{aligned} \tag{A.7}$$

where R is the nominal resistance, R_b is an offset resistance, Note that $v_{Re} = v_{pRe} - 0$ and $v_{Im} = v_{p,Im} - 0$ because the NPin is not connected in the ADAPT-Lite circuit. The connection points and constant parameters can be seen in Table A.8. $[OK]$ stands for no fault, $[OS]$ for a offset fault.

PPin	i_{Re}, i_{Im}, $V_{p,Re}$, $V_{p,Im}$
NPin	-
constants	R

Table A.8. the table shows the connection points and the constant parameters of the real AC load.

A.8 Sensors

The sensors are modelled as

$$\begin{aligned} z &= y + b\ [OK] \\ b &= 0\ [OK] \\ \frac{db}{dt} &= 0\ [B] \end{aligned} \tag{A.8}$$

where y is the measured signal without bias, b is the bias, and z is what the sensor shows.

The sensors only have one pin that is connected to where the sensor is measuring. $[OK]$ stands for no fault, $[B]$ for a bias fault.