Institutionen för systemteknik
Department of Electrical Engineering

**Examensarbete**

# Development of Methods for Automatic Design of Residual Generators

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan i Linköping
av

**Carl Svärd & Henrik Wassén**

LITH-ISY-EX--06/3888--SE

Linköping 2006


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

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

# Development of Methods for Automatic Design of Residual Generators

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan i Linköping
av

**Carl Svärd & Henrik Wassén**

LITH-ISY-EX--06/3888--SE

Handledare: **Erik Frisk**
ISY, Linköpings universitet
**Mattias Nyberg**
Scania CV AB
**Gustav Arrhenius**
Scania CV AB

Examinator: **Erik Frisk**
ISY, Linköpings universitet

Linköping, 14 December, 2006

**Avdelning, Institution**
Division, Department

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

**Datum**
Date

2006-12-14

**Språk**
Language

☐ Svenska/Swedish
☒ Engelska/English
☐ _____

**Rapporttyp**
Report category

☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport
☐ _____

**ISBN**
—

**ISRN**
LITH-ISY-EX--06/3888--SE

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

**URL för elektronisk version**
http://www.vehicular.isy.liu.se
http://www.ep.liu.se

**Titel**
Title

Utveckling av metoder för automatisk design av residualgeneratorer
Development of Methods for Automatic Design of Residual Generators

**Författare**
Author

Carl Svärd & Henrik Wassén

**Sammanfattning**
Abstract

Legislation requires substantially lowered emissions and that all trucks manufactured are equipped with an On-Board Diagnosis (OBD) system. One approach for designing an OBD system is to use model based diagnosis and residual generation. At Scania CV AB, a method for automatic design of a diagnosis system from a model has been developed but there are still possibilities for improvements to get more and better residual generators. The main objective of this thesis is to analyze and improve the existing method.

A theoretic outline of two methods using different causality assumptions is presented and the differences are analyzed and discussed. Stability of residual generators is analyzed and a method for constructing stable residual generators and its consequences for the diagnosis system is presented.

Methods using integral and derivative causality are found not to be equivalent for all dynamic systems, resulting in that a diagnosis system utilizing both methods would be preferred for detectability reasons. A stable residual generator can be constructed from an unstable residual generator. The method for stabilizing a residual generator affects the fault sensitivity of the residual generator and the fault detectability properties of the diagnosis system.

**Nyckelord**
Keywords

model based diagnosis, residual generator, residual generation, structural analysis, causality, stability

# Abstract

Legislation requires substantially lowered emissions and that all trucks manufactured are equipped with an On-Board Diagnosis (OBD) system. One approach for designing an OBD system is to use model based diagnosis and residual generation. At Scania CV AB, a method for automatic design of a diagnosis system from a model has been developed but there are still possibilities for improvements to get more and better residual generators. The main objective of this thesis is to analyze and improve the existing method.

A theoretic outline of two methods using different causality assumptions is presented and the differences are analyzed and discussed. Stability of residual generators is analyzed and a method for constructing stable residual generators and its consequences for the diagnosis system is presented.

Methods using integral and derivative causality are found not to be equivalent for all dynamic systems, resulting in that a diagnosis system utilizing both methods would be preferred for detectability reasons. A stable residual generator can be constructed from an unstable residual generator. The method for stabilizing a residual generator affects the fault sensitivity of the residual generator and the fault detectability properties of the diagnosis system.

# Sammanfattning

Lagkrav kräver väsentligt sänkta emissionsnivåer och att alla tillverkade lastbilar är utrustade med ett system för On-Board Diagnosis (OBD). Ett sätt att konstruera ett OBD system är att använda modellbaserad diagnos och residualgenerering. På Scania CV AB har en metod för automatisk konstruktion av ett diagnossystem utifrån en modell utvecklats, men det finns utrymme för bättringar som leder till att fler och bättre residualgeneratorer konstrueras. Huvudsyftet med examensarbetet är att analysera och förbättra den existerande metoden.

En teoretisk beskrivning av två metoder som använder sig av olika kausalitet presenteras och skillnaderna analyseras och diskuteras. Stabiliteten hos residualgeneratorer analyseras och en metod för att konstruera stabila residualgeneratorer och dess konsekvenser för diagnossystemet presenteras.

Metoder som använder sig av integrerande respektive deriverande kausalitet visar sig inte vara ekvivalenta för alla dynamiska system, vilket resulterar i att ett diagnossystem som använder sig av båda kausaliteterna är att föredra i ett diagnossystem med avseende på detekterbarhet. En stabil residualgenerator kan konstrueras från en instabil residualgenerator. Metoden för att stabilisera en residualgenerator påverkar felkänsligheten hos residualgeneratorn och feldetekterbarheten hos diagnossystemet.

# Acknowledgments

First of all, we would like to thank our supervisors at Scania CV AB, Mattias Nyberg for all long inspiring and fruitful discussions and Gustav Arrhenius for all help and valuable input regarding our report. We would like to thank Erik Frisk, our excellent supervisor and examiner at Linköping University for interesting and stimulating discussions and for always taking his time to support us during our work. The staff at NED deserves a thank for their welcoming and open-minded work environment and NMCE for a pleasant stay at their department.

We would like to thank our fellow master thesis workers, Jens Gunnar Molin, Joakim Hansen and Rikard Falkeborn for all motivating and humorous discussions regarding our projects and everything else from sports and spiders to economics and rocket-science. Klas Håkansson and Mikael Johansson, master thesis workers and our neighbors at the department, also deserves a thank for helping us hold the lunch and coffee-schedule and a relaxing and somehow fuzzy environment. Finally, our families deserves a big thank for always supporting and guiding us in both good and bad times!

*Carl Svärd and Henrik Wassén*
Södertälje, December 2006

# Contents


**I Introduction and Theory** **1**

**1 Introduction** **3**
1.1 Background . . . 3
1.2 Existing Work . . . 4
1.3 Objectives . . . 4
1.4 Overview . . . 4
1.5 Contributions . . . 5
1.6 Target Group . . . 5

**2 Background Theory** **7**
2.1 Graph Theory and Structural Analysis . . . 7
2.1.1 Structural Model . . . 7
2.1.2 Matchings . . . 9
2.1.3 Directed Graph Associated with a Matching . . . 9
2.1.4 Canonical Decomposition . . . 10
2.2 Stability of Linear and Non-Linear Systems . . . 11
2.2.1 Stability of Linear Systems . . . 11
2.2.2 Stability of Non-Linear Systems . . . 12
2.3 Observer Theory . . . 13
2.3.1 Linear Observers and Eigenvalue Assignment . . . 14
2.3.2 The Kalman Filter . . . 15
2.3.3 Non-linear Kalman Filter . . . 16

**3 Model Based Diagnosis** **19**
3.1 Diagnostic Tests . . . 19
3.1.1 Analytical Redundancy . . . 20
3.2 Residual Generators . . . 20
3.2.1 Consistency Relations . . . 20
3.3 Faults . . . 22
3.3.1 Fault Detectability and Sensitivity . . . 22
3.4 The Procedure of Residual Generation . . . 22

II Methods for Residual Generation 25

4 Residual Generation with The Scania Method 27
4.1 Extract Analytical Model Equations from the Simulink Model . . . 27
4.2 Transform the Analytical Model to an SM . . . 28
4.3 Extract all MSO sets in the SM . . . 28
4.4 Find Residual Generators in the MSO Sets . . . 29
4.4.1 Invertibility Properties . . . 29
4.4.2 Bipartite Matchings and Computation Sequence . . . 29
4.5 Structure of the Residual Generators . . . 32

5 Residual Generation with The Lille Method 35
5.1 Extending the Model . . . 35
5.2 Transform the Analytical Model to an SM . . . 36
5.3 Find Residual Generators in the SM . . . 36
5.3.1 Differential Loop . . . 38

6 Analysis of Methods for Residual Generation 41
6.1 Integral Causality . . . 41
6.2 Equivalence of Methods . . . 44
6.3 Comparison of Methods . . . 44
6.3.1 Equivalence of Methods with same Causality . . . 45
6.3.2 Methods with different Causality . . . 46
6.3.3 Some Examples . . . 48
6.4 Improvements of the Scania Method . . . 52

III Improvements and Evaluation of the Scania Method 53

7 Solving Algebraic Loops 55
7.1 Algebraic Loops . . . 55
7.2 Algebraic Loops in the Engine Model . . . 57
7.3 Methods for Solving Algebraic Loops . . . 58
7.3.1 Analytical Solution . . . 58
7.3.2 Iterative Solution . . . 59
7.3.3 Mapping . . . 59
7.3.4 Step-Through Solution . . . 59

8 Constructing Stable Residual Generators 63
8.1 Stability of Residual Generators . . . 63
8.1.1 Causes of Instability . . . 64
8.1.2 Residual Stability Investigation . . . 66
8.2 Stabilizing Residual Generators utilizing Observer Theory . . . 67
8.2.1 Design Method . . . 69
8.2.2 Stability Analysis . . . 72

**9 Analysis of Stabilized Residual Generators** **75**
9.1 Derivation of Transfer Functions . . . . . 75
9.2 Some Properties in the Frequency Domain . . . . . 76
9.3 Detectability Analysis . . . . . 79
9.3.1 Static Gains . . . . . 79
9.4 Discussion . . . . . 82

**10 Evaluation of Stabilized Residual Generators** **83**
10.1 Residual Generator 1 . . . . . 83
10.1.1 Implementation in Simulink . . . . . 84
10.1.2 Design of Stable Residual Generators . . . . . 84
10.1.3 Fault-Free Simulation . . . . . 89
10.1.4 Simulation with Gain Faults . . . . . 93
10.1.5 Simulation with Bias Faults . . . . . 94
10.2 Residual Generator 2 . . . . . 98
10.2.1 Design of Stable Residual Generators . . . . . 98
10.2.2 Fault-Free Simulation . . . . . 99
10.2.3 Simulation with Faults . . . . . 99

**11 Conclusions** **103**

**Bibliography** **105**

# Part I

# Introduction and Theory

# Chapter 1

# Introduction

This master thesis was performed at Scania CV AB in Södertälje in collaboration with the department of Electrical Engineering, division of Vehicular Systems, at Linköpings University. Scania is a worldwide manufacturer of heavy duty trucks, buses and engines for marine and industrial use. The work was carried out at the department for diagnosis, NED, which is responsible for the On-Board Diagnosis (OBD) software.

## 1.1 Background

New stricter emission legislations, especially in the European Union, has made truck manufactures face new challenges. Besides requirements of substantially lowered emissions, the laws require that all heavy duty trucks manufactured are equipped with an OBD system. To keep emissions below legislation demands, sensors and actuators in the truck engine must be supervised and diagnosed continuously with the OBD system.

One approach for designing an OBD system, is to use model based diagnosis. From a model of the process to be diagnosed, tests are constructed which run in real-time in an engine control unit (ECU) located on the truck. A typical model based diagnostic test is to check if a, so called, residual is within some limits. A residual, computed by a residual generator, is often a comparison between a measured quantity and a computed value based on a model of the process. In general, model based tests are constructed by engineers with knowledge of the process and it may be a time-consuming and error-prone task. Even for a small change of the engine, the diagnosis system may have to be redesigned.

At Scania CV AB, a method for automatic design of diagnosis systems has been developed. The method extracts overdetermined subsystems in a non-linear model utilizing structural analysis. Residual generators based on these subsystems are then created and evaluated automatically. For a specific engine model, the method found 3401 possible residual generators. After evaluation, all residual generators but 42 were discarded due to e.g. non-invertability and instability. To be able

to design a diagnosis system with satisfactory properties, regarding e.g. isolation and detection, a larger selection of residual generators is desirable.

## 1.2 Existing Work

The work behind the method for automatic design of a diagnosis system has been carried out in three master thesis projects, all performed at Scania CV AB. In [4], algorithms for transforming a Simulink model to an analytical equation system and finding all overdetermined subsystems in equation systems were presented. A more efficient algorithm for finding overdetermined subsystems is presented in [13]. In [11] a method for automatic construction of model based diagnostic test based on overdetermined subsystems was developed. In [3] algorithms for residual evaluation and test selection were presented.

## 1.3 Objectives

The main objective of this thesis is to analyze and improve the existing method for automatic design of diagnosis systems, previously developed at Scania CV AB. The focus lies on the part of the procedure where residual generators are extracted from overdetermined subsystems. The main objective can be divided into the following parts

- The existing method shall be analyzed and described from a theoretically point of view.
- The existing method shall be compared with other methods for residual generation.
- Improvements, resulting in more and better residual generators, shall be done.

## 1.4 Overview

**Part I:** Presents the background, objectives and main contributions of the thesis and introduces the reader to important concepts of model based diagnosis.

**Part II:** Presents outlines of two different methods for constructing residual generators and analyzes the most obvious differences. Possible improvements of the method used at Scania CV AB is discussed and suggested.

**Part III:** Carries out some of the suggested improvements. Methods for solving algebraic loops and constructing stable residual generators are presented. Properties of a stabilized residual generator and its consequences for the diagnosis system is analyzed.

## 1.5 Contributions

The main contributions of this thesis are

**Chapter 4-5:** A theoretic outline of the methods used at Scania CV AB and Université Lille.

**Chapter 6:** An analysis of equivalence of methods for residual generation with the same and different causality approaches.

**Chapter 7:** Methods for solving algebraic loops.

**Chapter 8:** Definitions and an analysis of stability of residual generators. A method for designing stable residual generators that utilizes observer theory.

**Chapter 9:** An analysis of diagnostic related properties in the time and frequency domains of stable residual generators designed with the method in Chapter 8.

**Chapter 10:** An evaluation of stable residual generators designed with the method in Chapter 8.

## 1.6 Target Group

The target group for this thesis is undergraduate and graduate engineering students and employees at Scania CV AB. Knowledge in model based diagnosis, control theory and structural analysis is preferred for deeper understanding of some parts.

# Chapter 2

# Background Theory

This chapter will give a brief introduction to important concepts and the field of theories used in the rest of the thesis.

## 2.1 Graph Theory and Structural Analysis

Today there are basically two different ways of constructing diagnosis systems using model based diagnosis, one based on control theory and another based on AI. A common framework, known as BRIDGE, has been discussed in a group of researcher from both fields and it is presented in [19]. In this thesis, the methods are based on control theory and structural analysis will be described as it is an important tool in the methods.

### 2.1.1 Structural Model

The idea of structural models is to only consider the variables present in an equation, instead of considering the whole analytical expression. Consider the equation

$$e : f(x, z) = 0$$

where $x$ are the unknown variables, $z$ the known variables and $f$ the analytical expression combining them. The structural model corresponding to the equation $e$ does only contain the information about what variables $x$ and $z$ that are included in the equation and the analytical expression $f$ is ignored. A structural model can be represented as a biadjacency matrix or a bipartite graph.

A dynamic system consists of differentiated unknown variables, which can be treated in two different ways in a structural model. The differentiated variables, $\dot{x}$ and $x$, can be treated as separated variables and the model is then called a differentially separated structural model. The other type of structural model is called differentially lumped structural model, where the differentiated variables, $\dot{x}$ and $x$, are treated as structurally the same variable. From here, only differentially lumped structural models are considered in this thesis and hence will only be referred to as a structural model or shortly an SM.

Structural analysis is an important tool for identifying overdetermined subsystems in a model. Let $M$ denote a set of equations and $X$ the set of unknown variables present in the equations $M$.

**Definition 2.1** *The set $M$ is **structurally overdetermined (SO)** if $|M| > |X|$, **justdetermined** if $|M| = |X|$ and **underdetermine**d if $|M| < |X|$.*

One specific type of SO sets with only one more equation than unknown variables are of special interest. This type of subsystems is called MSO sets and are formally defined as follows.

**Definition 2.2 (Minimal Structurally Overdetermined)** *A structurally overdetermined set is a **Minimal Structurally Overdetermined (MSO)** set if none of its proper subsets are structurally overdetermined.*

### Bipartite Graph

One way of representing an SM is with a bipartite graph.

**Definition 2.3 (Bipartite Graph)** *A **bipartite graph** $G = (U \cup V, \bar{A})$ consists of two sets, a vertex set $\{U, V\}$, where $U \cap V = \emptyset$, and an edge set $\bar{A} \subseteq U \times V$, where*

$$(u_i, v_j) \in \bar{A}, \text{ if } v_j \in V \text{ is connected to } u_i \in U.$$

The set $U$ is here seen as the set of equations in the analytical model and the set $V$ is the set of variables. An edge in the bipartite graph represents the existence of a variable in the equation with which it is connected.

Consider the analytical equation system

$$\begin{array}{lrcl} e_1: & \dot{x}_1 & = & f(x_1, u) \\ e_2: & 0 & = & g(x_1, x_2) \\ e_3: & y & = & h(x_2). \end{array} \tag{2.1}$$

Figure 2.1 shows the bipartite graph of the SM for the equation system (2.1).

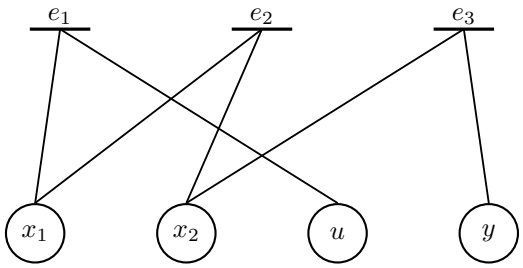



**Figure 2.1.** The bipartite graph of the SM corresponding to the equation system (2.1).

A graph $H = \left(V_H, \bar{A}_H\right)$ is a subgraph of $G = \left(V_G, \bar{A}_G\right)$ if $V_H \subseteq V_G$ and $\bar{A}_H \subseteq \bar{A}_G$. This is denoted with $H \subseteq G$. If equality does not hold $H$ is the proper subgraph of $G$ and it is denoted $H \subset G$. More about bipartite graphs can be found in [1].

### Biadjacency Matrix

A biadjacency matrix is another way of representing a structural model. For a biadjacency matrix to be valid, the vertex set has to consist of two disjoint sets. The SM corresponding to the analytical equation system (2.1) can be represented by the following biadjacency matrix.

| Equation | Unknown | | Known | |
|---|---|---|---|---|
| | $x_1$ | $x_2$ | $u$ | $y$ |
| $e_1$ | X | | X | |
| $e_2$ | X | X | | |
| $e_3$ | | X | | X |

If a variable is present in an equation, it is denoted with 'X' in the biadjacency matrix and represents an edge in a bipartite graph. 'X':s in the same row share the equation vertex and in the same column share the variable vertex.

## 2.1.2 Matchings

Matchings in a structural model can be made in both bipartite graphs and biadjacency matrices.

**Definition 2.4 (Matching)** *A **matching**, $\Gamma \subseteq \bar{A}$, is a set of edges in a bipartite graph $G = (U \cup V, \bar{A})$ such that no two edges have common vertices.*

An edge in a bipartite graph is represented with an 'X' in a biadjacency matrix, hence a matching in a biadjacency matrix is a collection of 'X':s such that no two 'X':s in the matching are in the same row or column.

**Definition 2.5 (Complete Matching)** *A matching is a **complete matching** with respect to a vertex set, if the matching covers all vertices in the set.*

**Definition 2.6 (Perfect Matching)** *A matching is a **perfect matching** of bipartite graph $G$, if the matching covers every vertex of $G$.*

The definitions of complete and perfect matchings are here made since the properties it hold will be important in this thesis.

## 2.1.3 Directed Graph Associated with a Matching

A bipartite graph is a directed graph[1] if all edges, $\epsilon \in \bar{A}$, has a direction. A matching $\Gamma$, of a bipartite graph can introduce a direction to the bipartite graph with which it is associated. In this thesis, an edge $(u_i, v_i) \in \Gamma$ introduces a direction of the edge from the equation $u_i$ to the variable $v_i$. An edge $(u_j, v_j) \notin \Gamma$ introduces an edge with the opposite direction.

Consider the bipartite graph in Figure 2.1 associated with the matching $\Gamma = \{(e_1, x_1), (e_2, x_2)\}$. The corresponding directed graph is shown in Figure 2.2.

[1] Referred to as oriented graph in [2].

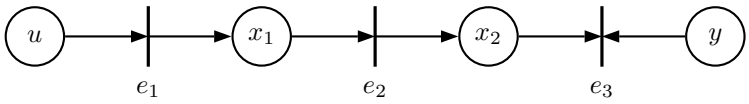



**Figure 2.2.** The corresponding directed graph from the bipartite graph in Figure 2.1 associated with the matching $\Gamma = \{(e_1, x_1), (e_2, x_2)\}$.

An edge directed to a variable vertex gives in what equation the variable will be computed. An edge directed from a variable vertex gives for which equations the variable is needed.

### 2.1.4 Canonical Decomposition

Any finite-dimensional SM can, according to classical results from the bipartite graph theory, be decomposed into three subsystems with specific properties, see figure 2.3. The subsystems can be associated with an over-, a just- and an under-determined subsystem and the decomposition can be done with e.g. a Dulmage-Mendelsohn decomposition. For diagnosis, only the overdetermined subsystem $M^+ = M_1^+ \cup \cdots \cup M_n^+ \cup R$ is used, where $R$ is the set of redundant equations. $M_i^+$ is a König-Hall component, here called strongly connected component or SCC, corresponding to a set of equations that have to be solved simultaneously.

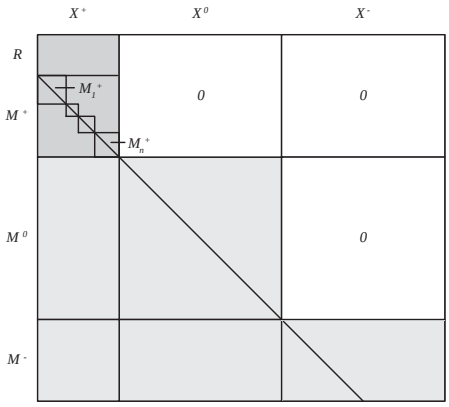



**Figure 2.3.** Decomposition of an SM into an over- , a just- and an under-determined subsystem ($M^+$, $M^0$ and $M^-$).

## 2.2 Stability of Linear and Non-Linear Systems

There are several kinds of stability, such as stability of equilibrium points, stability of periodic orbits and input-output stability. For more information see for example [10]. One common definition of stability, regarding stability of solutions to a differential equation, will here be presented. For a general autonomous non-linear system written in state-space form

$$\dot{x} = f(x) \tag{2.2}$$

with initial conditions $x(0)$, stability of the solutions is defined as follows.

**Definition 2.7 (Stability of solutions)** *Let $x^*(t)$ be a solution to the differential equation (2.2) corresponding to the initial condition $x^*(0)$. The solution is said to be*

- ***stable**, if there for every $\epsilon$ exists a $\delta$ so that*

$$|x^*(0) - x(0)| < \delta \Rightarrow |x^*(t) - x(t)| < \epsilon, \quad \forall t \geq 0$$

  *where $x(t)$ is the solution to (2.2) corresponding to the initial condition $x^(0)$.*

- ***unstable**, if not stable.*

- ***asymptotic stable**, if it is stable and there is a $\delta$ so that*

$$|x^*(0) - x(0)| < \delta \Rightarrow |x^*(t) - x(t)| \to 0, \quad t \to \infty.$$

Definition 2.7 is in general hard to use for stability investigation of linear or non-linear systems. Some more applicable results will now be presented.

### 2.2.1 Stability of Linear Systems

For a linear system written in state-space form

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{2.3}$$

following result and its proof can be found in for example [7], [9] or [16].

**Theorem 2.1 (Stability of Linear Systems)** *A linear system described by (2.3) is **asymptotically stable** if and only if*

$$Re\{\lambda_i(A)\} < 0$$

*where $\lambda_i(A)$ are the eigenvalues of $A$.*

Note that it is only the properties of the matrix $A$ that determines the stability of the linear system (2.3), Theorem 2.1 can therefore also be used for an autonomous system, i.e. when $B = D = 0$.

### 2.2.2 Stability of Non-Linear Systems

For non-linear systems, stability analysis is a bit more complicated than for linear systems. One way to investigate stability for non-linear systems is to analyze the stability of equilibrium points.

#### Equilibrium Points and Linearization

For a non-linear system

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x,u) \end{aligned} \tag{2.4}$$

an equilibrium point or singular point is defined as follows.

**Definition 2.8 (Equilibrium Point)** *An **equilibrium point** to* (2.4) *is a point* $(\bar{x},\bar{u})$ *so that*

$$f(\bar{x},\bar{u}) = 0.$$

If the functions $f$ and $h$ in (2.4) is continuously differentiable near the point $(\bar{x},\bar{u})$ the system can be linearized and written as

$$\begin{aligned} \dot{z} &= Az + Bv \\ w &= Cz + Dv \end{aligned}$$

where $z = x - \bar{x}$, $v = u - \bar{u}$, $w = y - h(\bar{x},\bar{u})$ and

$$A = \left.\frac{\partial f}{\partial x}\right|_{x=\bar{x}}, \quad B = \left.\frac{\partial f}{\partial u}\right|_{u=\bar{u}}, \quad C = \left.\frac{\partial h}{\partial x}\right|_{x=\bar{x}}, \quad D = \left.\frac{\partial h}{\partial u}\right|_{u=\bar{u}}.$$

It should be noted that only the matrix $A$ is needed for stability investigation. Linearization of the non-autonomous system (2.4) will be used later and is therefore presented here.

#### Stability of Equilibrium Points

Stability properties for an equilibrium point can be analyzed. Let the system

$$\dot{x} = f(x) \tag{2.5}$$

have an equilibrium point $\bar{x}$ so that $f(\bar{x}) = 0$. Stability of the equilibrium point can then be defined as follows.

**Definition 2.9 (Stability of equilibrium points)** *The equilibrium point* $\bar{x}$ *of* (2.5) *is said to be*

- ***stable**, if for each* $\epsilon > 0$, *there is a* $\delta = \delta(\epsilon) > 0$ *such that*

$$||x(0)|| < \delta \Rightarrow ||x(t)|| < \epsilon, \quad \forall t \geq 0$$

- ***unstable***, *if not stable*
- ***asymptotically stable***, *if it is stable and* $\delta$ *can be chosen such that*

$$||x(0)|| < \delta \Rightarrow \lim_{t\to\infty} x(t) = 0$$

If (2.5) is continuously differentiable near $\bar{x}$, the system can be linearized and described by

$$\dot{x} = Ax. \tag{2.6}$$

The following theorem spells out conditions under which we can draw conclusions about stability of the equilibrium point $\bar{x}$ to the non-linear system (2.5) by investigating stability for the linearized system (2.6). The theorem and its proof can be found in for example [10].

**Theorem 2.2 (Local Asymptotic Stability of an Equilibrium Point)** *The equilibrium point* $\bar{x}$ *to* (2.5) *is*

- ***locally asymptotic stable***, *if* $Re\{\lambda_i(A)\} < 0$ *for all eigenvalues* $\lambda_i(A)$ *of* $A$ *given by the linearization* (2.6) *of* (2.5) *near* $\bar{x}$.
- ***unstable***, *if* $Re\{\lambda_i(A)\} > 0$ *for one or more of the eigenvalues of* $A$.

It is important to note that this type of stability analysis is local and only says something about the system near an equilibrium point. Even if one can show that a non-linear system has numerous asymptotic stable equilibrium points theres is no guarantee that the non-linear system is globally stable.

## 2.3 Observer Theory

In control theory, state feedback is a well studied and commonly used strategy when designing a control system, see for example [6] and [7]. When sufficient measurements of the states are unavailable, a commonly used approach is to design an *observer* for state estimation. The basic idea is to use information about current and past values of the input and output signals of the system to generate an estimate of the (assumed unknown) current state. There are many approaches for designing an observer to a non-linear system. For example, an observer design for the system

$$\begin{aligned} \dot{x} &= f(x,u) \\ y &= h(x,u) \end{aligned} \tag{2.7}$$

is

$$\dot{\hat{x}} = f(\hat{x},u) + \ell(y - h(\hat{x},u))$$

where $\ell(\cdot)$ is some linear or non-linear function. One problem arising, is how to choose $\ell$ so that $\bar{e} = 0$ is an asymptotic stable equilibrium point to the estimation error $e = \hat{x} - x$ . This is a difficult problem to solve in general.

### 2.3.1 Linear Observers and Eigenvalue Assignment

Consider a linear system given by

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du. \end{aligned} \tag{2.8}$$

A commonly used linear observer design for (2.8), see for example [16], is

$$\dot{\hat{x}} = A\hat{x} + Bu + K(y - C\hat{x} - Du). \tag{2.9}$$

There are certain conditions under which the dynamic properties of the observer (2.9) can be chosen freely. Before presenting those, some needed properties of linear systems will be defined.

**Definition 2.10 (Observability)** *The linear system* (2.8) *with* $u(t) \equiv 0$ *is called* ***observable*** *on* $[t_0, t_f]$ *if any initial state* $x(t_0) = x_0$ *is uniquely determined by the corresponding response* $y(t)$ *for* $t \in [t_0, t_f]$.

**Theorem 2.3 (Observability Conditions)** *The linear system* (2.8) *is* ***observable*** *if and only if the observability matrix* $\mathcal{O}$ *satisfies*

$$rank\{\mathcal{O}(A, C)\} = n$$

*where*

$$\mathcal{O}(A, C) = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

*and* $n$ *is the number of states in* (2.8).

#### Eigenvalue Assignment

Following result regarding eigenvalue assignment or pole-placement can be found in for example [6], [7], [9] or [16].

**Theorem 2.4 (Eigenvalue Assignment)** *If the linear system* $(A, C)$ *described by* (2.8) *is observable, the state feedback gain* $K$ *in* (2.9) *can be chosen such that the matrix* $A - KC$ *gets arbitrary eigenvalues* $\lambda_i(A - KC)$.

Theorem 2.4 states that if the system (2.8) is observable, the dynamics of the observer

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

can be chosen arbitrary by modifying the eigenvalues, i.e. the poles, to the matrix $A - KC$.

### Stability of Linear Observers

The dynamic equation of the estimation error, $e = \hat{x} - x$, can be written as

$$\dot{e} = (A - KC)e. \tag{2.10}$$

According to Theorem 2.1 the system (2.10) is asymptotic stable if and only if

$$\mathrm{Re}\left\{\lambda_i(A - KC)\right\} < 0.$$

This result is also applicable to linearized non-linear systems. Assume that the system (2.7) can be linearized near the equilibrium point $(\bar{x}, \bar{u})$. The linear observer (2.9) can then be used and the stability of the equilibrium point $(\bar{x}, \bar{u})$ can be analyzed. Since the analysis is based on a linear approximation of the underlying non-linear system the analysis is only valid in a region near the equilibrium operating point. The conclusion that can be made is that the dynamics of the estimation error is *locally asymptotic stable* near $(\bar{x}, \bar{u})$.

Even if one have the possibility to choose the placement of the poles, there is no easy answer on where they should be placed. The placement of poles is a trade-off between sensitivity for disturbances and the rate of which the estimation-error decreases to zero. There are numerous concepts and theories available for computing the feedback-gain $K$. A few of them will be further described.

## 2.3.2 The Kalman Filter

When a stochastic description of system is considered, the Kalman Filter can be used as an observer. The Kalman Filter has been the subject of extensive research and application during the years and is widely used. Predictions from a stochastic model and measurements of the system are weighted together so that the variance of the estimation error, $e = \hat{x} - x$, is minimized. Consider a system described as

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

where $w$ and $v$ are white noises and

$$\begin{aligned} E\left[w\right] = E\left[v\right] &= 0 \\ E\left[ww^T\right] &= Q \\ E\left[vv^T\right] &= R \\ E\left[wv^T\right] &= N. \end{aligned}$$

The steady-state Kalman Filter, see for example [8], is given by

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

where

$$K = \left[PC^T + GN\right] R^{-1} \tag{2.11}$$

and $P$ is given by the matrix equation

$$AP + PA^T + GQG^T - \left[PC^T + GN\right] R^{-1} \left[PC^T + GN\right]^T = 0. \tag{2.12}$$

When $w$ and $v$ are white and gaussian noises, the Kalman Filter is the optimal linear filter. As can be seen in (2.11) and (2.12), the size of the feedback-gain $K$ depends on the covariance matrices $Q$ and $R$. It can be shown that it is only the relative difference between the matrices $Q$ and $R$ that is important. If the stochastic properties of the disturbances $w$ and $v$ can not be modeled, the matrices $Q$ and $R$ may be used as parameters in the filter design.

If $Q > R$ is used, one assume that there are more disturbances affecting the system than the measurements. This results in an observer where estimations of the states are more based on measurements than on the predictions from the model. The observer will respond quicker to changes than if $R > Q$ but instead be very sensitive to disturbances affecting the measurements. If instead $R > Q$, one say that there are more disturbances affecting the measurements and the estimations of the states will be more based on what the model predicts than the measurements. The resulting observer will be insensitive for measurement disturbances but not that fast.

### 2.3.3 Non-linear Kalman Filter

For a non-linear system on the form

$$\begin{aligned} \dot{x} &= f(x,u) + Gw \\ y &= h(x,u) + v \end{aligned} \tag{2.13}$$

there are non-linear versions of the Kalman Filter, see for example [8]. The basic idea is to linearize (2.13) along a trajectory and then apply linear Kalman Filter theory on the linearized system. If linearizing continuously, the matrices $A$, $C$, $Q$ and $R$ will change in time and the matrix $P$ may not converge to a constant matrix as $t \to \infty$. Hence, the formulas for the steady-state Kalman filter given earlier can not be used straight ahead. The trajectory, along which the linearizations are made, can either be a nominal trajectory given by the solution to the equation $\dot{x} = f(x,u)$ or a trajectory based on the state-estimation currently available. The second approach is called Extended Kalman Filter (EKF) or Schmidt EKF and the linearization of (2.13) must be done in real-time which may require lots of computations. The Extended Kalman Filter is not optimal in the same way as the linear Kalman Filter. According to [8] there are almost no useful analytical results on the performance of an EKF (at the time of publication). To get a reasonable filter, a considerable amount of testing and tuning is needed. More information regarding this can be found in for example [7] or [8].

#### Constant Gain Extended Kalman Filter

Another design approach, called Constant Gain Extended Kalman Filter (CGEKF) is to linearize the non-linear model (2.13) at operating points and compute a

Kalman feedback gain for every linearization, according to Section 2.3.2. The pre-computed Kalman gains are then integrated with the non-linear model instead of the linearized model. The resulting observer can be written as

$$\dot{\hat{x}} = f(\hat{x}, u) + K_i(y - h(\hat{x}, u)$$

where the calculation of $K_i$ is based on a linearized model of (2.13) valid near the operating point $(\bar{x}_i, \bar{u}_i)$. Present operating conditions, i.e. values of states and input signals, then determines which Kalman gain to use. By linearizing in a number of different points the observer can be used in a wide operating area. This design approach may not require the same amount of computations as the EKF and is therefore better suited for real-time applications. What concerns the performance of the CGEKF, the situation is the same as for the EKF described earlier. More information about the CGEKF can be found in [17].

# Chapter 3

# Model Based Diagnosis

Computers are used continuously to control and evaluate data in many technical systems today. Measurements from sensors and actuators can be used to diagnose if a system is working normally and if not, locate where a fault has occurred and take necessary actions. In many systems even a small fault may have consequences, the occurrence of a fault in a truck engine may for example lead to a higher level of emissions, higher fuel consumption or worse, a breakdown.

If, for example, two sensors are used to measure the same quantity, it is possible to test if one of the sensors is faulty. These types of diagnosis tests are based on hardware redundancy, but in many cases, satisfactory mathematical models of the physical processes to be diagnosed do exist. These mathematical models can then be used to find analytical redundancy, which is essential for the construction of a model based diagnostic test.

The basis of model based diagnosis is shortly described in the following sections and for a more detailed description see [14].

## 3.1 Diagnostic Tests

A diagnosis system often consists of a number of diagnostic tests based on observations of the system to be diagnosed. The purpose of each diagnostic test is to investigate if a specific behavioral mode is present or not. A diagnostic test can be viewed as a hypothesis test $\delta$, with the hypothesis

$$\begin{aligned} H^0: &\quad F_p \in M \\ H^1: &\quad F_p \in M^C \end{aligned}$$

where $F_p$ denotes the present behavioral mode in the system and $M$ the behavioral modes corresponding to the non-monitored faults. The common convention is that when $H^0$ is rejected, it is assumed that $H^1$ is true. The outcome of the hypothesis test $\delta$ is a decision

$$S = \begin{cases} S^1 = M^C & \text{if } H^0 \text{ is rejected} \\ S^0 \subseteq \Omega & \text{if } H^0 \text{ is not rejected} \end{cases}$$

where $\Omega$ denotes all behavioral modes. For each hypothesis test $\delta$, a rejection region is defined, i.e. a subset of observations where the hypothesis $H^0$ is rejected. This is often done via a test quantity, which is a function $T(z)$ from observations to a scalar value that can be compared to a threshold $J$. The hypothesis test $\delta$ is then defined as

$$S = \delta(z) = \begin{cases} S^1 & \text{if } T(z) \geq J \\ S^0 & \text{if } T(z) < J \end{cases}$$

This means that the test quantity $T(z)$ must be designed such that it is low if the observations $z$ match the hypothesis $H^0$, i.e. a behavioral mode in $M$ can explain the observations.

### 3.1.1 Analytical Redundancy

A sufficient and necessary condition to find test quantities is that the system contains analytical redundancy, which can be formally defined as follows.

**Definition 3.1 (Analytical Redundancy)** *There exists **analytical redundancy** if there are two or more different ways to determine a variable $x$ by only using the observations $z$, i.e. $x = f_1(z)$ and $x = f_2(z)$ where $f_1(z) \not\equiv f_2(z)$.*

For analytical redundancy to exist, a system must be overdetermined or consist of at least one overdetermined subsystem.

## 3.2 Residual Generators

In a system where analytical redundancy exists, a common way to construct a test quantity is to base it upon a residual. Two important properties of a residual, is that it should be zero in the fault-free case and non-zero in the case of a fault. A residual generator can be formally defined as

**Definition 3.2 (Residual Generator)** *A system in state-space form with inputs $z$ and output $r$ is a **residual generator** and $r$ is a **residual** if*

$$z \in \mathcal{O} \Rightarrow \lim_{t \to \infty} r = 0.$$

The set $\mathcal{O}$ is the set of all known signals $z$ that is consistent with the fault-free model describing the system that should be diagnosed with the residual. If the model is valid under the hypothesis $H^0$, the set is called $\mathcal{O}_{H^0}$.

### 3.2.1 Consistency Relations

A consistency relation[1] is any relation between known signals that, in the fault-free case, always holds. Because of this property, consistency relations are often the basis for residuals. A consistency relation can formally be defined as

[1] Also referred to as parity relation, parity equation, parity function or analytical redundancy relation.

**Definition 3.3 (Consistency Relation)** *An analytical relation $C$ of known signals $z$ is a* ***consistency relation*** *if*

$$z \in \mathcal{O} \Rightarrow C(z) = 0.$$

Note that a consistency relation can directly be used as a residual generator if all signals in $z$ are non-differentiated, i.e. $C(z)$ is a static relation.

**Example 3.1**

Consider the system

$$\begin{aligned} \dot{x} &= -x + u \\ y_1 &= x \\ y_2 &= x. \end{aligned} \tag{3.1}$$

Obviously the system contains analytical redundancy and a consistency relation derived from (3.1) is

$$0 = y_1 - y_2$$

which can be used to form the residual

$$r_1 = y_1 - y_2. \tag{3.2}$$

The static system (3.2) is then a residual generator. Another consistency relation can be derived from (3.1) as

$$0 = \dot{y}_1 + y_1 - u$$

which requires some modification to form a residual generator. The same equations can be used to construct a residual generator as

$$\begin{aligned} \dot{x} &= -x + u \\ r_2 &= y_1 - x. \end{aligned} \tag{3.3}$$

## 3.3 Faults

As shown in Figure 3.1, a technical system can often be separated into three subsystems: actuators, the process and sensors. Depending on in what subsystem a fault occurs, a fault is classified to be an *actuator fault*, *process fault* or *sensor fault*. Typical actuator and sensor faults can be changes in gain and bias.


**Figure 3.1.** General structure of a technical system.

### 3.3.1 Fault Detectability and Sensitivity

Definition 3.2 does not state anything on what happens with the residual in case of a fault. A specific fault $f_i$ is *detectable* in a system if there exists a $f_i \neq 0$, where the observations are distinguishable from observations when the system is fault-free. Hence, fault detectability is a system property.

A specific linear residual generator is *sensitive* to a fault $f_i$ if $G_{rf_i}(s) \neq 0$, where $G_{rf_i}$ is the transfer function from fault to residual. A fault $f_i$ is strongly detectable in the residual $r$ if $G_{rf_i}(0) \neq 0$.

**Example 3.2**

Consider again the model in Example 3.1. If the sensor faults affecting $y_1$ and $y_2$ are called $f_{y_1}$ and $f_{y_2}$ respectively, the model (3.1) can be written as

$$\begin{aligned} \dot{x} &= -x + u \\ y_1 &= x + f_{y_1} \\ y_2 &= x + f_{y_2}. \end{aligned}$$

It is obvious that both $f_{y_1}$ and $f_{y_2}$ is detectable. By studying (3.2) and (3.3) one can conclude that $r_1$ is sensitive to $f_{y_1}$ and $f_{y_2}$ and that $r_2$ is sensitive to $f_{y_1}$.

## 3.4 The Procedure of Residual Generation

The procedure of designing residual generators based on a model is shown in Figure 3.2 and can briefly be outlined by the following steps.

**Step 1:** In most applications, the system to be diagnosed is modeled with some tool, e.g. SIMULINK or MODELICA. This step transforms the model to an analytical equation system, more suitable for further analysis and manipulation.

**Step 2:** Redundancy is a requirement for model based diagnosis. This step aims to find subsystems in an analytical equation system where redundancy exists.

**Step 3:** This step creates residual generators based on the subsystems containing redundancy.

Two methods for residual generation will be described in Chapter 4 and 5. In both methods Step 2 above is performed using structural analysis, see Section 2.1.

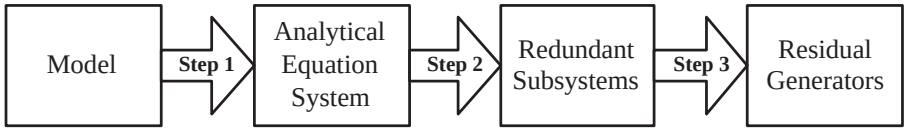



**Figure 3.2.** The procedure of residual generation.

# Part II

# Methods for Residual Generation

# Chapter 4

# Residual Generation with The Scania Method

This chapter describes the method used at Scania CV AB for computing all possible residual generators from a given Simulink-model. The method is outlined by the following steps.

1. Extract analytical model equations from a Simulink-model.
2. Transform an analytical model to an SM.
3. Extract all MSO sets in the SM.
4. Find residual generators in all MSO sets.

The Scania method is based on work carried out in [3], [4] and [11]. In this method, step 2 and 3 are contained in the second step in the more general design procedure shown in Figure 3.2. The method is summarized by Algorithm 4.1 and the different steps will be further described in the following sections.

Algorithm 4.1 outputs all residual generators as an unmatched residual equation, $e$, associated with a computation sequence, $C_e$, for the unknown variables contained in $e$.

## 4.1 Extract Analytical Model Equations from the Simulink Model

As mentioned above, all models used at Scania are implemented in Simulink. This step in the design algorithm retrieves the analytical model equations from the Simulink-file. The procedure is divided into three sub-steps. The first step is to simplify the Simulink-model by arranging the information in a less complicated structure. The analytical equations from the Simulink structure are then extracted and stored together with information regarding the Simulink-blocks.

**Algorithm 4.1** The Scania Method

**Input:** A SIMULINK-model file (mdl), $Sim$

```
1:  procedure FindResidualGenerators(Sim)
2:      R := ∅
3:      E := Sim2ME(Sim)
4:      S := ME2SM(E)
5:      S_M := FindAllMSO(S)
6:      for all MSO sets M ∈ S_M do
7:          M' := RemoveNonInvertibleEdges(M)
8:          for all equations e ∈ M' do
9:              M'^- := M' \ {e}
10:             SCC := FindStronglyConnectedComponents(M'^-)
11:             Γ := FindPerfectMatching(M'^-)
12:             if |SCC| ≤ 1 ∧ |Γ| ≠ 0 then
13:                 if BlockTypesOK(e) then
14:                     C_e := CalcCompSeq(equ(Γ))
15:                     R := R ∪ {C_e, e}
16:                 end if
17:             end if
18:         end for
19:     end for
20: end procedure
```

**Output:** A set of residual generators $R$

Finally the analytical equations are simplified to decrease the complexity of the system. For a more detailed description of this step see [4]. The step is performed by SIM2ME in Algorithm 4.1.

## 4.2 Transform the Analytical Model to an SM

The analytical equations derived from the SIMULINK model are in this step transformed to an SM. This is a straight forward process since information about the SIMULINK blocks are included in the analytical model. The SM also contains information about invertibility i.e. which variables an equation will solve for. For a more detailed description of this step see [4]. The step is performed by ME2SM in Algorithm 4.1.

## 4.3 Extract all MSO sets in the SM

The procedure of finding all MSO sets in the SM is divided into three sub-steps. First the overdetermined part of the model is extracted since just- or underdetermined parts can not contain any MSO sets. The model is then simplified by

combining equations that have to be used together in an MSO set, resulting in a less complex model. The last step is to find all MSO sets. This is important but complex and several algorithms for doing this has been developed, see [12]. The algorithm used in this method is based on [4] and performed by FindAllMSO in Algorithm 4.1. Briefly the algorithm performs a systematical reduction of the set of equations in the overdetermined SM until all MSO sets are extracted. For a detailed description see [4].

## 4.4 Find Residual Generators in the MSO Sets

Since the MSO sets have been extracted with structural analysis there is no guarantee that the equations in the MSO set can be ordered and executed. Since an MSO sets always has one more equation than unknown variables a residual generator can be constructed by computing all unknown variables and then use the redundant equation as the residual equation. To guarantee that a unknown variable is assigned by only one equation a computation sequence for the variables in the MSO set has to be found. The main objective in this step is to find a computation sequence in every MSO set.

### 4.4.1 Invertibility Properties

Many Simulink blocks used in the engine models at Scania are not invertible e.g. maps, saturations, min and max functions. The Scania method uses integral causality which means that a differential equation $x = \int x_d \, dt$ can not be inverted as $x_d = \frac{d}{dt}x$ since calculating derivatives are considered impossible. As mentioned in Section 4.2, information about invertibility is included in the structural model and this information is used to create a directed bipartite graph representation of the MSO set. The invertibility properties are included in the bipartite graph by removing edges that represent a non-invertible relation between a variable and a equation. This is done by RemoveNonInvertibleEdges in Algorithm 4.1.

### 4.4.2 Bipartite Matchings and Computation Sequence

The implemented algorithm for finding all possible residual generators in the MSO set (with non-invertible edges removed) removes one equation at a time and searches the reduced equation set for a perfect matching, free of algebraic loops. For doing this, the algorithm uses some special properties of bipartite graphs. If the bipartite graph does not contain any SCC (Strongly Connected Components) of size $> 1$, the found perfect matching is free of algebraic loops and this implies that there only exists one matching, see [11].

Finally the equations in the cycle-free perfect matching are reordered so that unknown variables used as inputs to an equation are calculated first, i.e. a computation sequence is created from the matching. The removed, unmatched equation, only contains unknown variables computed by the computation sequence, and can therefore be used as a residual equation. If the residual equation contains the

SIMULINK-block `Integrator`, `Mux` or `From Workspace` the equation is not used due to implementation difficulties and the residual generator is discarded.

The step is performed by the functions FINDSCC, CALCCOMPSEQ and BLOCKTYPESOK in Algorithm 4.1. For a more detailed description of the whole step, see [11].

**Example 4.1: The Scania Method**

Consider the model

$$\begin{aligned} e_1: &\quad \dot{x} = \alpha x + u \\ e_2: &\quad y = x. \end{aligned} \tag{4.1}$$

This example will use SIMULINK/MATLAB to calculate all executable residual generators from the model (4.1). An implementation of the model in SIMULINK is shown in Figure 4.1.

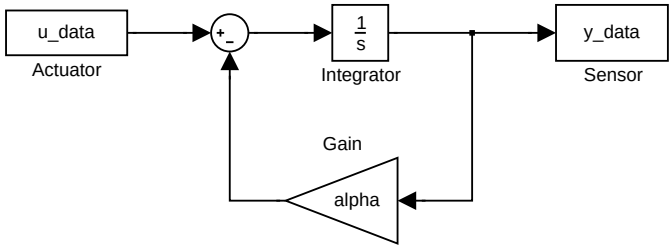



**Figure 4.1.** SIMULINK implementation of model (4.1).

First MATLAB was used to extract model equations from the SIMULINK-model file. The result is shown below.

```
1 Actuator   FromWorkspace 0 : a4 = u_data
2 Gain       Gain          0 : a3 = alpha * a2
3 Integrator Integrator    1 : a2 = Int(a1)
4 Sensor     ToWorkspace   0 : y_data = a2
5 Sum        Sum           0 : a1 = a4 - a3
```

The first column contains the number of the extracted analytical equation, the second column the physical interpretation of the equation, the third column the SIMULINK blocktype and the fourth column a 1 if the equation can not be inverted and 0 else. Note that $a1, \ldots, a4$ are variables used by SIMULINK representing the signals connecting the blocks in the model. The extracted analytical model

equations can be written in a more convenient way as

$$e_1: \quad a_4 = u \tag{4.2a}$$
$$e_2: \quad a_3 = \alpha a_2 \tag{4.2b}$$
$$e_3: \quad a_2 = \int a_1\, dt \tag{4.2c}$$
$$e_4: \quad y = a_2 \tag{4.2d}$$
$$e_5: \quad a_1 = a_4 - a_3. \tag{4.2e}$$

Note that the integration in (4.2c) is performed numerically and requires knowledge about the initial value of variable $a_2$. The model is then transformed to a biadjacency matrix shown below

| Equation | Unknown | | | | Known | |
|---|---|---|---|---|---|---|
| | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $u$ | $y$ |
| $e_1$ | | | | X | X | |
| $e_2$ | | X | X | | | |
| $e_3$ | $\Delta$ | X | | | | |
| $e_4$ | | X | | | | X |
| $e_5$ | X | | X | X | | |

and the corresponding bipartite graph is shown in Figure 4.2. Note that the edge $(e_3, a_1)$ marked with a dashed line represents the non-invertible relation corresponding to the $\Delta$ in the biadjacency matrix due to integral causality.

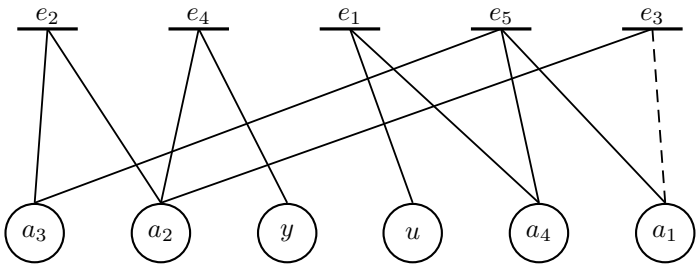



**Figure 4.2.** The bipartite graph corresponding to model (4.2).

The model (4.2) consists of four unknown variables, $\{a_1, a_2, a_3, a_4\}$, and five equations, $\{e_1, e_2, e_3, e_4, e_5\}$, and the MSO found contains all equations. By looking at the biadjacency matrix one can see that there are no SCC present and therefore no unsolvable algebraic cycles. Next, one equation at a time is removed and the bipartite graph is searched for complete matchings. Only two perfect matchings can be found. When equation $e_3$ is removed the matching found is

$$\Gamma_3 = \{(e_1, a_4), (e_2, a_3), (e_4, a_2), (e_5, a_1)\}.$$

Since $e_3$ contains an integrator it is not used as a residual equation and the corresponding residual generator is discarded. The perfect matching found when

equation $e_4$ is removed is

$$\Gamma_4 = \{(e_1, a_4), (e_2, a_3), (e_3, a_2), (e_5, a_1)\}.$$

Equation $e_4$ can be used as a residual equation and the computation sequence given from the matching $\Gamma_4$ is shown in Figure 4.3.

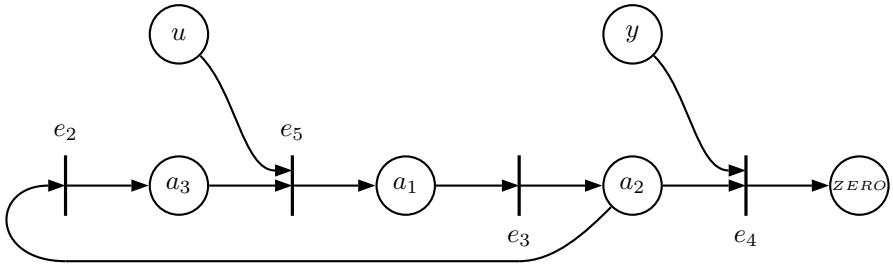



**Figure 4.3.** The directed graph for residual generator $r$.

The found residual generator $r$ can be written as

$$\begin{aligned} a_3(t) &:= \alpha a_2(t) \\ a_1(t) &:= u(t) - a_3(t) \\ a_2(t) &:= \int a_1 \, dt \\ r(t) &:= y(t) - a_2(t). \end{aligned}$$

This residual generator is considered executable since integral causality is assumed in the Scania Method i.e. the initial condition for the integrator is known and therefore the initial value of variable $a_2$.

## 4.5 Structure of the Residual Generators

All models considered in this thesis are in state-space form, and can be described with

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

Searching the model for MSO sets, see Section 4.3, may result in sets where variables $x_i$ is included, but the differentiated variables $\dot{x}_i$ is not. Hence, we separate the variables in $x$ in two disjoint sets, $x_1$ and $x_2$. For a variable $x_i \in x_1$, the differentiated variable $\dot{x}_i$ is included in the MSO set. The set $x_2$ includes all the variables in the MSO set where the differentiated variable is not included. With the separation of the variables, the MSO set can be written as

$$\dot{x}_1 = f_1(x_1, x_2, u) \tag{4.3a}$$
$$0 = g_1(x_1, x_2, u, y). \tag{4.3b}$$

Let $n_w$ denote the number of elements in the vector $w$. Due to (4.3) being an MSO set, it holds that $n_{g_1} = n_{x_2} + 1$. To obtain a residual generator in state-space form $x_2$ has to be solved for in (4.3b) and $x_2$ can then be substituted into (4.3a). If $x_2$ can not be solved for, the Scania method will discard the MSO set for not being possible as a residual generator on state-space form. With the substitution, the MSO set (4.3) can be written as

$$\dot{x}_1 = \tilde{f}(x_1, z) \tag{4.4a}$$
$$0 = \tilde{g}(x_1, z), \tag{4.4b}$$

where $z = \{u, y\}$ is a vector of known variables and $\tilde{f}$ and $\tilde{g}$ are the function vectors $f_1$ and $g_1$ where $x_2$ is substituted. Hence, the equation (4.4b) is the part of (4.3b) where $x_2$ was not solved for and $n_{\tilde{g}} = 1$. This shows that only DAE systems with index 0 and 1 are found with the Scania Method and that all found DAE systems can be written in state-space form. Note that there might exist another solution for $x_2$ in (4.3b). This may result in that another equation instead of equation (4.4b) will be used as the residual equation.

# Chapter 5

# Residual Generation with The Lille Method

This section describes a method for finding all computable residual generators for a given set of analytical model equations. The method is based on work mostly done at the Université Lille and the Technical University of Denmark, described in [2]. In this thesis the method will be referred to as the Lille method and it can be outlined by the following steps.

1. Extend a model with equations describing differentiation relations between variables.
2. Transform an analytical model to an SM.
3. Find residual generators in the SM.

The analytical model equations are here considered as given, i.e. step 1 in Figure 3.2, an extraction from a Simulink model as in Section 4.1, is not performed. In this method, step 1 and 2 are contained in the second step in Figure 3.2. The method is summarized by Algorithm 5.1 and the different steps are further described in the following sections.

## 5.1 Extending the Model

The dynamic aspect of the model variables can be considered as an extra set of equations. In this method, the unknown variables $x$ and $\dot{x}$ are considered to be structurally different. A new set of variables is introduced $x_i^d = \dot{x}_i$ and extended differential equations are added for every $x_i^d$ in the analytical model. The extended differential equations are

$$x_i^d = \dot{x}_i.$$

This step is performed by AddExtendedDifferentialEquations in Algorithm 5.1 and gives an extended analytical model.

**Algorithm 5.1** The Lille Method

**Input:** A Set of Model Equations, $E$

1: **procedure** FINDRESIDUALGENERATORS($E$)
2: $RG := \varnothing$
3: $E' :=$ ADDEXTENDEDDIFFERENTIALEQUATIONS($E$)
4: $S :=$ ME2SM($E'$)
5: $SO :=$ IDENTIFYSOSUBSYSTEM($S$)
6: $C :=$ ALLCOMPLETECAUSALMATCHINGS($SO$)
7: **for all** complete causal matchings $\Gamma \in C$ **do**
8: $CS :=$ RANK($equ(\Gamma)$)
9: $R := SO \setminus equ(\Gamma)$
10: $CR := R \cup \{CS, R\}$
11: $RG :=$ CONSTRUCTRG($CR$)
12: **end for**
13: **end procedure**

**Output:** A Set of Residual Generators, $RG$

## 5.2 Transform the Analytical Model to an SM

The extended analytical model equations are in this step transformed to an SM, which is further described in Section 2.1.1. In Algorithm 5.1, this is done by ME2SM.

Due to stability problems with integration, the Lille method uses derivative causality which means that a differential equation $x^d = \frac{d}{dt}x$ is used to compute $x^d$, but it is not inverted as $x(t) = x(0) + \int_0^t x^d(\tau)\, d\tau$. The problem when calculating derivatives if noise is present is not considered here. If the initial condition $x(0)$ is known, $x(t)$ can be computed uniquely with integration and the variable $x$ can be matched in the differential equation. This is called integral causality, but it is not further considered in the method. The variable which can not be matched is denoted with a $\Delta$ in the biadjacency matrix. A more detailed description is given in [2].

## 5.3 Find Residual Generators in the SM

If present, the SO part of the structural model is extracted by canonical decomposition, see Section 2.1.4. This is done by IDENTIFYSOSUBSYSTEM in Algorithm 5.1. In the SO part, consistency relations have to be found which are constructed from complete causal matchings of the unknown variables, associated with unmatched equations. When searching for the matchings the assumption of derivative causality is considered. In Algorithm 5.1, this is done in steps 6 - 12. $equ(\Gamma)$ represents the equations in the matching $\Gamma$.

A causal matching is a matching that does not contain any differential loops,

see Section 5.3.1. A non-causal matching can not be unambiguously determined, if the initial condition is not known, and hence can not be used to construct a consistency relation. If the matching contains algebraic loops, the loops have to be condensed into an equation system where all the unknown variables can be computed from known variables, this is further described in Section 7.1.

The matchings found can be represented with a directed graph and an algorithm is used on the biadjacency matrix to find a possible computation sequence, called ranking algorithm. The ranking algorithm 5.2 is similar to the one presented in [2], but some adjustments are made, due to the algorithm not working satisfactorily. The ranking algorithm does only find computation sequences without loops, so it may not find any complete matching even though it exists. In more complex situations complete matchings can be found by selecting an initial matching and trying to increase the alternated chain by changing the matched and unmatched variables.

**Algorithm 5.2** Ranking Algorithm

**Input:** Biadjacency matrix or bipartite graph

1: Mark all known variables with 0.
2: $i := 1$.
3: Find an equation with exactly one unmarked variable. Associate rank $i$ with this equation and mark it as well as the corresponding variable.
4: If there are unmarked equations whose variables are all marked, associate them with rank $i$, mark them and connect them with the pseudo-variable ZERO.
5: $i := i + 1$.
6: If there are unmarked variables or equations, continue with step 3.

**Output:** Ranking of the equations

After finding a computation sequence for a complete matching, the unmatched equations are denoted $ZERO$. These are the equations that are used as residual equations in the overdetermined subgraph. With the computation sequence for the unknown variables and the residual equations, consistency relations are obtained.

Assuming that all derivatives of known variables can be approximated, the consistency relations obtained can be written in state-space form. The frequencies of the known signals must be considered to have an upper limit for the assumption to hold. As the consistency relation can be modified to a system in state-space form, the consistency relation can be used to construct a residual generator. Approaches to this are proposed in [18] and [5], but it is considered out of scope of this thesis. Since faults affecting a consistency relation is the same as in the corresponding residual generator, it is still interesting to compare the two. Hence, all examples of methods assuming derivative causality will in this thesis end up in a consistency relation. Modifying a consistency relation to a residual generator is performed by ConstructRG in Algorithm 5.1.

### 5.3.1 Differential Loop

A differential loop is an algebraic loop, see Section 7.1, containing one or more differential equations. Due to the differential equations, the loop can not be unambiguously determined if the initial values of the unknown variables in the loop are not known.

**Example 5.1: A Differential Loop**

Consider the model

$$\begin{aligned} e_1: \quad x^d &= x \\ e_2: \quad x^d &= \dot{x} \end{aligned}$$

that leads to the directed graph in Figure 5.1. The obtained equation to be solved is $x = \dot{x}$. The solution $x = x_0 e^t$ can not be unambiguously determined, if $x_0$ is not known.

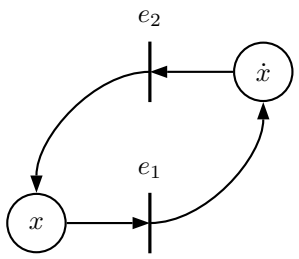



**Figure 5.1.** Directed graph showing a differential loop, giving a non-causal matching.

**Example 5.2: The Lille Method**

Consider the model

$$\begin{aligned} e_1: \quad \dot{x} &= \alpha x + u \\ e_2: \quad y &= x \end{aligned}$$

Adding extended differential equations to the model, see Section 5.1, gives

$$\begin{aligned} e_1: \quad x^d &= \alpha x + u \\ e_2: \quad y &= x \\ e_3: \quad x^d &= \dot{x}. \end{aligned} \tag{5.1}$$

The transformation to a SM gives the biadjacency matrix shown below and the bipartite graph shown in Figure 5.2.

| Equation | Unknown | | Known | |
|---|---|---|---|---|
| | $x^d$ | $x$ | $u$ | $y$ |
| $e_1$ | X | X | X | |
| $e_2$ | | X | | X |
| $e_3$ | X | $\Delta$ | | |

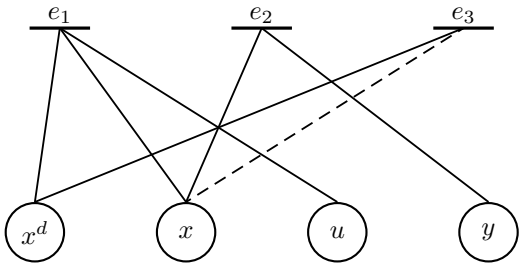



**Figure 5.2.** The bipartite graph corresponding to model (5.1).

This structural model is a SO system and hence does not need to be decomposed. If the ranking algorithm is used, Algorithm 5.2, it is only possible to find two different matchings. By changing edges in the matching, three complete causal matchings with respect to the unknown variables are found

$$\begin{aligned}\Gamma_1 &= \left\{\left(e_1, x^d\right), (e_2, x)\right\} \\ \Gamma_2 &= \left\{(e_1, x), \left(e_3, x^d\right)\right\} \\ \Gamma_3 &= \left\{(e_2, x), \left(e_3, x^d\right)\right\}.\end{aligned}$$

For $\Gamma_1$ with the unmatched equation $e_3$ the directed graph is shown in Figure 5.3 and the corresponding consistency relation is

$$0 = \alpha y + u - \dot{y}. \tag{5.2}$$

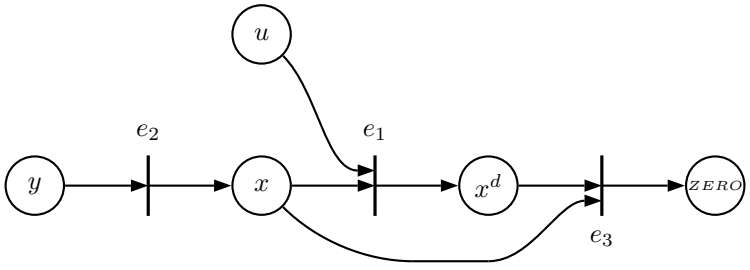



**Figure 5.3.** The directed graph of $\Gamma_1$ with $e_3$ as residual equation giving the consistency relation (5.2).

For $\Gamma_2$ with the unmatched equation $e_2$, the directed graph is shown in Figure 5.4.

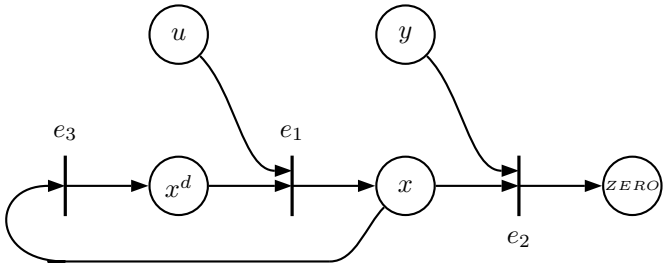



**Figure 5.4.** The directed graph of $\Gamma_2$ with $e_2$ as residual equation.

Due to the differential loop in $\Gamma_2$, the consistency relation is discarded. For $\Gamma_3$ with the unmatched equation $e_1$, the directed graph is shown in Figure 5.5 and the corresponding consistency relation is

$$0 = \dot{y} - \alpha y - u. \tag{5.3}$$

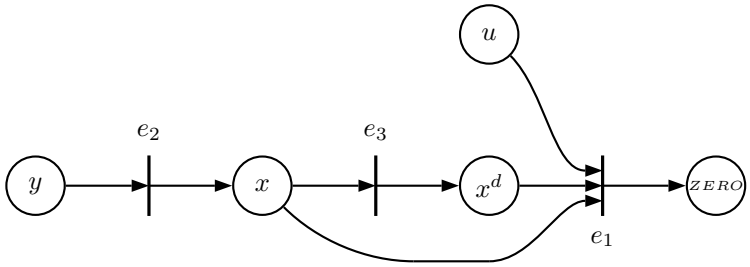



**Figure 5.5.** The directed graph of $\Gamma_3$ with $e_1$ as residual equation giving the consistency relation (5.3).

# Chapter 6

# Analysis of Methods for Residual Generation

In this chapter an analysis of methods for residual generation will be made. The most obvious difference in the methods described in Chapters 4 and 5 is the causality assumption, hence the methods compared here are both methods with integral and derivative causality. As methods using derivative causality, the Lille method described in Chapter 5 are used with one modification. The SO systems, where residual generators are searched for, are the MSO sets of the system. For methods with integral causality a slightly different approach, compared to the Scania method, is used with extended equation systems. In the rest of this thesis, methods assuming integral causality will use the approach described below.

## 6.1 Integral Causality

When considering integral causality a new set of variables is introduced, to make the comparison easier the same notion as with derivative causality in the Lille method is used, $\dot{x}_i = x_i^d$. For every $x_i^d$ the analytical model is extended with the differential equation

$$x_i^d = \dot{x}_i.$$

Integral causality implies that a matching in the equation can be made with $x_i$, but not with $x_i^d$. This is the opposite to derivative causality, and the Lille Method, and $x_i^d$ will, instead of $x$, be denoted with $\Delta$ in the biadjacency matrix. The computation of $x_i$ is

$$x_i(t) = x_i(0) + \int_0^t x_i^d(\tau)\, d\tau$$

where $x_i(0)$ is assumed to be known. This is actually the same as what is done in the Example 4.1, though the differential equation is denoted with $x = \int x^d$. The

equation $e_3$ in (4.2) can be seen as the differential equation of the DAE system where $a_1 = \dot{a}_2$

$$\begin{aligned} e_1: \quad a_4 &= u \\ e_2: \quad a_3 &= \alpha a_2 \\ e_4: \quad y &= a_2 \\ e_5: \quad \dot{a}_2 &= a_4 - a_3. \end{aligned}$$

In this section, integral causality has been discussed and a comparison with the Scania method is done. This is also shortly described in [2] and, for the AI approach to diagnosis, in for example [15].

**Example 6.1: Integral Causality**

Consider the example model

$$\begin{aligned} e_1: \quad \dot{x} &= \alpha x + u \\ e_2: \quad y &= x. \end{aligned}$$

The extended model is

$$\begin{aligned} e_1: \quad x^d &= \alpha x + u \\ e_2: \quad y &= x \\ e_3: \quad x^d &= \dot{x}. \end{aligned} \tag{6.1}$$

Transformation of the extended analytical model to a structural model gives the following biadjacency matrix and the bipartite graph in Figure 6.1.

| Equation | Unknown | | Known | | |
|---|---|---|---|---|---|
| | $x^d$ | $x$ | $u$ | $y$ | $x(0)$ |
| $e_1$ | X | X | X | | |
| $e_2$ | | X | | X | |
| $e_3$ | $\Delta$ | X | | | X |

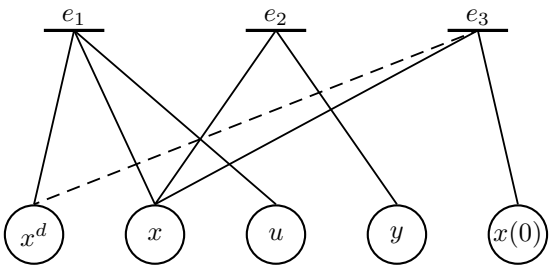



**Figure 6.1.** The bipartite graph corresponding to model (6.1).

The model (6.1) consists of two unknown variables, $x$ and $x^d$, and three equations, $e_1$, $e_2$ and $e_3$. It is trivial to extract all MSO sets from the structural model

since the only MSO set is $\{e_1, e_2, e_3\}$. By looking at the biadjacency matrix one can see that there are an SCC present, if equations $e_1$ and $e_3$ are used in a matching. Though this is no problem when integral causality is used, since the residual generators obtained always must be systems in state-space form.

In the MSO set there are two complete matchings found, $\Gamma_1 = \left\{\left(e_1, x^d\right), (e_2, x)\right\}$ and $\Gamma_2 = \left\{\left(e_1, x^d\right), (e_3, x)\right\}$. Now, a residual generator is obtained from the matching $\Gamma_2$ with the residual equation $e_2$, the directed graph is shown in Figure 6.2 and the system is

$$\begin{array}{lrcl} e_1: & x^d & = & \alpha x + u \\ e_3: & \dot{x} & = & x^d \\ e_2: & r_1 & = & y - x. \end{array}$$

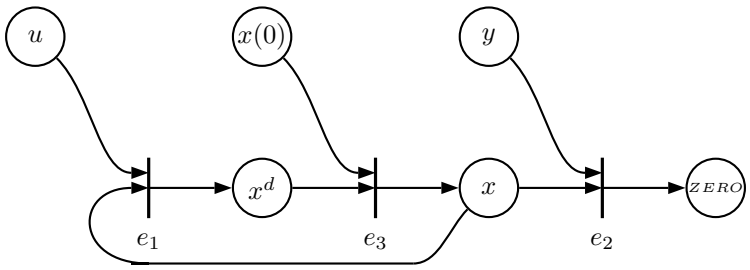



**Figure 6.2.** The directed graph for residual generator $r_1$.

To get a residual generator from the matching $\Gamma_1$ with the residual equation $e_3$ some modifications of the system are required, this since it is not in state-space form. The directed graph is shown in Figure 6.3, and a corresponding equation system is

$$\begin{array}{lrcl} e_2: & x & = & y \\ e_1: & x^d & = & \alpha x + u \\ e_3: & \dot{r}_2 & = & \dot{x} - x^d. \end{array}$$

where dynamics are introduced in the residual $r_2$ due to a differentiated variable in the residual equation. Depending on how the residual generator is modified, different dynamics can be introduced. Modification approaches are presented in e.g. [14].

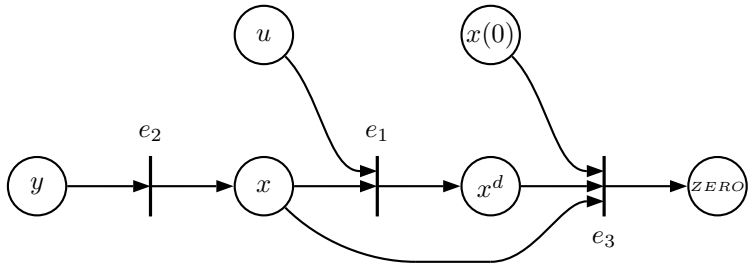



**Figure 6.3.** The directed graph for residual generator $r_2$.

Since $x(0)$ is considered given when integral causality is used and obviously no fault can occur in the signal, $x(0)$ will not be represented in SM in following sections.

## 6.2 Equivalence of Methods

To be able to show if two different methods are equivalent, it is first required that a definition is made of what is meant. When designing a diagnosis system the important results are what fault detectability that can be obtained. Hence it is important which system equations the residual generators are constructed from.

**Definition 6.1 (Input-Output Equivalence for Model $M$)** *Two methods for designing residual generators are* ***input-output equivalent for the given analytical model*** *$M$, if all sets of system equations used to construct residual generators are the same in both methods.*

For a fault to be detectable in a diagnosis system there has to be a residual generator sensitive to that specific fault, see [14]. For a residual generator to be sensitive to a fault, that fault has to be present in the system of equations from which the residual generator is constructed. Hence, two residual generators constructed from different system of equations can be sensitive to different faults.

## 6.3 Comparison of Methods

In this section, methods with integral and derivative causality are defined and compared. In the end, some examples with similarities and differences due to causality assumptions are given. If a method uses the extended differential equations as residual equations it is said to use differential residual equations or shorter, DRE. All methods will use MSO sets to find residual generators and hence all methods can be outlined by the following steps.

1. Add the extended differential equations to the analytical model.

2. Transform the extended analytical model to an SM.
3. Find all MSO sets in the SM.
4. Find all complete matchings of unknown variables in the MSO sets where all edges in the matchings represents invertible relations.
5. Construct a residual generator from every matching together with the remaining residual equation in the MSO set where the matching was found.

Now, four methods using this four steps for finding residual generators are proposed and will be used in different comparisons. The methods are shown in Table 6.3.

| Method | Causality | DRE |
|---|---|---|
| $\mathcal{A}$ | Derivative | No |
| $\mathcal{B}$ | Derivative | Yes |
| $\mathcal{C}$ | Integral | No |
| $\mathcal{D}$ | Integral | Yes |

**Table 6.1.** Different methods for comparison.

The meaning of the table is e.g. that method $\mathcal{A}$ uses derivative causality and that all residuals with a DRE will be discarded in the method.

### 6.3.1 Equivalence of Methods with same Causality

In this section a comparison of methods with the same causality assumption will be done, but where one is using DRE and the other is not. This means method $\mathcal{A}$ will be compared to method $\mathcal{B}$ and method $\mathcal{C}$ to method $\mathcal{D}$.

To simplify the understanding of the proofs, it is here given a general semi-explicit equation system with extended differential equations

$$x_1^d = f(x_1, x_2, z) \tag{6.2a}$$
$$0 = g(x_1, x_2, z) \tag{6.2b}$$
$$\dot{x}_1 = x_1^d \tag{6.2c}$$

where $x_1$,$x_1^d$ and $x_2$ are vectors of unknown variables and $z$ are a vector of known variables. A DRE is then an residual equation in (6.2c), here denoted with $d_i$. A specific equation in (6.2a) or (6.2b) is here denoted with $e_i$. The following theorems shows the relation between some of the methods compared.

**Theorem 6.1** *For a given equation system, $M$, method $\mathcal{A}$ and $\mathcal{B}$ are input-output equivalent.*

**Theorem 6.2** *For a given equation system, $M$, method $\mathcal{C}$ and $\mathcal{D}$ are input-output equivalent.*

**Proof (Theorem 6.1)** First, from the definitions of the methods it holds that every residual generator obtained with method $\mathcal{A}$ will also be obtained with method $\mathcal{B}$. Assume a residual generator is obtained with method $\mathcal{B}$ from a perfect matching, $\Gamma$, in the equation set $M$ together with the DRE, $d_i$. Then, the equation set leading to the residual generator is $M_d = M \cup d_i$. Derivative causality gives that $x_i^d \in x_1^d$ can always be matched in $d_i$. Switch the edge $(e_i, x_i^d)$, where $x_i^d$ is matched with equation $e_i \in M$, with the edge $(d_i, x_i^d)$. This gives the equation set $M_2 = (M \setminus e_i) \cup d_i$ of the matching, $\Gamma_2$. The residual generator obtained from the perfect matching $\Gamma_2$, with equation set $M_2$, and the residual equation $e_i$, has the equation set $M_e = M_2 \cup e_i$. It holds that

$$M_e = M_2 \cup e_i = (M \setminus e_i) \cup d_i \cup e_i = M \cup d_i = M_d$$

which shows that both residual generators obtained use the same equation set. The conclusion is that, for every residual generator obtained with a DRE, there exists a residual generator using the same equation set but not using a DRE. Hence for all equation sets leading to a residual generator in method $\mathcal{B}$ there will exist at least one residual generator with the same equation set obtained in method $\mathcal{A}$. □

**Proof (Theorem 6.2)** The proof is the same as above, with the difference that the edges switched are $(e_i, x_i) \in \Gamma$ and $(d_i, x_i)$, where $x_i \in x_1$. □

Theorem 6.1 and 6.2 show that some of the methods are equivalent, hence one do not have to use all of the methods when constructing residual generators. For dynamic systems we will, from now on, only use method $\mathcal{A}$ and $\mathcal{C}$.

With a similar discussion as above it is also possible to compare the residuals that are obtained from the two residual generators that switched edges in the proof. Theoretically, and if the same computations of integrals and derivatives are used, one can show that these residuals will at every time have the same absolute value. This is another reason to only compare method $\mathcal{A}$ and $\mathcal{C}$ for dynamic systems.

### 6.3.2 Methods with different Causality

As said earlier in this chapter, the most obvious difference between the Scania method and the Lille method is the causality assumption. What differences the causality assumptions can have to the obtained diagnosis systems is here discussed by equivalence of methods for different types of systems. The two types of systems that are investigated are static systems and dynamic systems of first order derivatives.

#### Static Systems

In this section a comparison is made using static systems. For a static system it holds that $x^d = \emptyset$ and hence no differential equations will be added to the model. Consider a static equation system

$$0 \quad = \quad g(x, z)$$

where $x$ and $z$ are vectors of respectively known and unknown variables. All methods will get the same SM and also the same MSO sets. Since no differential

equations are added to the model, obviously method $\mathcal{A}$ and $\mathcal{B}$ are input-output equivalent as well as method $\mathcal{C}$ and $\mathcal{D}$. There are no extended differential equations added for a static model in any of the methods, which gives that the SM will not contain any information about causality. Since none of the differences for the methods is considered in static systems, conclusions can be drawn that, for all static systems, the methods ($\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$) are pairwise input-output equivalent.

### Dynamic Systems

A discussion regarding equivalence of methods with different causality is found in this section. Some examples of dynamic systems and the finding of residual generators, which might simplify the understanding of this discussion, is made in the next section. Due to Theorems 6.1 and 6.2 only methods $\mathcal{A}$ and $\mathcal{C}$ will be considered from now on. First, consider a general dynamic system in semi-explicit form

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

where $x_1$, $x_2$ and $z$ are vectors and $x_2$ consists of the unknown variables where $\dot{x}_2$ are not in the system. An MSO set found in 6.3, with extended differential equations, can be written as

$$x_1^d = f(x_1, x_2, z) \tag{6.3a}$$
$$0 = g(x_1, x_2, z) \tag{6.3b}$$
$$\dot{x}_1 = x_1^d. \tag{6.3c}$$

Note that the causality marking in the SM will only be made for one variable of the equations in (6.3c). Also note that what is actually done when adding the extended differential equations (6.3c) is to transform a general equation system to semi-explicit form.

If a residual generator is found with method $\mathcal{A}$, all elements in $x_1^d$ have to be matched in (6.3c) and $x_1$ and $x_2$ have to be solved for in the equations (6.3a) and (6.3b). Similarly, if a residual generator is to be found with method $\mathcal{C}$, all elements in $x_1$ have to be matched in (6.3c) and $x_1^d$ and $x_2$ have to be solved for in the equations (6.3a) and (6.3b).

Now assume that $x_1$ and $x_2$ but not $x_1^d$ can be solved for in (6.3a) and (6.3b), this results in that a residual generator can be found with method $\mathcal{A}$ but not with method $\mathcal{C}$. If instead it is assumed that (6.3a) and (6.3b) can be solved for $x_1^d$ and $x_2$ but not $x_1$, a residual generator can be found with method $\mathcal{C}$ but not with method $\mathcal{A}$. Hence, one can conclude that for dynamic systems in general, method $\mathcal{A}$ and $\mathcal{C}$ are not input-output equivalent. Though there are dynamic systems where the methods are. Consider for example an equation system with only one MSO set, if it is possible to solve (6.3a) and (6.3b) for $x_1^d$ and $x_2$ and also for $x_1$ and $x_2$ both method $\mathcal{A}$ and $\mathcal{C}$ finds residual generators in the only equation set of the system.

Since for all dynamic systems method $\mathcal{A}$ and $\mathcal{C}$ are not input-output equivalent, there could be differences in detectability for diagnosis systems constructed with the different methods. The most obvious example is a system where one of the methods finds a residual generator but the other does not, see for example 6.3.3. Hence using both methods would for detectability be preferable when designing diagnosis systems.

### 6.3.3 Some Examples

This section will give some examples of dynamic systems and how residual generators are found using method $\mathcal{A}$ and $\mathcal{C}$. They will show both similarities and differences.

#### An Equivalent Dynamic System

This example shows an equation system where method $\mathcal{A}$ and $\mathcal{C}$ are input-output equivalent. Consider the extended equation system

$$\begin{array}{lrcl} e_1: & x^d & = & -x+u \\ e_2: & y & = & x \\ e_3: & x^d & = & \dot{x}. \end{array} \tag{6.4}$$

The system has one MSO set and the SM considered with integral causality are represented with the following biadjacency matrix.

| Equation | Unknown | | Known | |
|---|---|---|---|---|
| | $x^d$ | $x$ | $u$ | $y$ |
| $e_1$ | (X) | X | X | |
| $e_2$ | | X | | X |
| $e_3$ | $\Delta$ | (X) | | |

One complete matching is denoted with circles and the residual generator obtained from this matching is

$$\begin{array}{lrcl} e_1: & x^d & = & -x+u \\ e_3: & \dot{x} & = & x^d \\ e_2: & r & = & y-x. \end{array}$$

Now, look at the SM with derivative causality represented with the following biadjacency matrix.

| Equation | Unknown | | Known | |
|---|---|---|---|---|
| | $x^d$ | $x$ | $u$ | $y$ |
| $e_1$ | X | X | X | |
| $e_2$ | | (X) | | X |
| $e_3$ | (X) | $\Delta$ | | |

One complete matching is denoted with circles and the consistency relation obtained from this matching is

$$0 = \dot{y} + y - u.$$

As there is only one MSO set found in the dynamic system and both method $\mathcal{C}$ and $\mathcal{A}$ finds a residual generator, the methods are input-output equivalent for the equation system (6.4).

### A Non-Equivalent System 1

This example shows an equation system with only one MSO set where method $\mathcal{C}$ finds a residual generator, but where method $\mathcal{A}$ does not. The problem arising when using derivative causality is due to non-invertible functions. Method $\mathcal{A}$ would, if the functions were invertible, find a complete matching. Consider the extended equation system

$$\begin{array}{lrcl} e_1: & x_1^d & = & f_1(x_1, x_2) \\ e_2: & x_2^d & = & f_2(x_2, u) \\ e_3: & y & = & x_1 \\ e_4: & x_1^d & = & \dot{x}_1 \\ e_5: & x_2^d & = & \dot{x}_2. \end{array} \tag{6.5}$$

where $f_1(x_1, x_2)$ is non-invertible for both variables and $f_2(x_2, u)$ for $x_2$. The SM considered with integral causality are represented with the following biadjacency matrix.

| Equation | Unknown | | | | Known | |
|---|---|---|---|---|---|---|
| | $x_1^d$ | $x_2^d$ | $x_1$ | $x_2$ | $u$ | $y$ |
| $e_1$ | (X) | | $\Delta$ | $\Delta$ | | |
| $e_2$ | | (X) | | $\Delta$ | X | |
| $e_3$ | | | X | | | X |
| $e_4$ | $\Delta$ | | (X) | | | |
| $e_5$ | | $\Delta$ | | (X) | | |

The model is an MSO set and one complete matching is denoted with circles in the biadjacency matrix. The residual generator found from this matching is

$$\begin{array}{lrcl} e_1: & x_1^d & = & f_1(x_1, x_2) \\ e_2: & x_2^d & = & f_2(x_2, u) \\ e_4: & \dot{x}_1 & = & x_1^d \\ e_5: & \dot{x}_2 & = & x_2^d \\ e_3: & r & = & y - x_1. \end{array}$$

Now, look at the SM with derivative causality represented with the following biadjacency matrix.

| Equation | Unknown | | | | Known | |
|---|---|---|---|---|---|---|
| | $x_1^d$ | $x_2^d$ | $x_1$ | $x_2$ | $u$ | $y$ |
| $e_1$ | X | | $\Delta$ | $\Delta$ | | |
| $e_2$ | | X | | $\Delta$ | X | |
| $e_3$ | | | X | | | X |
| $e_4$ | X | | $\Delta$ | | | |
| $e_5$ | | X | | $\Delta$ | | |

Since the column of $x_2$ only includes $\Delta$, there are no complete matchings found in the system and obviously no consistency relations. Since only method $\mathcal{C}$ finds a residual generator in the MSO set the methods are not input-output equivalent for the equation system (6.5).

### A Non-Equivalent System 2

This example shows an MSO set where method $\mathcal{A}$ finds a residual generator, but where method $\mathcal{C}$ does not. Consider the equation system with extended differential equations

$$\begin{array}{lrcl} e_1: & x_1^d & = & -x_1 + x_3 \\ e_2: & x_2^d & = & -x_2 + x_3 \\ e_3: & y_1 & = & x_1 \\ e_4: & y_2 & = & x_2 \\ e_5: & x_1^d & = & \dot{x}_1 \\ e_6: & x_2^d & = & \dot{x}_2. \end{array} \tag{6.6}$$

The SM of (6.6) with integral causality is shown in the following biadjacency matrix.

| Equation | Unknown | | | | | Known | |
|---|---|---|---|---|---|---|---|
| | $x_1^d$ | $x_2^d$ | $x_1$ | $x_2$ | $x_3$ | $y_1$ | $y_2$ |
| $e_1$ | X | | X | | X | | |
| $e_2$ | | X | | X | X | | |
| $e_3$ | | | X | | | X | |
| $e_4$ | | | | X | | | X |
| $e_5$ | $\Delta$ | | X | | | | |
| $e_6$ | | $\Delta$ | | X | | | |

No complete matchings can be found in the MSO set, due to the fact that e.g. $x_1^d$ and $x_2^d$ must be matched in $e_1$ and $e_2$, which are the only equations including $x_3$, giving that $x_3$ can not be matched. Now, look at the following biadjacency matrix of the SM for (6.6) with derivative causality.

| Equation | Unknown | | | | | Known | |
|---|---|---|---|---|---|---|---|
| | $x_1^d$ | $x_2^d$ | $x_1$ | $x_2$ | $x_3$ | $y_1$ | $y_2$ |
| $e_1$ | X | | X | | (X) | | |
| $e_2$ | | X | | X | X | | |
| $e_3$ | | | (X) | | | X | |
| $e_4$ | | | | (X) | | | X |
| $e_5$ | (X) | | $\Delta$ | | | | |
| $e_6$ | | (X) | | $\Delta$ | | | |

A complete matching is shown with circles in the biadjacency matrix. The equation $e_2$ is used as the residual equation and the obtained consistency relation is

$$0 = \dot{y}_1 + y_1 - \dot{y}_2 - y_2.$$

This example shows that the system (6.6) can be used to construct a residual generator with derivative causality but not with integral causality. Hence for the system (6.6), method $\mathcal{A}$ and $\mathcal{C}$ will not be input-output equivalent.

### An Unstable System

Consider the unstable equation system with an added differential equation

$$\begin{array}{lrcl} e_1: & x^d & = & x+u \\ e_2: & y & = & x \\ e_3: & x^d & = & \dot{x}. \end{array}$$

The corresponding SM with integral causality is shown in the following biadjacency matrix.

| Equation | Unknown | | Known | |
|---|---|---|---|---|
| | $x^d$ | $x$ | $u$ | $y$ |
| $e_1$ | Ⓧ | X | X | |
| $e_2$ | | X | | X |
| $e_3$ | $\Delta$ | Ⓧ | | |

The SM is an MSO set and the matching $\Gamma = \{(e_1, x^d), (e_3, x)\}$ with $e_2$ as residual equation will be the only found residual generator[1]. The equation system is

$$\begin{array}{lrcl} e_1: & x^d & = & x+u \\ e_3: & \dot{x} & = & x^d \\ e_2: & r & = & y-x. \end{array} \tag{6.7}$$

This residual generator is unstable, hence can not be used in a diagnosis system as it is. Though, by utilizing the method described in Chapter 8, a stable residual generator based on (6.7) can be constructed. Now, consider the SM with derivative causality shown in the following biadjacency matrix.

| Equation | Unknown | | Known | |
|---|---|---|---|---|
| | $x^d$ | $x$ | $u$ | $y$ |
| $e_1$ | X | X | X | |
| $e_2$ | | Ⓧ | | X |
| $e_3$ | Ⓧ | $\Delta$ | | |

There are two matchings found, but both matchings results in the same consistency relation

$$0 = \dot{y} - y - u. \tag{6.8}$$

Since (6.8) only consists of signals considered as known, it can always be used in a diagnosis system. This is an example showing the problem of instability with integral causality, a problem that is not present with derivative causality.

[1] According to the Scania method this is a residual generator. Actually, due to violation of Definition 3.2, it is not. In Section 8.1, this is defined as an unstable residual generator.

## 6.4 Improvements of the Scania Method

As said in Section 3.4, the procedure of designing residual generators can generally be divided into three steps.

1. Transform the model to an analytical equation system.
2. Find redundant subsystems in the analytical equation system.
3. Create residual generators from the redundant subsystems.

There are several improvements of the Scania Method to be further investigated. The first and second step in the design procedure, described in Sections 4.1 - 4.3, are considered to be satisfactory and will not be further investigated in this thesis. The following improvements all belong to the third step and would most likely increase the number of residual generators found.

- Handling algebraic loops, i.e. SCC of size larger than 1.
- Solve the problem of stabilizing unstable residual generators.
- Use derivative causality, i.e. make `Integrator` blocks invertible.
- Handle more than one solution to the unknown variables in the model.

Algebraic loops were addressed in Section 4.4, some methods for solving them are presented in Chapter 7. Unstable residual generators were briefly discussed in Section 6.3.3, and is further investigated in Chapters 8 - 10. A method with derivative causality were analyzed in Chapter 5. The main problem with this approach arises when trying to construct residual generators in state-space form from the obtained consistency relations. Since derivatives of measurement signals need to be estimated or the order of the consistency relation lowered, this is considered to be a hard problem to solve and out of scope of this thesis. Inverting functions may result in variables having more than one solution. This leads to computation sequences where variables and the corresponding residual generator can not be unambiguously calculated. In the Scania method, residual generators with this property are discarded and solving this problem would definitely be of great interest. It would probably increase the number of found residual generators, nevertheless the problem is considered out of scope of this thesis.

# Part III

# Improvements and Evaluation of the Scania Method

# Chapter 7

# Solving Algebraic Loops

Algebraic loops were first addressed in Section 4.4 giving that all residual generators containing algebraic loops are discarded in the Scania method. This section gives an explanation to algebraic loops and how they are found in structural analysis. Some solutions to the problem of solving algebraic loops will also be discussed.

## 7.1 Algebraic Loops

If an algebraic loop exists, two or more variables in a model have to be solved simultaneously. An algebraic loop can be found either by looking at SCCs in the canonical decomposition, see Section 2.1.4, or loops[1] in the matching of a bipartite graph.

If the equations can be solved for the unknown variables in the loop, it can be condensed to a system of equations where the computations of the unknown variables only consist of known variables. The condensed equation system can then be used to compute the residual generator.

**Example 7.1: An Algebraic Loop**

Consider the model

$$\begin{aligned} e_1: \quad y_1 &= x_1 + x_2 \\ e_2: \quad y_2 &= x_1 - x_2 \end{aligned}$$

that gives the following biadjacency matrix.

| Equation | Unknown | | Known | |
|---|---|---|---|---|
| | $x_1$ | $x_2$ | $y_1$ | $y_2$ |
| $e_1$ | X | (X) | X | |
| $e_2$ | (X) | X | | X |

[1] Often referred to as *cycles* in bipartite graph theory.

Two complete matchings with respect to unknown variables can be found. A directed graph from the one that is denoted with circles is shown in Figure 7.1. Analytically, the algebraic loop can easily be solved which gives

$$\begin{array}{lrcl} e_1: & x_1 & = & \frac{y_1+y_2}{2} \\ e_2: & x_2 & = & \frac{y_1-y_2}{2}. \end{array} \tag{7.1}$$

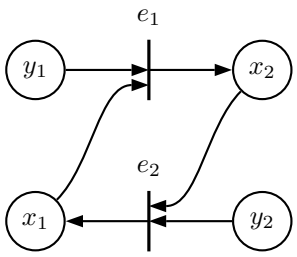



**Figure 7.1.** The directed graph from the matching in 7.1 showing an algebraic loop.

The condensed equation system (7.1) can be shown as the directed graph in Figure 7.2. Another way of solving the algebraic loop is to first solve it for one of the unknown variables and use the result to solve the other. One possibility is shown here with an equation system and a directed graph.

$$\begin{array}{lrcl} e_1: & x_1 & = & \frac{y_1+y_2}{2} \\ e_2: & x_2 & = & x_1 - y_2 \end{array}$$

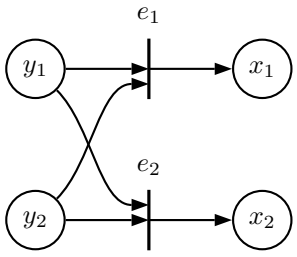



**Figure 7.2.** The condensed directed graph from the equation system 7.1.

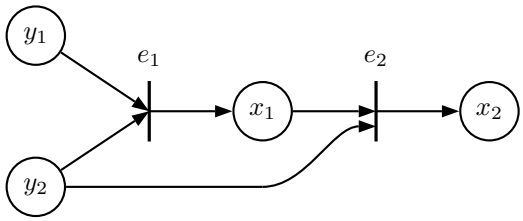



**Figure 7.3.** The condensed directed graph from the equation system 7.1.

A residual generator constructed with the Scania method can not handle more than one unique solution of the states in a system. This gives that even if a solution of an algebraic loop does exist, the solution can not be used if the unknown variables can not be uniquely defined.

## 7.2 Algebraic Loops in the Engine Model

If using the Scania method on the engine model, there are 31 possible residual generators discarded due to algebraic loops. Hence, if the algebraic loops are solved, it might be possible to save all these residual generators.

The Scania method was used with an engine model developed at Scania CV AB. In all discarded possible residual generators, only one algebraic loop was found, which is shown in (7.2).

$$\begin{aligned} x_1 &= \tfrac{x_5-1}{z_1-1} \\ x_2 &= kx_4 \\ x_3 &= \tfrac{z_2 z_3}{x_2} \\ x_4 &= m(x_1, z_4) \\ x_5 &= \tfrac{z_5}{x_3} \end{aligned} \tag{7.2}$$

The algebraic loop consists of five unknown, $x$, and five known variables, $z$, called known here since they are the variables that need to be computed before the loop can be solved. The function $m$ is a two-dimensional map of size $60 \times 60$, shown in Figure 7.4. The operating range of the map is concentrated to the center of the $z_4$-axis. The large values of $m$, for values of $x_1$ close to 1, originates from the parameter settings of the engine model and will not be used in a fault-free engine cycle.

If reducing the algebraic loop (7.2) to one equation with only one unknown variable $x_1$, it ends up with

$$0 = g(x_1, z_1, z_2, z_3, z_5) - m(x_1, z_4) \tag{7.3}$$

where

$$g(x_1, z_1, z_2, z_3, z_5) = \frac{z_2 z_3 (z_1 - 1)}{k z_5} x_1 + \frac{z_2 z_3}{k z_5}.$$

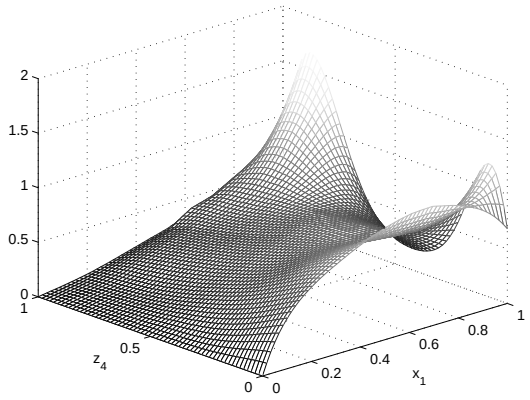



**Figure 7.4.** The map represented by $m(x_1, z_4)$ in equation system (7.2).

There are connections between known variables and constants which can be merged to make the map smaller. If $z_6 = \frac{z_2 z_3}{k z_5}$ and $z_7 = (z_1 - 1)z_6$, the algebraic loop can be written as

$$0 \quad = \quad g(x_1, z_6, z_7) - m(x_1, z_4) \tag{7.4}$$

where

$$g(x_1, z_6, z_7) \quad = \quad z_6 x_1 + z_7$$

If the algebraic loop (7.2) could be solved for one of the unknown variables it would in this case be easy to use these equations to solve the remaining. This gives that, if e.g. the algebraic loop could be solved for $x_1$ in (7.3) or (7.4) the remaining unknown variables can be solved for using $x_1$ as known variable in (7.2). This is not always the case for a general algebraic loop and if not, all included algebraic loops also need to be solved.

## 7.3 Methods for Solving Algebraic Loops

There are different ways for solving algebraic loops, *analytical solution*, *iterative solution* and *mapping* are commonly used methods. These methods together with a solution based on the specific case, here called *step-through solution*, will be further described.

### 7.3.1 Analytical Solution

An algebraic loop can be written as

$$0 = f(x, z)$$

where $x$ is a vector of unknown variables and $z$ is a vector of known variables in the algebraic loop. Analytically, by hand or with an analytical solver (e.g. Maple), the equation might be solved, which would give

$$x = g(z).$$

If this holds, the unknown variables $x$ can be computed from only known variables. Since the map $m$ is included in the algebraic loop (7.3) there exist no algebraic solution to this specific example.

### 7.3.2 Iterative Solution

An algebraic loop can be solved iteratively by e.g. Newton's method or fix-point iterations. The basic idea is to make iterations converging to the wanted value, terminating the iterations when a specified accuracy is obtained. These methods are non-deterministic in time and hence not a preferred option when computing in real-time. To limit the maximum time used, it is possible to terminate the computation after a certain number of iterations. Though, if terminating a solution due to this condition it will not get the specified accuracy. To be able to find all possible solutions of the algebraic loop, first it is needed to find all intervals, of the variable, leading to a unique solution and that is not an easy task.

It is possible to solve the algebraic loop (7.2) using this method, but since it is not deterministic in time with a specified accuracy it could take too much CPU time and hence the method will not be used.

### 7.3.3 Mapping

It is possible to map the inverse function to some degree of accuracy. The larger the map, the better accuracy, but also more memory is demanded. In the equation (7.3.1), consider $z$ as the inputs and $x$ as the outputs. Every combination of inputs $z$ can be mapped to a unique value in $x$. A problem arises if $x$ has more than one unique solution, then all the values have to be taken in account when computing the residual generator. This will not be considered in this thesis. The amount of memory required for the map grows with the number of elements in $z$ and the desired accuracy. Mapping for more values of an element in $z$ results in better accuracy. If $n_{z_i}$ is the number of values that $z_i \in z$ are mapped for, then the amount of memory required for the map depends on the product of all $n_{z_i}$. If mapping (7.2), $z$ has five elements and the amount of memory required for the map is too much and hence will not be used.

The reduced algebraic loop (7.4) has a $z$ with only three three elements and less memory will be required for the map. This map can be used, though this has not been done due to time shortage.

### 7.3.4 Step-Through Solution

Consider again the merged algebraic loop (7.4). Solving this equation only considers the dimension of $x_1$ in the map $m$, due to the consideration that $z_4$ in $m$ is

a known variable in the loop. If linear interpolation is used in intervals between points in the map, a solution at an interval is found by solving a linear equation, since $g(x_1, z_6, z_7)$ is also linear. Hence, solving (7.4) for $x_1$ can be done by stepping through all these intervals in the map, computing $x_1$ at all intervals where solutions do exist. Due to the linear equations, solutions in intervals are either unique or indefinite. If only one unique solution exists for an algebraic loop, this way of solving it will have a constant evaluation time depending on the number of intervals, in the map, that have to be checked. The solving of an algebraic loop with linear intervals is summarized in Algorithm 7.1.

**Algorithm 7.1** Step-Through Solution

**Input:** A function $f(x, z)$, where $x$ is the variable to be solved for, $f$ is linear at intervals of $x$ and $z$ is a vector of known signals

```
1:  procedure StepThroughSolution(f(x, z))
2:      x := ∅
3:      for all linear intervals, I = (x_i, x_{i+1}), of f do
4:          f_i = f(x_i, z)
5:          f_{i+1} = f(x_{i+1}, z)
6:          if f_i = 0 and f_{i+1} = 0 then
7:              x := IndefiniteSolution(f_i, f_{i+1})
8:              return
9:          else if f_i = 0 then
10:             x := x ∪ x_i
11:         else if f_i f_{i+1} < 0 then
12:             sol := FindSolution(x_i, x_{i+1}, f_i, f_{i+1})
13:             x := x ∪ sol
14:         end if
15:     end for
16: end procedure
```

**Output:** A solution for the variable $x$

In the specific case of the algebraic loop in the engine model, (7.4) is used as $f(x, z)$ in Algorithm 7.1. The intervals $I = (x_i, x_{i+1})$ are the intervals between breaking points of the x-axis where interpolation of the map $m$ is made. The goal is to find all values of $x$ where the function $f(x, z) = 0$, to do this all linear intervals of $f(x, z)$ have to be checked. If $f = 0$ at a whole interval, there are indefinite number of solutions and IndefiniteSolution is called. If one solution exists at an interval, FindSolution computes the solution using the fact that two uniform triangles can be formed in the interval, shown in Figure 7.5. The solution is

$$x = x_i - \frac{f_i(x_{i+1} - x_i)}{f_{i+1} - f_i}.$$

This method could be used to solve the algebraic loop (7.2). For example, solve (7.4) for $x_1$ using 7.1 and use the result to solve the remaining unknown variables.

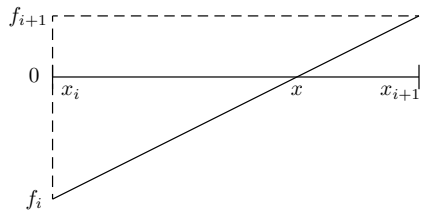



**Figure 7.5.** How to compute the solution for $x$ at the interval $I = (x_i, x_{i+1})$, where the function $f$ is linear.

It would be deterministic in time and Algorithm 7.1 would require some memory. Though compared to mapping, if more algebraic loops with linear intervals are found in a model, the same algorithm can be used and no additional memory is required.

# Chapter 8

# Constructing Stable Residual Generators

The Scania Method does not consider stability of residual generators at all during the design process. Some of the residual generators found are unstable and the corresponding residuals are not suitable for diagnosis. When using the Scania Method on a specific engine model, 70 residual generators were found in total. After running the stability investigation algorithm described in Section 8.1.2, 18 residual generators were removed due to instability, which is almost 26 %. For detectability and isolability matters, a method for constructing stable residual generators would be of great importance.

## 8.1 Stability of Residual Generators

With definitions 3.2 and 2.7 in mind, stability of residual generators in state-space form will now be defined. As described in Section 4.5, a general form for a residual generator found with the Scania Method is

$$\dot{x} = f(x, z) \tag{8.1a}$$
$$r = g(x, z) \tag{8.1b}$$

Assume that the solution $x(t)$ to (8.1a) is uniquely determined by the initial conditions $x_0 = x(0)$.

**Definition 8.1 (Stability of a Residual Generator for a given $z$)** *A residual generator* (8.1) *is* ***stable for a given*** $z$ *if:*

1. *The solution* $x(t)$ *to* (8.1a) *is stable, according to Definition 2.7, for all* $x_0$
2. $\lim_{t\to\infty} r = 0$

*A residual generator is* ***unstable for a given*** $z$ *if not stable.*

**Definition 8.2 (Stability of a Residual Generator for $\Omega$)** *A residual generator* (8.1) *is* ***stable for a set*** $\Omega$ ***of*** $z$ *if it is stable for all* $z \in \Omega$*. A residual generator is* ***unstable for*** $\Omega$ *if not stable.*

**Definition 8.3 (Stability of a Residual Generator)** *A residual generator* (8.1) *is* ***stable*** *if it is stable for all* $z \in \mathcal{O}_{H_0}$*. A residual generator is* ***unstable*** *if not stable.*

### 8.1.1 Causes of Instability

In order to evaluate a residual generator, the non-linear system (8.1a) must be simulated. Since this is done in a computer, numerical integration methods are used which can result in numerical instability. An investigation concerning the accuracy and stability of different integration methods is carried out in [3], where fixed step simulation is compared with variable step simulation. The conclusions is that forward Euler with stepsize 2.5 ms is well enough and does not cause instability problems when simulating residual generators. Numerical stability is considered out of the scope of this thesis and will not be further investigated.

With the assumption that numerical problems are not the cause of instability, stability properties of the residual generator must be investigated. Non-ideal circumstances may cause instability during execution of a residual generator. The non-ideal circumstances can for example be certain approximations and assumptions made for simplicity during the modeling work or measurement noise and errors in actuators and sensors. They both lead to inconsistency between the measured data and what the model predicts which may cause a stable residual generator to behave like an unstable residual generator. The instability may be related to the *robustness* of the residual generator. Robustness is a measure on how the difference between the model and the real system affects the stability properties of the system, see [7] or [10].

Another cause of instability is bad initial conditions of states. A residual generator may have both stable and unstable operating points. An initial condition near an unstable operating point may lead to instability even if the data, with which the residual generator is run, originates from a stable operating point. This is illustrated in example 8.1.

**Example 8.1**

Consider the autonomous non-linear system described by

$$\dot{x} = x^2 - 1 \tag{8.2a}$$
$$y = x. \tag{8.2b}$$

A residual generator based on (8.2) is

$$\dot{x} = x^2 - 1 \tag{8.3a}$$
$$r = y - x. \tag{8.3b}$$

The system (8.3a) has the equilibrium points $\bar{x}_1 = 1$ and $\bar{x}_2 = -1$. The linearization in $\bar{x}_1$ is

$$A_1 = 2, \quad B_1 = 0, \quad C_1 = 1, \quad D_1 = 0$$

and

$$A_2 = -2, \quad B_2 = 0, \quad C_2 = 1, \quad D_2 = 0$$

in $\bar{x}_2$. The eigenvalues of $A_1$ and $A_2$ are 2 and -2 respectively, and the equilibrium point $\bar{x}_1$ is unstable and $\bar{x}_2$ is stable. To illustrate that initial conditions can affect the stability of a residual generator, (8.3) was implemented and simulated in Simulink. The implementation is shown in Figure 8.1.

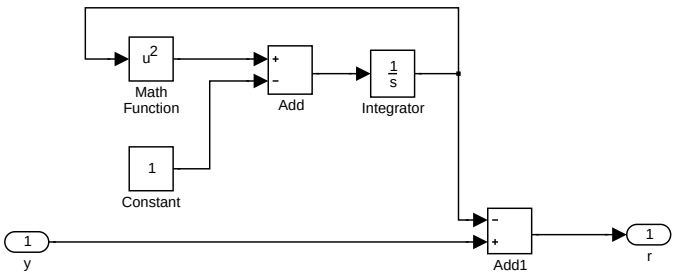



**Figure 8.1.** Simulink implementation of residual generator (8.3).

The residual generator was simulated with initial conditions $x_0 = -1$, $x_0 = -3.5$ and $x_0 = 1.5$. A constant measurement sequence where $y = -1$, corresponding to the stable equilibrium point $\bar{x}_2$, with added gaussian noise was used. The result is shown in Figure 8.2. It is obvious that $x_0 = 1.5$ causes instability and that $x_0 = -3.5$ and $x_0 = -1$ does not. The conclusion is that bad initial conditions may cause instability even if the residual generator is run in a stable operating point. Initial conditions must therefore be carefully chosen.

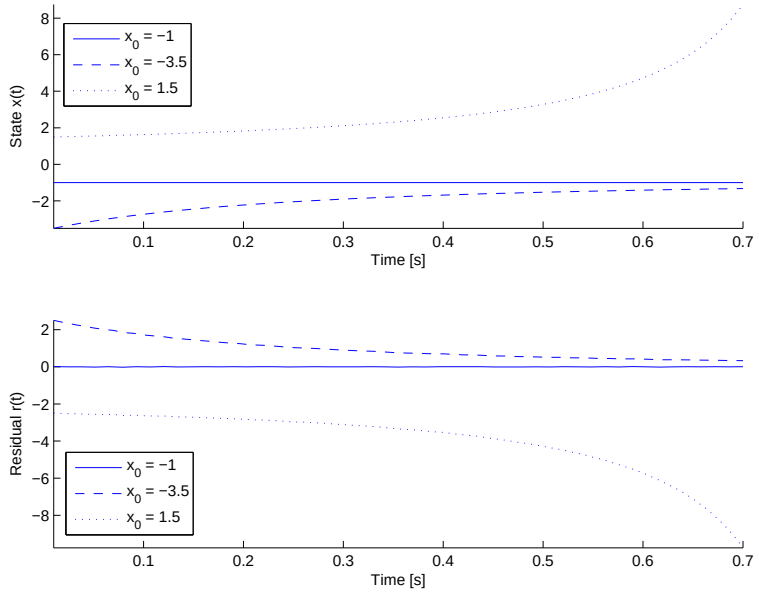



**Figure 8.2.** Simulation of residual generator (8.3) for initial conditions $x_0 = -1$, $x_0 = -3.5$ and $x_0 = 1.5$ and constant $y = -1$. It is obvious that $x_0 = 1.5$ causes instability of the residual generator and that $x_0 = -3.5$ and $x_0 = -1$ does not.

### 8.1.2 Residual Stability Investigation

In Section 2.2, results regarding stability of linear and non-linear systems were presented. The methods for investigating stability of non-linear systems consider stability of equilibrium points. One approach would then be to linearize the non-linear systems in all found equilibrium points and then use the theory described in Section 2.2. Stability of equilibrium points are relevant in control theory since a control system often is designed to keep the closed system near a stable equilibrium point. For the application considered in this thesis, i.e. non-linear residual generators, local stability of equilibrium points are not satisfying enough since it do not say anything about global stability of the system. Another approach is to analyze stability with Lyapunov theory, see for example [10]. The non-linear systems involved in the residual generators considered here, are complex and this is therefore considered hard and out of the scope of this thesis. Another, more realistic option, for stability investigation is to simulate the residual generator and then analyze the resulting data. A method for doing this has been developed at Scania and is described in [3]. The method uses data from an ETC test cycle and investigates stability of the solutions to the simulated system (8.1a) somewhat according to Definition 2.7. Some properties of the computed residual are also

concerned which means that the stability somehow is analyzed also according to Definition 8.1. The algorithm regards the following aspects:

- **Model drift:** check that states cross their own mean-value more than a certain number of times, depending on input data.
- **Feasibility:** verify that states are inside feasible regions during the entire simulation.
- **Correlation:** check that cross-correlation between input and the residual is not to high in the fault-free case.
- **Mean-error consistency:** check that the mean-value of the residual does not change too much when running the residual generator with different data.

If any of these tests fail, the residual generator is classed as unstable and hence discarded.

## 8.2 Stabilizing Residual Generators utilizing Observer Theory

Some of the residual generators extracted with the Scania Method are found to be unstable after investigation with the method described in Section 8.1.2. During execution of residual generators, the presence of model and measurement noise, disturbances or errors may cause a stable residual generator to behave like an unstable residual generator. As said in Section 8.1.1, this may be related to the robustness of the residual generator. An unstable residual is not suited for diagnosis since it may not detect faults like proposed or even stay close to zero in the fault-free case. A method for creating stable residual generators have been developed and will here be described.

Consider a residual generator

$$\dot{x} = f(x, z) \tag{8.4a}$$
$$r = g(x, z). \tag{8.4b}$$

If $z \in \mathcal{O}_{H_0}$, Definition 8.1 says that a stable residual generator should have properties so that the solution $x(t)$ to (8.4a) are stable for all initial conditions $x(0)$ and that $r \to 0$ when $t \to \infty$. Since $r = g(x, z)$, instability of $x(t)$ may cause the residual to drift away and not stay close to zero. One intuitive approach to design a stable residual generator is to use feedback of the residual $r$ along with equation (8.4a). A residual generator designed with this approach, based on (8.4), can be written as

$$\dot{\hat{x}} = f(\hat{x}, z) + \kappa(\hat{x}, z) \tag{8.5a}$$
$$r = g(\hat{x}, z) \tag{8.5b}$$

where $\kappa(\cdot)$ is some function of $\hat{x}$ and $z$. The main idea with the feedback approach is that if the residual $r$ starts to drift away, the dynamic equation (8.5a) will compensate this by calculating $\hat{x}$ so that the residual $r$ goes back to zero. The design is based on that the function $\kappa(\cdot)$ is chosen so that the residual generator gets this property. As mentioned earlier, the aim of residual generation is to produce a system that effectively detects and isolates faults, so $\kappa$ must be chosen with this in mind.

To be able to better analyze the problem, consider a linearization of (8.4) valid near an equilibrium point $(\bar{x}, \bar{z})$

$$\dot{x} = Ax + Bz \tag{8.6a}$$
$$r = Cx + Dz. \tag{8.6b}$$

A residual generator based on (8.6), designed according to the approach described above and with the non-linear function $\kappa$ replaced with the matrix $K$, is

$$\dot{\hat{x}} = A\hat{x} + Bz + Kr \tag{8.7a}$$
$$r = C\hat{x} + Dz \tag{8.7b}$$

which can be written as

$$\dot{\hat{x}} = (A + KC)\hat{x} + (B + KD)z \tag{8.8a}$$
$$r = C\hat{x} + Dz. \tag{8.8b}$$

The problem of choosing a non-linear function $\kappa$ is now instead the less complex problem of choosing a matrix $K$, so that residual generator (8.8) is stable. This is a well studied problem for control theory applications, see Section 2.3. The type of system (8.8) is then called an observer. The methods for calculating the matrix $K$ considered in Section 2.3 are

1. Eigenvalue Assignment
2. The Kalman Filter
3. Extended Kalman Filter
4. Constant Gain Extended Kalman Filter

Eigenvalue Assignment, Section 2.3.1, does not solve the problem completely since we instead have to choose where to place the eigenvalues. As said in Section 2.2.1, eigenvalues to the matrix $A + KC$ corresponding to poles in the left hand side of the complex plane guarantees stability of (8.7). The Kalman Filter, Section 2.3.2, requires a stochastic model of the system or that the matrices $Q$ and $R$ are used as design variables. The Kalman Filter approach, on the other hand, guarantees stability of (8.7). The Extended Kalman Filter, Section 2.3.3, would be a good candidate to create a stable residual generator for (8.4), since it handles non-linear systems. Due to the heavy computational burden required, this is not a realistic option because the residual generator will be implemented in a real-time system. Since the original residual generator (8.4) is non-linear and The Kalman Filter gain matrix $K$ can be easily calculated with MATLAB, a design approach based on the Constant Gain Extended Kalman Filter, Section 2.3.3, is chosen in this thesis.

### 8.2.1 Design Method

The method developed here for creating stable residual generators only handles a specific class of (8.4) where the function $g(x, z)$ is linear, and the residual generator can be written as

$$\dot{x} = f(x, z) \tag{8.9a}$$
$$r = Cx + Dz. \tag{8.9b}$$

This delimitation is done since a residual in many applications is a comparison of a sensor value $y$ and states $x$ on the form $r = y - \tilde{C}x$, which can be written as (8.9) with $z = \begin{bmatrix} u & y \end{bmatrix}^T$, $C = -\tilde{C}$ and $D = \begin{bmatrix} 0 & 1 \end{bmatrix}$. The different steps in the design process will now be further described.

**Searching for Equilibrium Points:** First, the non-linear system (8.9a) is searched for equilibrium points. Since the system is complex and hard to analyze analytically, it is done numerically. To be able to do this, a Simulink implementation of (8.9) is used. From the data structures created when searching for residual generators with The Scania Method, Simulink implementations of all found residual generators can be generated. The function `findop` in Simulink Control Design is used to find operating points. The function finds equilibrium points to a Simulink-model near a specified starting point by using optimization methods. To find equilibrium points in the region where the residual generator will be run, the starting values of the state variables $x$ and variables $z$ to `findop` are picked from measurement data.

**Linearization:** The system (8.9a) is linearized in every equilibrium point $(\bar{x}_i, \bar{z}_i)$, $i = 1, \ldots, M$, where M is the number of equilibrium points found. The linearization is done with the Matlab function `linearize`. To specify the equilibrium points, the function uses the values of variables $\bar{x}$ and $\bar{z}$ that were saved in the previous step. The linearizations can be written as

$$\dot{x} = A_i x + B_i z, \qquad i = 1, \ldots, M$$

and a linear approximation of the non-linear system (8.9a) for every equilibrium point is now available.

**Computation of Feedback Gain:** For every equilibrium point $(\bar{x}_i, \bar{z}_i)$ a Kalman gain is computed according to Section 2.3.2. The matrices $Q$ and $R$ are used as design parameters and tuned by hand so that the residual generator achieves satisfactory diagnostic performances. In Matlab, the computation of Kalman gain is done with the function `lqe`. How the feedback-gain $K$ and placement of the poles affects the performance of the residual generator is further investigated in Chapter 9.

**Gain Switching:** When the residual generator is run, the feedback gain corresponding to the operating conditions closest to present conditions will be used. All computed Kalman gains are stored in a look-up table and therefore

a look-up key must be chosen. In other words, we need a function, $\sigma(\cdot)$, that on the basis of present operating conditions gives the index to the most suitable Kalman gain. This look-up key can for example be a function of states and known variables. The main properties of the function $\sigma$ is that it must assign a unique integer value $i \leq M$ for all possible operating conditions. In this thesis $\sigma$ is chosen to be a function of a single known variable $z_s$

$$\sigma : z_s \rightarrow i \qquad z_s \in \mathbb{R},\, i \in [1, \ldots, M]\,.$$

The signal $z_s$ must characterize present operating conditions well and hence, be carefully chosen.

A residual generator created with the method described above, can finally be written as

$$\begin{aligned} \dot{\hat{x}} &= f(\hat{x}, z) + K_i r \\ r &= C\hat{x} + Dz \end{aligned}$$

where $i = \sigma(z_s)$, $i \in [1, \ldots, M]$, $z_s \in \mathbb{R}$ and $z_s \subseteq z$.

**Example 8.2**

Consider again the non-linear residual generator in Example 8.1

$$\dot{x} = x^2 - 1 \tag{8.10a}$$
$$r = y - x. \tag{8.10b}$$

As seen in Example 8.1, a bad initial condition caused instability of (8.10). A residual generator based on (8.10) will now be designed, according to the method proposed in Section 8.2.1.

### Finding Equilibrium Points

This was done in Example 8.1, the system (8.10a) has the equilibrium points $\bar{x}_1 = 1$ and $\bar{x}_2 = -1$.

### Linearization

This was also done in Example 8.1, the linearization valid near $\bar{x}_1$ can be written as

$$\dot{x} = 2x$$

with the corresponding system matrices

$$A_1 = 2, \quad B_1 = 0, \quad C_1 = 1, \quad D_1 = 0$$

where $C$ and $D$ originates from the linear equation (8.2b) in Example 8.1. The linerization valid near $\bar{x}_2$ can be written as

$$\dot{x} = -2x$$

and the corresponding system matrices are

$$A_2 = -2, \quad B_2 = 0, \quad C_2 = 1, \quad D_2 = 0.$$

### Computation of Feedback Gain

Different values of the covariance matrices $Q$ and $R$ was tested, finally $Q = 100$ and $R = 1$ was chosen since this gave the residual generator satisfying performances. Following command was used in MATLAB

```
K1 = lqe(A1,1,C1,Q,R,0)
K2 = lqe(A2,1,C2,Q,R,0)
```

This gave $K_1 = 8.1980$ and $K_2 = 2.1980$.

### Gain Switching

Since the model (8.2) in Example 8.1 says that $y = x$, the known signal $y$ characterizes present operating conditions and is chosen as switch signal. If $y$ is near 1, the matrix $K_1$ is chosen and if $y$ is near -1, matrix $K_2$ is chosen. This is implemented with the block `Direct Lookup Table` in SIMULINK.

The residual generator can now be written as

$$\dot{\hat{x}} = \hat{x}^2 - 1 + K_i r \tag{8.11a}$$

$$r = y - \hat{x} \tag{8.11b}$$

where $i = 1$ near the operating point $\bar{x}_1$ and $i = 2$ near $\bar{x}_2$. An implementation of (8.11) in SIMULINK is shown in Figure 8.3 and 8.4.

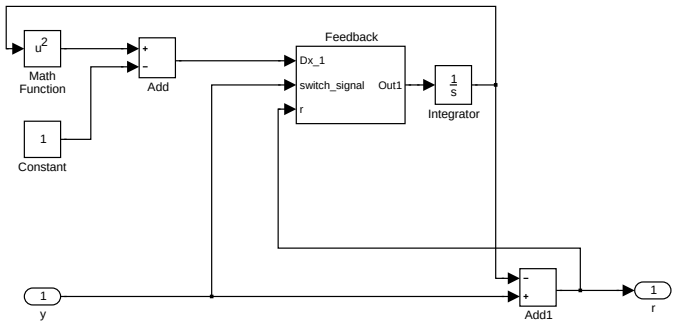



**Figure 8.3.** SIMULINK implementation of residual generator (8.11). The block `Feedback` is shown in Figure 8.4

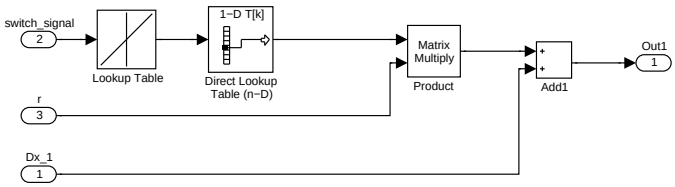



**Figure 8.4.** The block `Feedback` in the SIMULINK implementation of residual generator (8.11) shown in Figure 8.3

Simulation of the residual generator with initial conditions $x_0 = -1$, $x_0 = -3.5$, $x_0 = 1.5$ and a constant $y = -1$ with added simulated noise, was performed. The result is shown in 8.5, which should be compared to Figure 8.2. According to Figure 8.5, the residual $r$ approaches zeros even if a bad initial condition is chosen and the state $x$ does not drift away as in Figure 8.2. Residual generator (8.11) seems to be stable for $y = -1$, since initial condition $x_0 = 1.5$ does not cause instability of (8.11). This confirms that the method for constructing stable residual proposed in Section 8.2.1 is working well in this case.

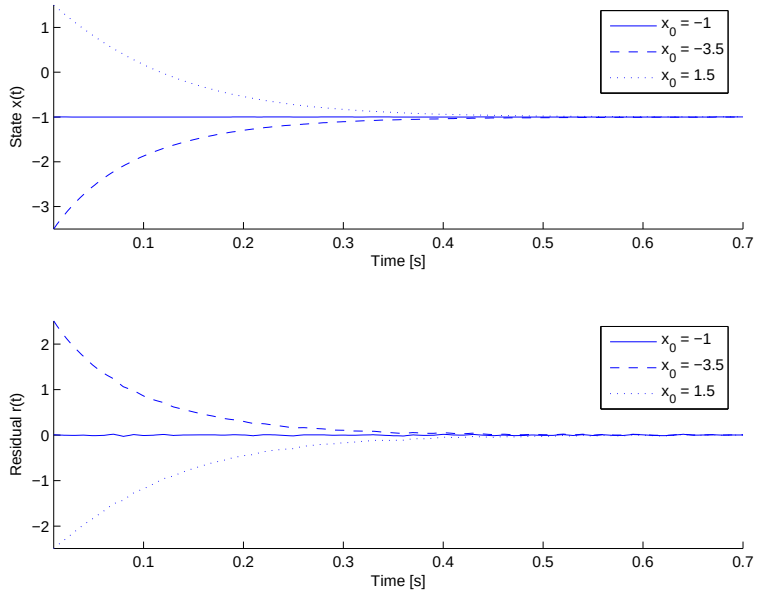



**Figure 8.5.** Simulation of residual generator (8.11) for initial conditions $x_0 = -1$, $x_0 = -3.5$ and $x_0 = 1.5$ and constant $y = -1$. The residual generator is stable for all three initial conditions.

### 8.2.2 Stability Analysis

Consider again a residual generator designed according to the method in Section 8.2.1

$$\dot{\hat{x}} = f(\hat{x}, z) + Kr \tag{8.12a}$$
$$r = C\hat{x} + Dz. \tag{8.12b}$$

According to the stability definitions presented, the residual generator (8.12) is stable if

1. The solution $x(t)$ to (8.12a) is stable, according to Definition 2.7 , for all $x_0$

2. $\lim_{t\to\infty} r(t) = 0$

for all $z \in \mathcal{O}_{H_0}$. To analyze and investigate the stability of (8.12), the residual generator needs to be simulated for all possible $x_0 \in \mathbb{R}$ and for all $z \in \mathcal{O}_{H_0}$ to see if the conditions above is fulfilled. This may be a complex and time consuming task. Another way to investigate stability is to analyze (8.12) with the method described in Section 8.1.2. If a residual generator is classed as stable after this investigation, it is in this thesis considered as stable. Even if this is possible in most cases, it would be satisfactory to see if the design method proposed in Section 8.2.1 produces residual generators that are stable, and this will now be investigated. The analysis will be performed under the limitation that the residual generator is linear and can be described by

$$\dot{\hat{x}} = A\hat{x} + Bu + Kr \tag{8.13a}$$
$$r = y - \tilde{C}\hat{x} - \tilde{D}u. \tag{8.13b}$$

Note that (8.13b) can be written as (8.12b) with $z = \begin{bmatrix} u & y \end{bmatrix}^T$, $C = -\tilde{C}$ and $D = \begin{bmatrix} -\tilde{D} & 1 \end{bmatrix}$. Now, assume that residual generator (8.13) is based on the linear model

$$\dot{x} = Ax + Bu$$
$$y = \tilde{C}x + \tilde{D}u$$

valid under $H_0$ if $(u, y) \in \mathcal{O}_{H_0}$. With these limitations and the assumption, it will now be shown that residual generator (8.13) is stable according to Definition 8.3.

### Stability Condition 1

Residual generator (8.13) can be rewritten as

$$\dot{\hat{x}} = (A - K\tilde{C})\hat{x} + (B - K\tilde{D})u + Ky \tag{8.14a}$$
$$\hat{r} = y - \tilde{C}\hat{x} - \tilde{D}u \tag{8.14b}$$

and according to Section 2.2.1, (8.14a) is asymptotic stable if $\mathrm{Re}\{\lambda_i(A - K\tilde{C})\} < 0$.

### Stability Condition 2

Under the assumption that $H_0$ is valid and $(u, y) \in \mathcal{O}_{H_0}$, $y = \tilde{C}x + \tilde{D}u$, and residual generator (8.13) can be written as

$$\begin{bmatrix} \dot{\hat{x}} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} A - K\tilde{C} & K\tilde{C} \\ 0 & A \end{bmatrix} \begin{bmatrix} \hat{x} \\ x \end{bmatrix} + \begin{bmatrix} B \\ B \end{bmatrix} u \tag{8.15a}$$
$$r = \begin{bmatrix} -\tilde{C} & \tilde{C} \end{bmatrix} \begin{bmatrix} \hat{x} \\ x \end{bmatrix} \tag{8.15b}$$

With the transformation $\xi = \hat{x} - x$, (8.15) becomes

$$\dot{\xi} = (A - K\tilde{C})\xi \tag{8.16a}$$
$$r = -\tilde{C}\xi. \tag{8.16b}$$

If $\mathrm{Re}\{\lambda_i(A - K\tilde{C})\} < 0$, the system is asymptotically stable and since (8.16) is an autonomous system

$$\lim_{t\to\infty} r(t) = 0$$

Condition 1 and 2 in stability Definition 8.1 is fulfilled and hence (8.13) is a stable residual generator for $(u, y) \in \mathcal{O}_{H_0}$.

# Chapter 9

# Analysis of Stabilized Residual Generators

In this chapter, the properties of a linearized residual generator with feedback, designed according to Section 8.2.1, is analyzed. Both properties in the frequency and time domain is concerned.

## 9.1 Derivation of Transfer Functions

To investigate how faults in known signals $u$ and $y$ affects the residual, the faults must be included in the model. Consider a linearized model valid near an equilibrium point $(\bar{x}, \bar{u})$, where additive faults affecting $u$ and $y$ are called $f_u$ and $f_y$ respectively.

$$\begin{aligned} \dot{x} &= Ax + B(u + f_u) && \text{(9.1a)} \\ y &= Cx + f_y. && \text{(9.1b)} \end{aligned}$$

The additive faults affecting $u$ are called actuator faults $f_u$ and the faults affecting $y$ are called sensor fault

$$\begin{aligned} \dot{x} &= Ax + Bu && \text{(9.2a)} \\ r &= y - Cx. && \text{(9.2b)} \end{aligned}$$

s $f_y$. A residual generator based on (9.1) is A stable residual generator, based on (9.2), designed according to Section 8.2 is

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + Bu + Kr \\ r &= y - C\hat{x} \end{aligned}$$

$$\Longleftrightarrow$$

$$\begin{aligned} \dot{\hat{x}} &= (A - KC)\hat{x} + Bu + Ky && \text{(9.3a)} \\ r &= y - C\hat{x}. && \text{(9.3b)} \end{aligned}$$

By using $y = Cx + f_y$ and $Bu = \dot{x} - Ax - Bf_u$ from (9.1) and introducing $\xi = \hat{x} - x$ as the estimation error, residual generator (9.3) can be written as

$$\begin{aligned} \dot{\xi} &= (A - KC)\xi - Bf_u + Kf_y && \text{(9.4a)} \\ r &= f_y - C\xi. && \text{(9.4b)} \end{aligned}$$

Under the assumption that $\xi(t_0) = 0$ and by using the one-sided Laplace transform, (9.4) can be transformed to

$$\begin{aligned} \xi(s) &= (sI - A + KC)^{-1} (-Bf_u(s) + Kf_y(s)) \\ r(s) &= f_y(s) - C\xi(s) \end{aligned}$$

$$\Longleftrightarrow$$

$$r(s) = \begin{bmatrix} C(sI - A + KC)^{-1}B & I - C(sI - A + KC)^{-1}K \end{bmatrix} \begin{bmatrix} f_u \\ f_y \end{bmatrix}. \quad (9.5)$$

From (9.5) we can obtain the transfer functions from actuator faults $f_u$ to the residual $r$

$$G_{rf_u}(s) = C(sI - A + KC)^{-1}B \quad (9.6)$$

and from sensor faults $f_y$ to the residual $r$

$$G_{rf_y}(s) = I - C(sI - A + KC)^{-1}K. \quad (9.7)$$

Note, that the transfer function from, for example, fault $f_{u_i}$ to the residual $r$ is denoted $G_{rf_{u_i}}$.

## 9.2 Some Properties in the Frequency Domain

Some conclusions can be drawn by studying the properties of a residual generator in the frequency domain. First the frequency functions

$$G_{rf_u}(j\omega) = C(j\omega I - A + KC)^{-1}B$$

are considered. Note that the limit

$$\lim_{\omega \to \infty} G_{rf_{u_i}}(j\omega) = 0$$

The frequency functions

$$G_{rf_y}(j\omega) = I - C(j\omega I - A + KC)^{-1}K$$

instead have the limit

$$\lim_{\omega \to \infty} G_{rf_{y_i}}(j\omega) = 1$$

For all stabilized residual generators studied in this thesis, $G_{rf_{u_i}}(0) > 0$ and $G_{rf_{y_i}}(0) < 1$. Therefore the corresponding frequency functions $G_{rf_{u_i}}(j\omega)$ and $G_{rf_{y_i}}(j\omega)$ have *low-pass* respective *high-pass* characteristics.

It can be noted that if no feedback is used, $K = 0 \Rightarrow G_{rf_y}(j\omega) = I$, this means that if a residual generator with feedback is used and $G_{rf_{y_i}}(0) < I$, the ability to detect the sensor fault $f_{y_i}$ is decreased due to the change in characteristics of $G_{rf_{y_i}}$ caused by the feedback. The difference in the frequency domain can also be seen in the time domain and results in that the step-responses for faults $f_{u_i}$ and $f_{y_i}$ will have different characteristics, this is illustrated in the following example.

**Example 9.1**

Consider an unstable linear residual generator

$$\begin{aligned} \dot{x} &= 0.1x + u \quad (9.8a) \\ r &= y - x. \quad (9.8b) \end{aligned}$$

A stable residual generator based on (9.8), designed according to Section 8.2.1 is

$$\begin{aligned} \dot{\hat{x}} &= 0.1\hat{x} + u + Kr \\ r &= y - \hat{x}. \end{aligned}$$

If $f_u$ is an additive fault in the actuator $u$ and $f_y$ an additive fault in the sensor $y$, the transfer functions from $f_u$ and $f_y$ to the residual $r$ are given by

$$\begin{aligned} G_{rf_u}(s) &= \frac{1}{s - 0.1 + K} \\ G_{rf_y}(s) &= \frac{s - 0.1}{s - 0.1 + K} \end{aligned}$$

where $K > 0.1$ ensures stability. If $K = 2.1$ is chosen then

$$\begin{aligned} G_{rf_u}(s) &= \frac{1}{s + 2} \\ G_{rf_y}(s) &= \frac{s - 0.1}{s + 2}. \end{aligned}$$

This means that $G_{rf_u}(0) = 0.5$ and $G_{rf_y}(0) = -0.05$. Furthermore, $G_{rf_u}(0) > 0$ and $G_{rf_y}(0) < 1$, and $G_{rf_u}(j\omega)$ have low-pass and $G_{rf_y}(j\omega)$ high-pass characteristics. A Bode magnitude plot of $G_{rf_u}(s)$ and $G_{rf_y}(s)$ is shown in Figure 9.1 and in Figure 9.2 responses for steps in the fault signals $f_u$ and $f_y$ are shown. It is clear that the transfer functions have different characteristics and that there is a significant difference between the two step-responses.

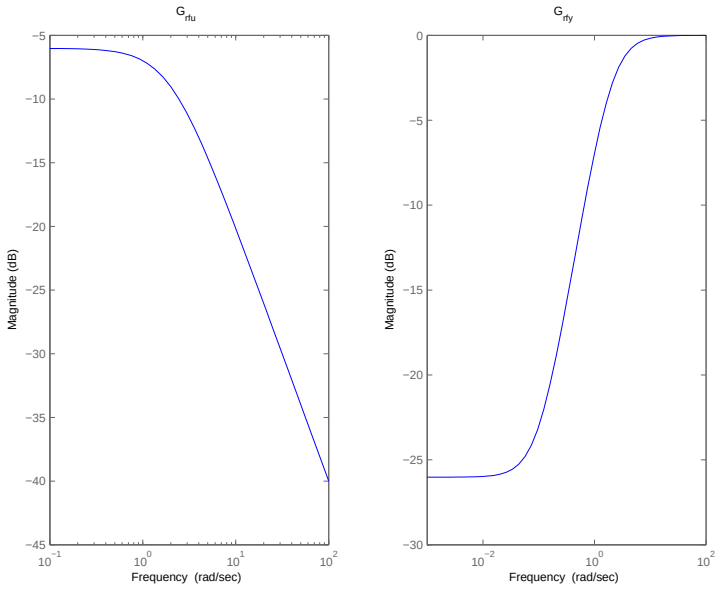



**Figure 9.1.** A Bode magnitude plot of $G_{rf_u}(s)$ (to the left) and $G_{rf_y}(s)$ (to the right). It is clear that $G_{rf_u}(s)$ have low-pass and $G_{rf_y}(s)$ high-pass characteristics.

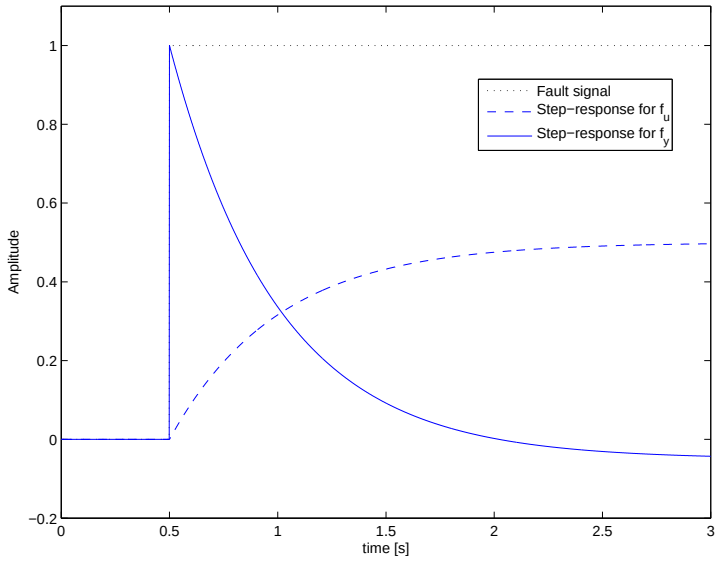



**Figure 9.2.** Responses for steps in fault signals $f_u$ (dashed) and $f_y$ (solid). There is a significant difference between the two step-responses.

## 9.3 Detectability Analysis

As said in Section 3.3.1, a fault $f$ is strongly detectable in the residual $r$ if $G_{rf}(0) \neq 0$. If $K \neq 0$, the transfer functions are described by (9.6) and (9.7) and if $s = 0$ we have

$$G_{rf_u}(0) = C\left(KC - A\right)^{-1} B \tag{9.9}$$
$$G_{rf_y}(0) = I - C\left(KC - A\right)^{-1} K. \tag{9.10}$$

Fault $f_{u_i}$ is strongly detectable if

$$\left\{C\left(KC - A\right)^{-1} B\right\}_i \neq 0$$

where $\{\bar{v}\}_i$ denotes element $i$ in the vector $\bar{v}$. Fault $f_{y_i}$ is strongly detectable if

$$\left\{C\left(KC - A\right)^{-1} K\right\}_i \neq 1.$$

This means that the feedback gain $K$ can be used to change the detectability properties for a residual generator. However, there are no guarantees that $K$ can be chosen so that all faults $f_u$ and $f_y$ are strongly detectable.

If the residual generator does not use any feedback at all, i.e. $K = 0$, the transfer functions (9.9) and (9.10) becomes

$$G_{rf_u}(0) = -CA^{-1}B$$
$$G_{rf_y}(0) = I.$$

One can see that fault $f_{u_i}$ is strongly detectable if

$$\left\{CA^{-1}B\right\}_i \neq 0$$

and that fault $f_{y_i}$ always is strongly detectable.

### 9.3.1 Static Gains

If the static gain from a specific fault to residual is close to zero, a constant fault may be hard to detect, even if the fault is strongly detectable. The static gains for actuator faults $f_u$ is, as said above, given by

$$G_{rf_u}(0) = C\left(KC - A\right)^{-1} B$$

and for sensor faults $f_y$ by

$$G_{rf_y}(0) = I - C\left(KC - A\right)^{-1} K$$

The static gain is highly dependent on the choice of feedback gain $K$. If $K = 0$, the static gains are $-CA^{-1}B$ and $I$ for $G_{rf_u}$ and $G_{rf_y}$ respectively. A large $K$ will,

for example, make $G_{rf_u}(0)$ small and therefore $f_u$ less detectable. This means that the static gains can be modified by using feedback in the residual generator and this must be considered when choosing the feedback gain $K$. How the properties of a residual generator depends on the choice of feedback gain is illustrated with an example.

**Example 9.2**

Consider again the model (9.8). In Example 9.1, a stable residual generator was designed and the transfer functions from faults $f_u$ and $f_y$ to the residual $r$ are given by

$$\begin{aligned} G_{rf_u}(s) &= \frac{1}{s-0.1+K} \\ G_{rf_y}(s) &= \frac{s-0.1}{s-0.1+K}. \end{aligned}$$

To ensure stability $K > 0.1$ must be chosen. How the Bode magnitude plots of $G_{rf_u}(s)$ and $G_{rf_y}(s)$ and the responses for steps in fault signals $f_u$ and $f_y$ depends on the feedback gain $K$, are shown in Figure 9.3, 9.4 and 9.5.

It is obvious that the static gain is highly dependent on the choice of $K$ and that there are significant differences in the step-responses for different values of $K$. A larger $K$ will make the residual generator fast but less sensitive for constant faults since the static gain is small. A smaller value on $K$ will result in a larger static gain but the residual generator will not be that fast and the $K$ may not be enough to stabilize the residual generator.

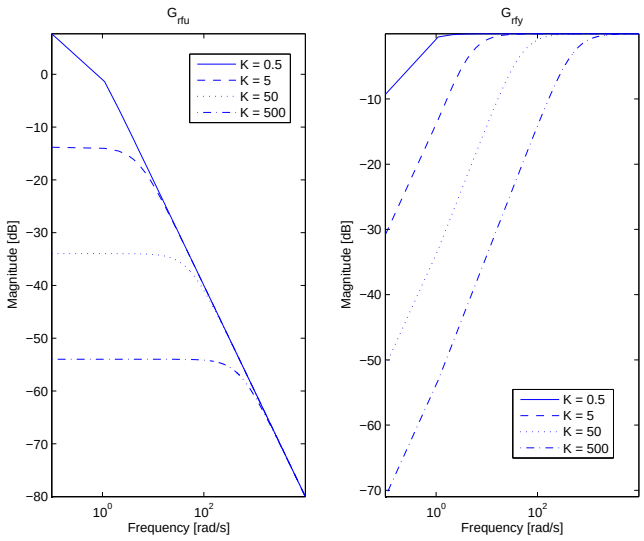



**Figure 9.3.** A bode magnitude plot of $G_{rf_u}(s)$ (to the left) and $G_{rf_y}(s)$ (to the right) for different values of the feedback gain $K$. The static gain is highly dependent on the choice of $K$.

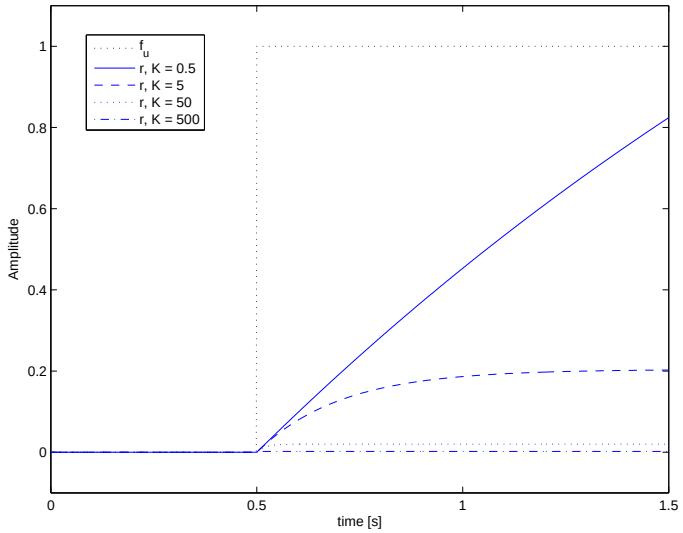



**Figure 9.4.** Responses for steps in the fault signal $f_u$ for different values of the feedback gain $K$. There are significant differences in the step-responses for different values of $K$.

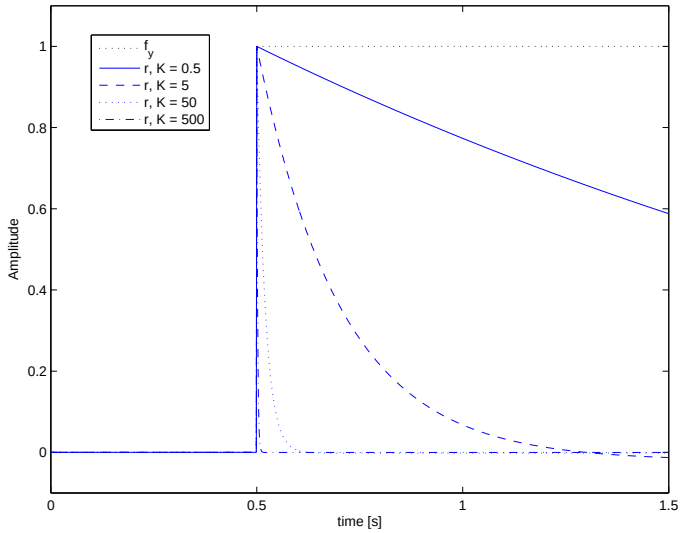



**Figure 9.5.** Responses for steps in the fault signal $f_y$ for different values of the feedback gain $K$. There are significant differences in the step-responses for different values of $K$.

## 9.4 Discussion

If the frequency functions $G_{rf_y}(j\omega)$ have high-pass characteristics, the response for a sudden fault $f_y$ will peak and then decrease to a small value which makes the fault hard to detect. This is a great disadvantage when using feedback in a residual generator, and is illustrated in Example 9.1 and 9.2.

The choice of feedback gain $K$ mainly affects the gain from fault to residual and the speed of the residual generator. A large $K$ will result in fast dynamics and the eigenvalues to the matrix $A-KC$ will be large and negative. The residual generator will respond fast to changes in the measurements $y$ but will on the other hand, be sensitive for disturbances affecting the measurements. A large $K$ will also make the static gains for the transfer functions $G_{rf_u}$ and $G_{rf_y}$ small and this will make the residual generator less sensitive for constant faults $f_u$ and $f_y$ and hence, decrease the detectability properties.

A small $K$ may, first of all, not be sufficient for stabilizing an unstable residual generator. It will also lead to slower dynamics of the residual generator so that the response to faults in the measurements $y$ will be slow. The response to abrupt faults $f_u$ and $f_y$ may be slow but the static gains for the transfer functions $G_{rf_u}$ and $G_{rf_y}$ will, on the other hand be larger than with a large $K$.

There is no guarantee that one specific $K$ will give a stabilized residual generator, designed according to Section 8.2.1, satisfying properties concerning both sensor and actuator faults. One approach that can be used, is to design different stable residual generators based on one given unstable residual generator. Assume that we have an unstable residual generator that requires a $K > \epsilon$ to ensure stability. We then design one residual generator with $K_1 > \epsilon$ with appropriate properties concerning actuator faults and one residual generator with $K_2 > \epsilon$, $K_2 \neq K_1$ with appropriate properties concerning sensor faults. This will give us two stable residual generators

$$\begin{aligned} \dot{\hat{x}}_1 &= f(\hat{x}_1, z) + K_1 r_1 \\ r_1 &= C\hat{x}_1 + Dz \end{aligned}$$

and

$$\begin{aligned} \dot{\hat{x}}_2 &= f(\hat{x}_2, z) + K_2 r_2 \\ r_2 &= C\hat{x}_2 + Dz \end{aligned}$$

Of course this approach can be extended and one residual generator can be designed for every fault that the original residual generator, is sensitive for.

# Chapter 10

# Evaluation of Stabilized Residual Generators

For a specific engine model and a set of measurement data, 70 potential residual generators were found with The Scania method. The stability investigation algorithm, mentioned in Section 8.1.2, discarded 18 of those residual generators due to instability. Most of those 18 can not be simulated at all, because they make the simulation software crash. In this chapter, two unstable residual generators with different properties are considered. One of the residual generators can be simulated, but the corresponding residual is far from zero in the none-faulty case. The other residual generator can only be simulated for a short time since the residual approaches infinity in the none-faulty case. The design method proposed in Section 8.2.1 is used to stabilize these residual generators which are then simulated and evaluated.

## 10.1 Residual Generator 1

One specific residual generator can be simulated but the performance of the residual is poor. The residual generator have three states, uses eleven known signals and can be written as

$$\dot{x} = f(x, u) \tag{10.1a}$$
$$r = y - Cx \tag{10.1b}$$

where

$$x = \begin{bmatrix} \hat{p}_{\text{es}} \\ \hat{p}_{\text{cmp}} \\ \hat{n}_{\text{trb}} \end{bmatrix}, \qquad u = \begin{bmatrix} p_{\text{em}} \\ p_{\text{im}} \\ w_{\text{cmp}} \\ \alpha \\ \delta \\ n_{\text{avg}} \\ p_{\text{amb}} \\ t_{\text{amb}} \\ t_{\text{im}} \\ u_{\text{vgt}} \end{bmatrix}, \qquad y = n_{\text{trb}}, \qquad C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}.$$

### 10.1.1 Implementation in Simulink

As said in Chapter 4, the Scania Method starts with a SIMULINK model when searching for residual generators. From the data structures created during this procedure, SIMULINK implementations of all residual generators can be generated. A SIMULINK implementation of the residual generator considered here, is shown in Figure 10.1 - 10.2. When simulating the residual generator, a sequence of measured data from a truck engine is used. The data is fed to the SIMULINK model in Figure 10.1 from the block `Insignals u` in Figure 10.2(a) where the data is imported from the MATLAB workspace with the block type `From Workspace`. The block `System f(x,u)`, is automatically generated from the data structures created during the extraction of residual generators from the engine model, therefore it looks a bit messy and is not shown. The integration of the result from the block `System f(x,u)` is done in the block `Integrators` shown in Figure 10.2(b). The simulation results are exported to the MATLAB workspace with the block type `To Workspace`.

### 10.1.2 Design of Stable Residual Generators

By using the method described in Section 8.2, two different stable residual generators based on (10.1) were designed for comparison. Each step will now be further described.

#### Searching for Equilibrium Points

By using the the MATLAB function `findop` and the SIMULINK implementation shown in Figure 10.1, the model (10.1a) was searched for equilibrium points. The initial values of the state $\hat{n}_{\text{trb}}$ and all inputs $u$ were picked from available measurement data. Initial values for the states $\hat{p}_{\text{es}}$ and $\hat{p}_{\text{cmp}}$ were set to $1.2 \cdot 10^5$ and $3.2 \cdot 10^5$ respectively, which was considered as reasonable values. In total, values from 111 points in the measured data were used and 22 operating points were found. For example, one equilibrium point called *operating point 9*, were

$$\bar{x}_9 = \begin{bmatrix} 1.185{\cdot}10^5 & 3.2{\cdot}10^5 & 9.198{\cdot}10^4 \end{bmatrix}$$
$$\bar{u}_9 = \begin{bmatrix} 3.331{\cdot}10^5 & 3.167{\cdot}10^5 & 0.1942 & 2.573 & 151.4 & 1232 & 1.023{\cdot}10^5 & 192.9 & 339.8 & 45.51 \end{bmatrix}.$$

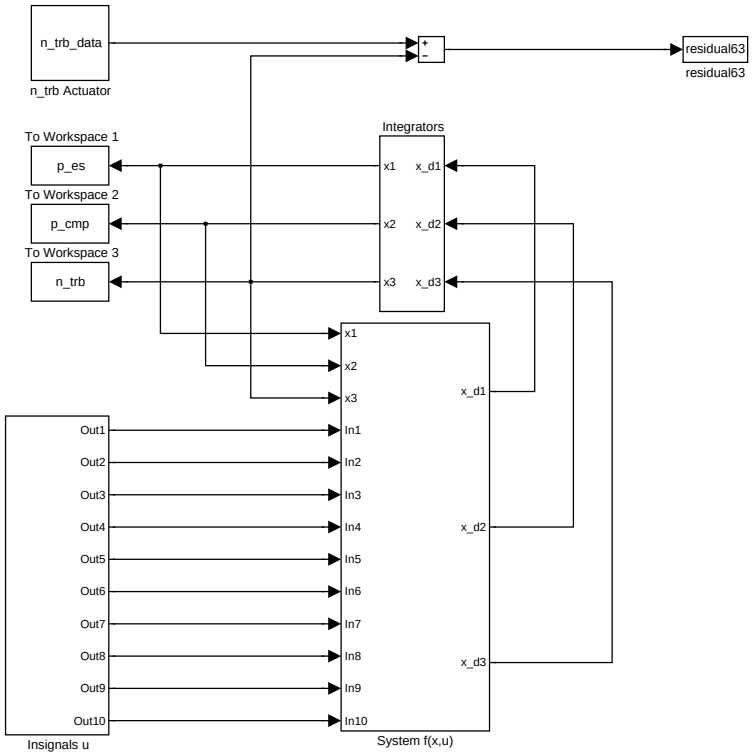
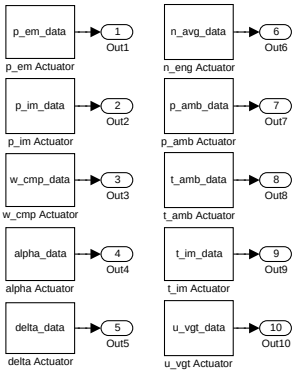
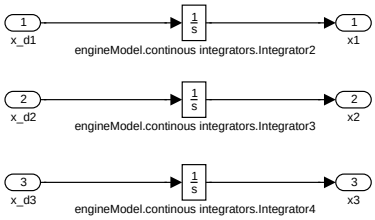



**Figure 10.1.** SIMULINK implementation of residual generator 1.




(a) The block `Insignals u`.




(b) The block `Integrators`.

**Figure 10.2.** SIMULINK implementation of the blocks `Insignals u` and `Integrators` in Figure 10.1.

### Linearization

The MATLAB function `linearize` was then used to linearize residual generator 1 in all equilibrium points found in the previous step. In operating point 9, the linearization of (10.1) were

$$\begin{aligned} \dot{w} &= Aw + Bv \\ r &= y - Cv \end{aligned}$$

where $w = x - \bar{x}_9$, $v = u - \bar{u}_9$,

$$A = \begin{bmatrix} -2.564 & 0 & -0.2748 \\ 0 & -97.43 & -0.6316 \\ -0.2453 & -0.1559 & 12.44 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.1295 & -0.01844 & 0 & 7.607 & 46.58 & 1.249 & 2.385 & 1.1 & 34.47 & 107.4 \\ 0 & 96.92 & 3.33 \cdot 10^6 & 0 & 0 & 0 & -1.57 & 1676 & 0 & 0 \\ 0.1857 & 0.797 & -1.731 \cdot 10^5 & -328.8 & -2013 & -53.98 & 0.4875 & -221.9 & -1490 & -6706 \end{bmatrix}$$

The corresponding eigenvalues are $\lambda_1 = -2.5685$, $\lambda_2 = 12.4460$ and $\lambda_3 = -97.4305$. Since $\mathrm{Re}\{\lambda_2\} > 0$, operating point 9 is not a stable equilibrium point. In total, 8 of the 22 operating points were found to be unstable. Note that this is a bit strange, since the residual generator is extracted from a stable engine model. Though this explains why the behavior of the residual is unstable.

### Computation of Feedback Gain

Kalman feedback gains were computed with the MATLAB function `lqe`. The covariance matrices $Q$ and $R$ were used as design parameters and two different Kalman gains were computed for every operating point, one for each residual generator. A set of Kalman gains were computed with the covariance matrices

$$Q = \begin{bmatrix} 10^{-3} & 0 & 0 \\ 0 & 10^{-3} & 0 \\ 0 & 0 & 10^{-3} \end{bmatrix} \text{ and } R = 1,$$

the residual generator using those was called *residual generator 1a*. The other set of Kalman gains were computed with the covariance matrices

$$Q = \begin{bmatrix} 10^{3} & 0 & 0 \\ 0 & 10^{3} & 0 \\ 0 & 0 & 10^{3} \end{bmatrix} \text{ and } R = 1,$$

the corresponding residual generator was called *residual generator 1b*. The covariance matrix $N$ and the matrix $G$, see Section 2.3.2, used in the calculation was set to

$$N = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \qquad G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

since the system and measurement noises affecting the residual generator were assumed to be uncorrelated. The Kalman gains computed for operating point 9 were

$$K_9 = \begin{bmatrix} -0.4557 \\ -0.1431 \\ 24.8921 \end{bmatrix}$$

for residual generator 1a and

$$K_9 = \begin{bmatrix} -1.6549 \\ -0.2293 \\ 46.4355 \end{bmatrix}$$

for residual generator 1b.

### Gain Switching

All computed Kalman gains were stored in a matrix, used as a look-up table in SIMULINK. As switch signal or look-up key for the table, the input signal $\delta$ was used. The reason for choosing $\delta$ is that it has a unique value in all operating points. The value of $\delta$ in every operating point is plotted in Figure 10.3.

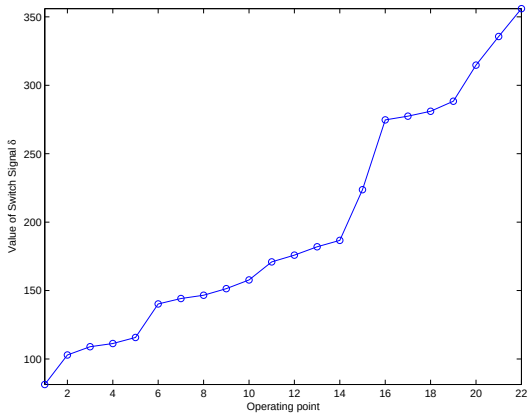



**Figure 10.3.** The value of the input signal $\delta$ plotted for every operating point. The curve is monotone, which means that $\delta$ has one unique value for every operating point and hence well suited as look-up key for gain switching.

The stabilized residual generators 1a and 1b can both be written as

$$\dot{\hat{x}} = f(\hat{x}, u) + K_i r \tag{10.2a}$$
$$r = n_{\text{trb}} - Cx \tag{10.2b}$$

where $i = \sigma(\delta)$, $i \in [1, \ldots, 22]$. Both residual generator 1a and 1b were implemented in SIMULINK as shown in Figures 10.4 and 10.5. The only difference between the two residual generators are the matrix containing the feedback gains.

The blocks `Insignals` and `Integrators` can be found in Figure 10.2(a) and 10.2(b). The Simulink block `Lookup Table` performs a linear interpolation of the switch signal $\delta$ and converts it to an integer $i \leq 22$ that specifies the current operating point. The integer $i$ is then used to select a feedback gain $K_i$ from the block `Direct Lookup Table (n-D)`.

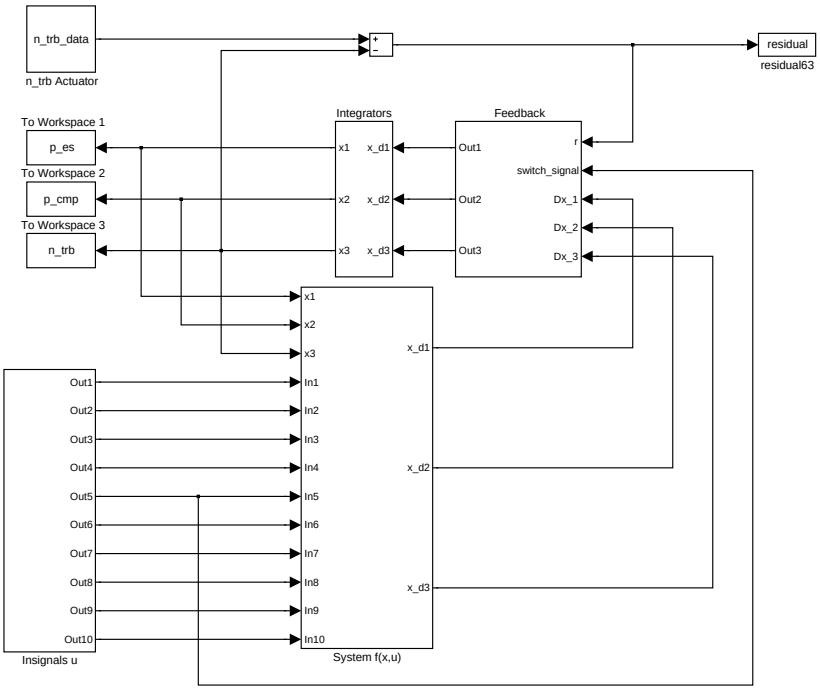



**Figure 10.4.** SIMULINK implementation of the stabilized residual generators 1a and 1b described by (10.2).

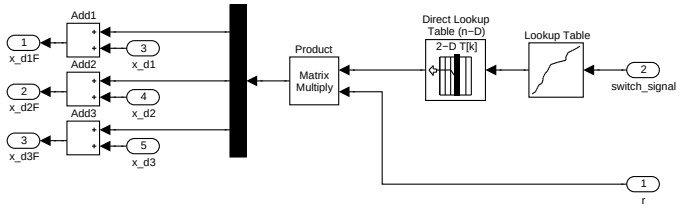



**Figure 10.5.** SIMULINK implementation of the block `Feedback` in Figure 10.4.

### 10.1.3 Fault-Free Simulation

To evaluate the performance of the three residual generators 1, 1a and 1b in the fault-free case, they were simulated in SIMULINK. As input data for the simulation, non-faulty measurement data from a truck engine were used. The input signals can be found in Figure 10.6.

#### Residual Generator 1

The three states $\hat{p}_{\text{es}}$, $\hat{p}_{\text{cmp}}$ and $\hat{n}_{\text{trb}}$ are shown in Figure 10.7 and the residual, $r = n_{\text{trb}} - \hat{n}_{\text{trb}}$, in Figure 10.8. By studying the figure, one can see that the residual is affected by the instability of residual generator 1. If residual generator 1 would have been stable, the computed state $\hat{n}_{\text{trb}}$ (solid line) and measured signal $n_{\text{trb}}$ (dotted line) in Figure 10.7 would have been more alike. Now, there is a significant difference between $\hat{n}_{\text{trb}}$ and $n_{\text{trb}}$, which causes the residual to be far away from zero, as can be seen in Figure 10.8.

#### Residual Generator 1a and 1b

The three states of residual generator 1a and 1b is shown in Figure 10.9. By comparing Figure 10.9 with Figure 10.7, one can see that the third state $\hat{n}_{\text{trb}}$ in both residual generator 1a and 1b better follows the measured signal $n_{\text{trb}}$ than the third state in residual generator 1. In Figure 10.10 the residual is shown for both residual generators. The same comparison can be done between Figure 10.10 and Figure 10.8. The residuals from residual generator 1a and 1b are closer to zero and more 'calm' than the residual from residual generator 1.

By studying Figure 10.9 and 10.10, one can also see that residual generator 1b (solid line) better follows the measured signal than residual generator 1a (dashed line). Furthermore, the residual from residual generator 1b is more 'calm' and close to zero than the residual from residual generator 1a. Residual generator 1b seem to have better properties than residual generator 1a, at least in the fault-free case.

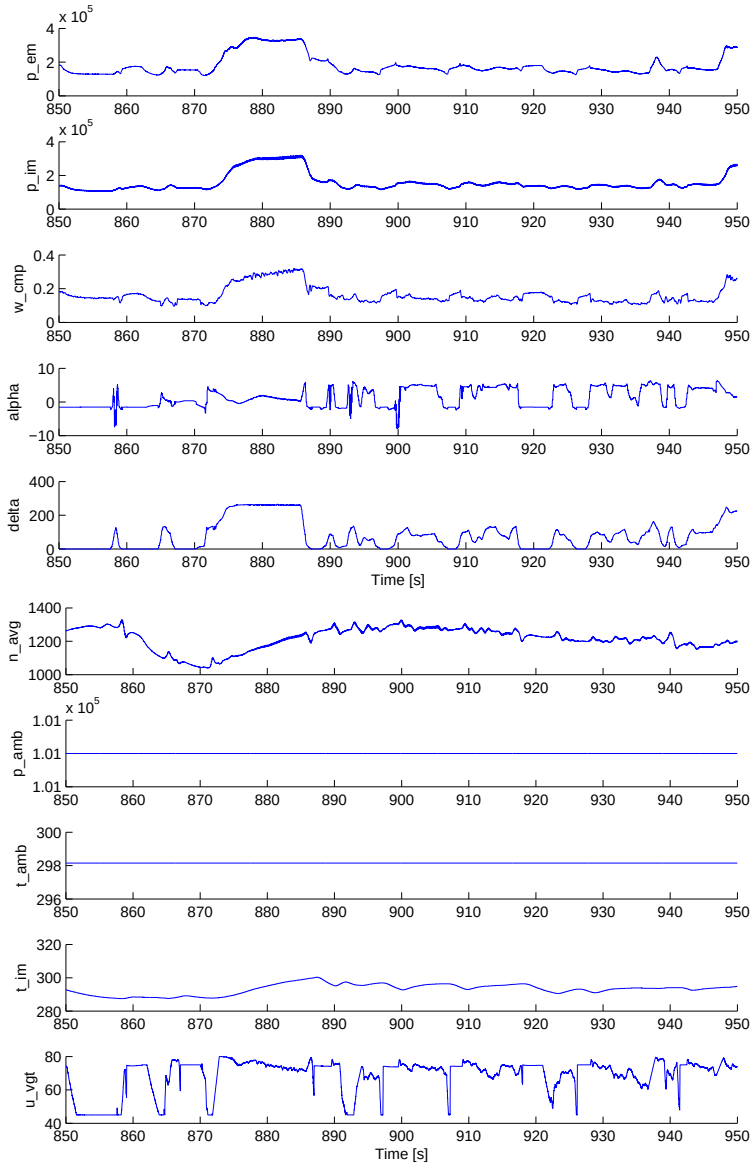



**Figure 10.6.** The input signals $u$ used in the fault-free simulation of residual generator 1, 1a and 1b.

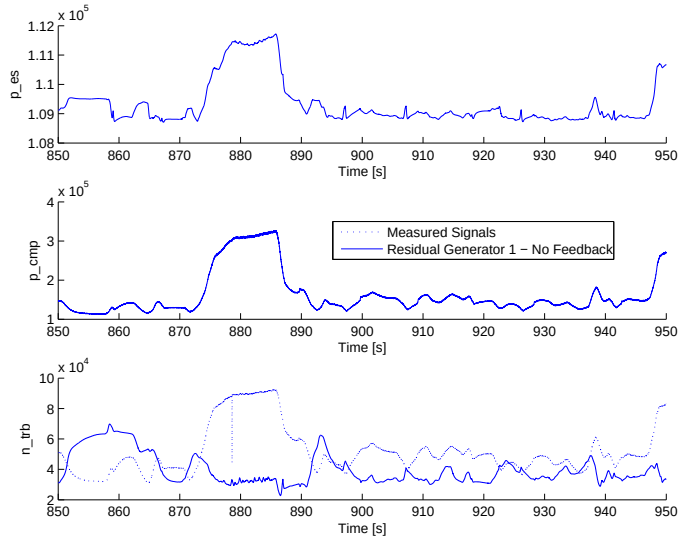



**Figure 10.7.** The states $\hat{p}_{\mathrm{es}}$, $\hat{p}_{\mathrm{cmp}}$ and $\hat{n}_{\mathrm{trb}}$ after the SIMULINK simulation of residual generator 1 using fault-free input data. There is a significant difference between the computed state $\hat{n}_{\mathrm{trb}}$ (solid line) and the measured signal $n_{\mathrm{trb}}$ (dotted line), this is a consequence of the instability of residual generator 1.

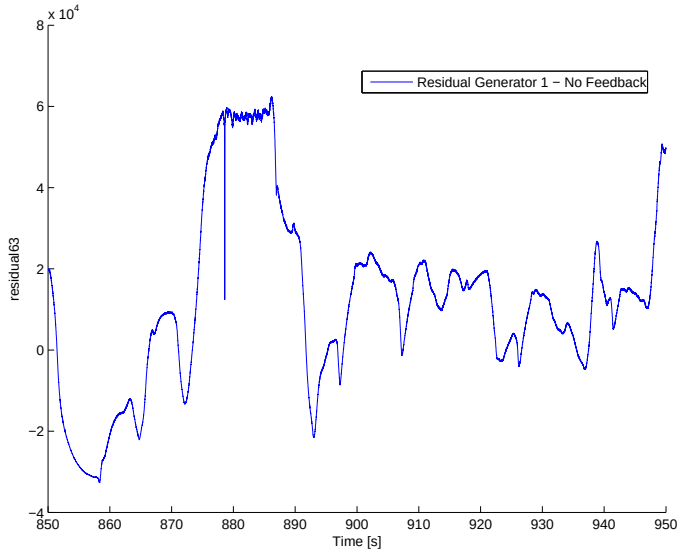



**Figure 10.8.** The residual $r = n_{\mathrm{trb}} - \hat{n}_{\mathrm{trb}}$ after the SIMULINK simulation of residual generator 1. Due to the instability, the residual is far away from zero and very 'alive'.

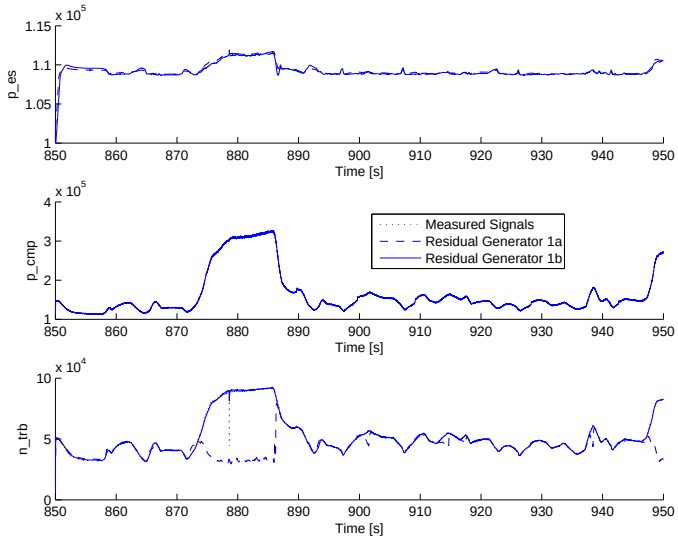



**Figure 10.9.** The three states $\hat{p}_{\mathrm{es}}$, $\hat{p}_{\mathrm{cmp}}$ and $\hat{n}_{\mathrm{trb}}$ after the SIMULINK simulation of residual generators 1a (dashed line) and 1b (solid line) using fault-free input data. The third state computed by residual generator 1b better follows the measured signal (dotted line) than the state from residual generator 1a.

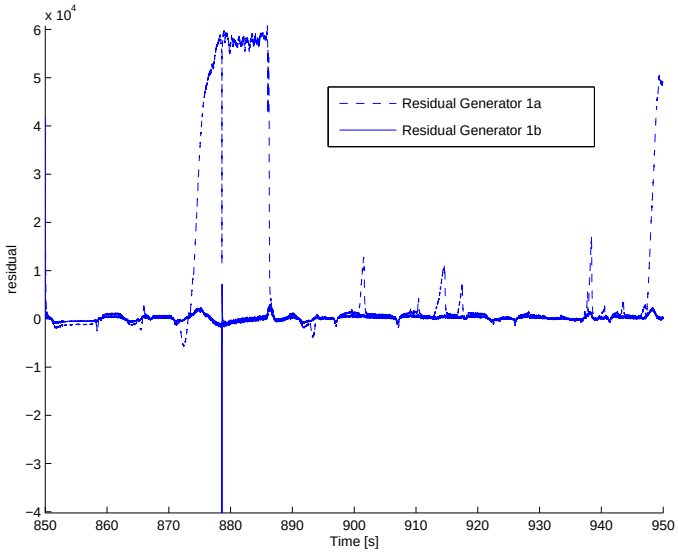



**Figure 10.10.** The residual $r = n_{\mathrm{trb}} - \hat{n}_{\mathrm{trb}}$ after the SIMULINK simulation of residual generator 1a (dashed line) and 1b (solid line). The residual computed by residual generator 1b (solid line) is closer to zero and more 'calm' than the residual from residual generator 1a.

### 10.1.4 Simulation with Gain Faults

To be able to better evaluate the performance of the stabilized residual generator, SIMULINK simulations with simulated gain faults in actuators and sensors were performed. For the case with residual generator 1, there are 10 actuator faults and one sensor fault to be studied. All actuator faults have been studied but only faults in actuator $w_{\text{cmp}}$ will be considered here. The reason for this, is that the input signal $w_{\text{cmp}}$ have the largest static gain of all input signals. This means that a static fault in $w_{\text{cmp}}$ will be more prominent than a static fault affecting any other input signal. For more information regarding static gains, see Chapter 9.

The faults were implemented as a 30% gain fault in the signals $w_{\text{cmp}}$ (actuator) and $n_{\text{trb}}$ (sensor). Only single faults were considered and both faults occured at t = 903 s and lasted for 5 s each. To be able to compare how the three residual generators 1, 1a and 1b handles faults, the residuals were normalized. The normalization was done by dividing the residual with the estimated standard deviation for the non-faulty residual, $\sigma_{NF}$, defined as

$$\sigma_{NF} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(r_i - \bar{r})}$$

where $r_i$ is sample number $i$ of the non-faulty residual and $\bar{r}$ the mean value of the same residual. Theoretically, this normalization would give the residuals computed by the three residual generators same variance. Due to peaks in the none-faulty residuals, see for example Figure 10.10 at t = 878 s, this is not really the case.

In Figure 10.11 the result after the simulation with gain faults in the actuator and sensor is shown. To show how the residual generators are affected by faults, the non-faulty residuals are also presented. Magnifications near the time when the faults occurs is shown in Figure 10.12.

#### Residual Generator 1

As said earlier, the residual from residual generator 1 does not stay close to zero in the fault free case due to instability. Despite the instability problems, residual generator 1 clearly reacts when a fault is present. By studying the Figures 10.11(a), 10.11(b), 10.12(a) and 10.12(b), one can see that the residual switches level when the faults occur. One common way to see if a fault is present is to compare the residual with a pre-computed threshold, see [14]. If the residual is very 'alive' and not that close to zero in the fault-free case, it can be hard to compute a constant threshold. This is the case for the residual generator 1 and the computed residual is not suited for diagnosis even if it clearly reacts on both sensor and actuator faults.

#### Residual Generator 1a and 1b

The residual from the stabilized residual generators, 1a and 1b, shown in Figures 10.11(c) and 10.11(d) and Figures 10.11(e) and 10.11(f) reacts a bit different on faults. As said in Chapter 9, the design approach proposed in Section 8.2.1 will

result in that the frequency functions $G_{rf_u}(j\omega)$ and $G_{rf_y}(j\omega)$ will have different characteristics. The response of residual generators 1a and 1b on a sensor fault, Figures 10.12(d) and 10.12(f), clearly shows that the corresponding frequency function from fault to residual has high-pass characteristics. How residual generator 1a and 1b reacts on a sensor fault can be a problem. The residual clearly peaks when the fault occurs but then decreases to a level close to zero, and this can make the fault hard to detect.

By studying the Figures 10.12(c) and 10.12(e) one can see that there is a difference in how residual generators 1a and 1b reacts on actuator gain fault. As said in Section 2.3, a larger $Q$ results in that the residual generator will be more sensitive for disturbances affecting the measured signal and this is exactly the case for residual generator 1b, as can be seen in Figures 10.12(e) and 10.11(e). On the other hand, the large $Q$ results in that the state, $\hat{n}_{\text{trb}}$, in residual generator 1b follows the measurement better than the same state in residual generator 1a, which can be seen in Figure 10.9. The resulting residual $r = \hat{n}_{\text{trb}} - n_{\text{trb}}$ will be closer to zero for residual generator 1b than for residual generator 1a, which can be seen in Figure 10.10.

In conclusion, residual generator 1b has a residual close to zero that is very 'noisy' and this can make a fault hard to detect. Residual generator 1a, is on the other hand, according to Figure 10.11(c) a bit more 'alive' but reacts more clearly on an actuator fault. The way residual generator 1a and 1b reacts on a sensor fault is a direct consequence of the feedback which gives the frequency functions from sensor fault to residual high-pass characteristics.

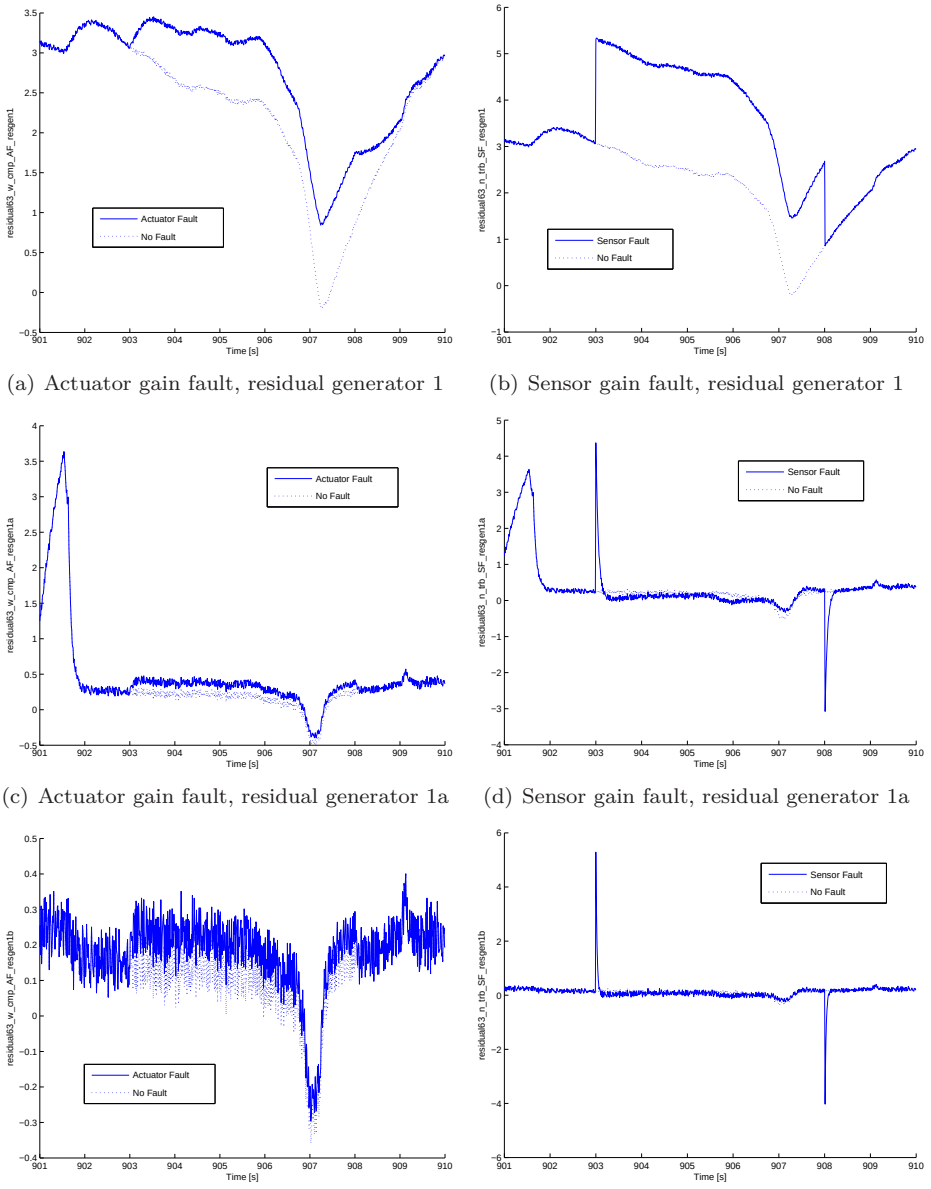
### 10.1.5 Simulation with Bias Faults

To study how the residual generators reacts on bias faults in actuators and sensors, simulations in SIMULINK were performed. Only bias faults in actuator $w_{\text{cmp}}$ and sensor $n_{\text{trb}}$ were considered, with the same motivation as for gain faults.

Only single faults were considered and the faults were implemented as sudden additive faults in the signals $w_{\text{cmp}}$ (actuator) and $n_{\text{trb}}$ (sensor). The size of the faults was chosen to be 30 % of the mean value of the signal affected by the fault and the faults both occured at t = 903 s and lasted 5 s each. Furthermore, the residuals were normalized as described in Section 10.1.4.

In Figure 10.13 the result of the simulation is shown. To show how the residual generators are affected by faults, the non-faulty residuals are also presented.

By studying Figures 10.13(b), 10.13(d) and 10.13(f) one can see that the response of a bias sensor fault is very prominent, especially for residual generator 1. The responses of a bias actuator fault, Figures 10.13(a), 10.13(c) and 10.13(e), can be compared with the corresponding plots in Figure 10.11. One can see that the behaviors of residual generators 1, 1a and 1b are almost the same for bias and gain faults.




(a) Actuator gain fault, residual generator 1

(b) Sensor gain fault, residual generator 1

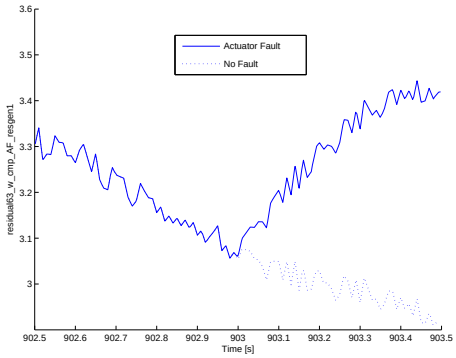
(c) Actuator gain fault, residual generator 1a

(d) Sensor gain fault, residual generator 1a

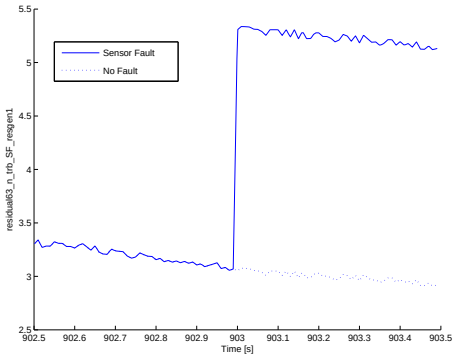
(e) Actuator gain fault, residual generator 1b

(f) Sensor gain fault, residual generator 1b

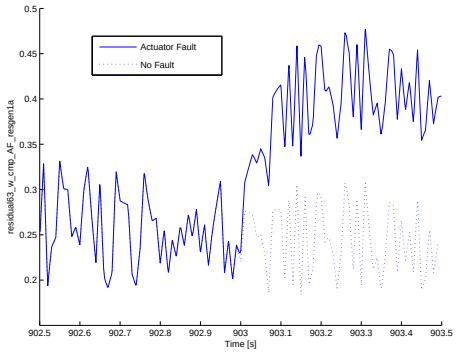
**Figure 10.11.** Simulation of residual generator 1, 1a and 1b in SIMULINK, with actuator and sensor gain faults affecting the measured signals $w_{\mathrm{cmp}}$ (actuator) and $n_{\mathrm{trb}}$ (sensor). The solid lines show the residuals with faults and the dotted lines the residuals in the fault-free case.



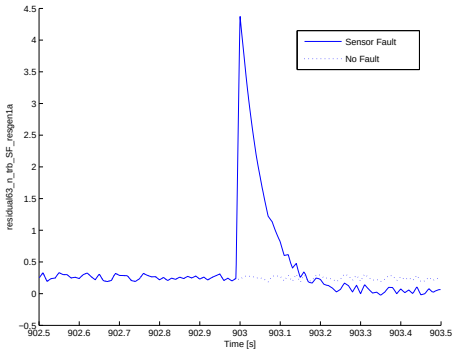

(a) Actuator gain fault, residual generator 1



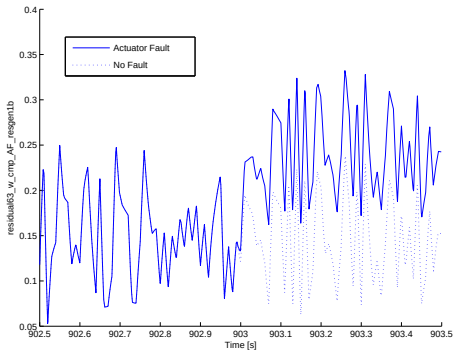

(b) Sensor gain fault, residual generator 1



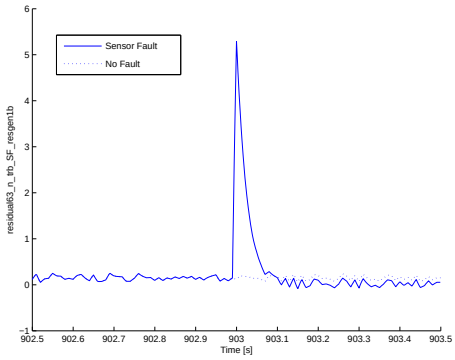

(c) Actuator gain fault, residual generator 1a




(d) Sensor gain fault, residual generator 1a

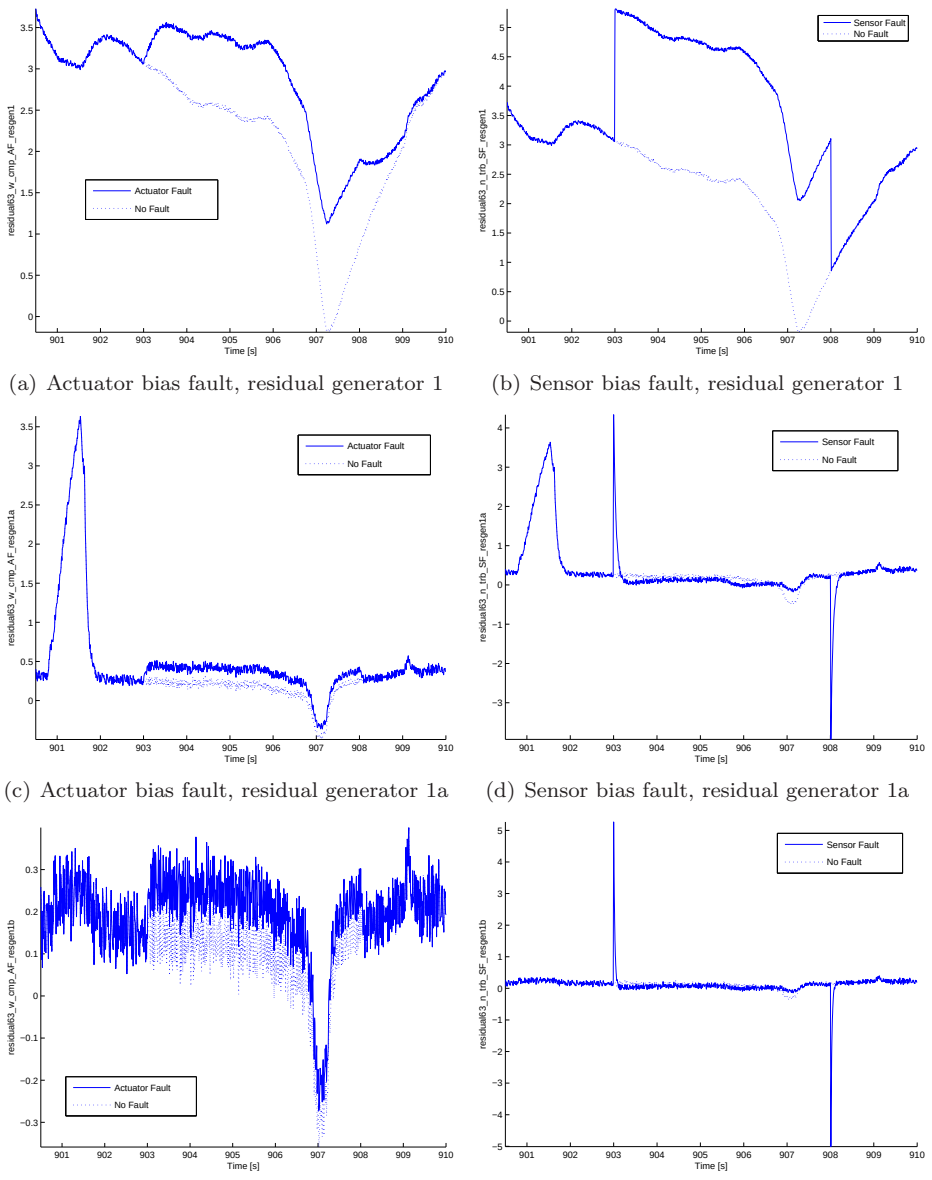



(e) Actuator gain fault, residual generator 1b




(f) Sensor gain fault, residual generator 1b

**Figure 10.12.** A magnification of Figure 10.11 near the time when the faults occur.




(a) Actuator bias fault, residual generator 1

(b) Sensor bias fault, residual generator 1

(c) Actuator bias fault, residual generator 1a

(d) Sensor bias fault, residual generator 1a

(e) Actuator bias fault, residual generator 1b

(f) Sensor bias fault, residual generator 1b

**Figure 10.13.** Simulation of residual generator 1, 1a and 1b in SIMULINK, with actuator and sensor bias faults affecting the measured signals $w_{\mathrm{cmp}}$ (actuator) and $n_{\mathrm{trb}}$ (sensor). The solid lines show the residuals with faults and the dotted lines the residuals in the fault-free case. The behaviors of the residual generators are almost the same as for gain faults, see Figure 10.11.

## 10.2 Residual Generator 2

Another residual generator can not be simulated at all due to instability problems. The residual generator have four states, uses eleven known signals and can be written as

$$\dot{x} = f(x,u) \tag{10.3a}$$
$$r = y - Cx \tag{10.3b}$$

where

$$x = \begin{bmatrix} \hat{p}_{\text{em}} \\ \hat{p}_{\text{es}} \\ \hat{p}_{\text{cmp}} \\ \hat{n}_{\text{trb}} \end{bmatrix}, \qquad u = \begin{bmatrix} p_{\text{im}} \\ w_{\text{cmp}} \\ \alpha \\ \delta \\ n_{\text{avg}} \\ p_{\text{amb}} \\ t_{\text{amb}} \\ t_{\text{im}} \\ u_{\text{egr}} \\ u_{\text{vgt}} \end{bmatrix}, \qquad y = n_{\text{trb}}, \qquad C = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}.$$

Since the implementation, design of stable residual generators and simulation were performed in the same way as for residual generator 1, see Section 10.1, the evaluation of residual generator 2 will be briefly described.

### 10.2.1 Design of Stable Residual Generators

By using the method described in Section 8.2, two stable residual generators based on (10.3) were designed in the same way described in Section 10.1.2. Totally 19 equilibrium points were found, and 9 of them were unstable. Two different Kalman gains were computed for every operating point, one set with the covariance matrices

$$Q = \begin{bmatrix} 10^{-3} & 0 & 0 & 0 \\ 0 & 10^{-3} & 0 & 0 \\ 0 & 0 & 10^{-3} & 0 \\ 0 & 0 & 0 & 10^{-3} \end{bmatrix} \text{ and } R = 1$$

and the resulting residual generator was called *residual generator 2a*. The other set were computed with the covariance matrices

$$Q = \begin{bmatrix} 10^{3} & 0 & 0 & 0 \\ 0 & 10^{3} & 0 & 0 \\ 0 & 0 & 10^{3} & 0 \\ 0 & 0 & 0 & 10^{3} \end{bmatrix} \text{ and } R = 1$$

and the residual generator was called *residual generator 2b*. The covariance matrix $N$ and the matrix $G$, see Section 2.3.2, used in the calculation were set to

$$N = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

For the same reason as for residual generator 1, the input signal $\delta$ was chosen as signal for gain switching. Finally, the stabilized residual generators 2a and 2b were implemented in SIMULINK.

### 10.2.2 Fault-Free Simulation

To evaluate the performance of the three residual generators 2, 2a and 2b in the fault-free case, they were simulated in SIMULINK. As input data for the simulation, non-faulty measurement data were used.

#### Residual Generator 2

Due to the instability of residual generator 2, the simulation could not be run for more than 0.7 s. The states $\hat{p}_{\text{em}}$, $\hat{p}_{\text{es}}$, $\hat{p}_{\text{cmp}}$ and $\hat{n}_{\text{trb}}$ are more or less affected by the instability and are drifting away, as can be seen in Figure 10.14. There is a significant difference between the fourth state $\hat{n}_{\text{trb}}$ (solid line) and the measured signal $n_{\text{trb}}$ (dotted line). This causes the residual, $r = n_{\text{trb}} - \hat{n}_{\text{trb}}$ to be far away from zero, as can be seen in Figure 10.15.

#### Residual Generator 2a and 2b

By using the design method described in Section 8.2.1 to design residual generators 2a and 2b, the clearly unstable residual generator 2 can be simulated without problems. The four states of residual generator 2a and 2b is shown in Figure 10.16 and the computed residuals in Figure 10.17. As for residual generator 1, residual generator 2b with the Kalman gains calculated with the covariance matrices related as $Q > R$, see Section 10.2.1, produces a residual with better properties. This can be seen by studying Figure 10.17.

### 10.2.3 Simulation with Faults

Due to the instability of residual generator 2, simulation with faults could not be performed because the simulation software crashed. What concerns residual generator 2a and 2b, the results for both gain and bias fault are similar as for residual generator 1a and 1b and therefore, no plots is presented.

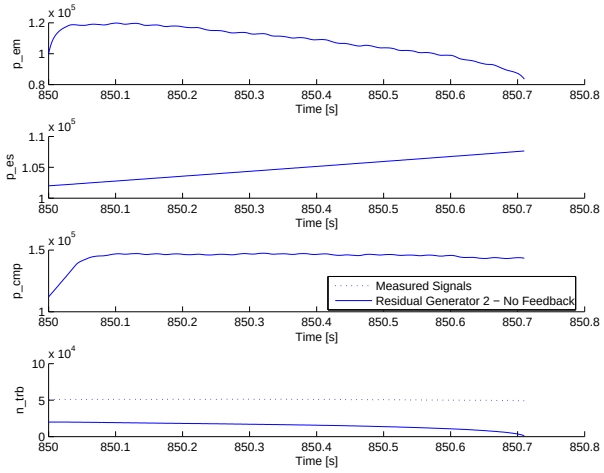



**Figure 10.14.** The states $\hat{p}_{\mathrm{em}}$, $\hat{p}_{\mathrm{es}}$, $\hat{p}_{\mathrm{cmp}}$ and $\hat{n}_{\mathrm{trb}}$ after the SIMULINK simulation of residual generator 2 using fault-free input data. Due to the instability, residual generator 2 can not be simulated for more than 0.7 s.

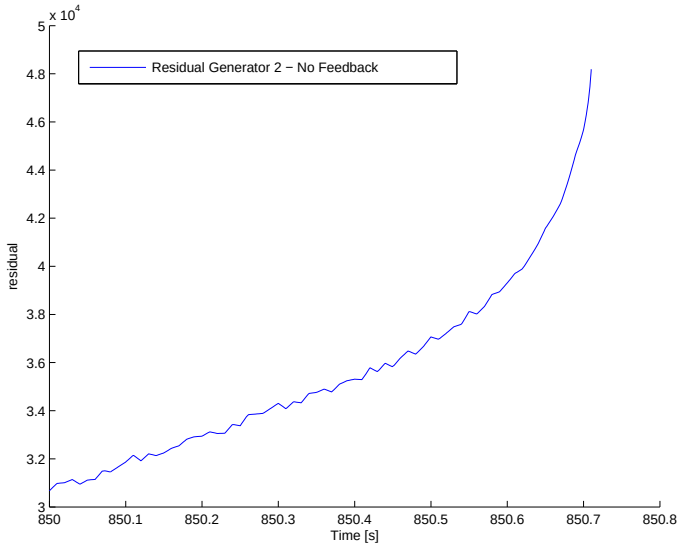



**Figure 10.15.** The residual $r = n_{\mathrm{trb}} - \hat{n}_{\mathrm{trb}}$ calculated residual generator 2. Due to the instability, residual generator 2 could not be simulated for more than 0.7 s and the residual is far away from zero.

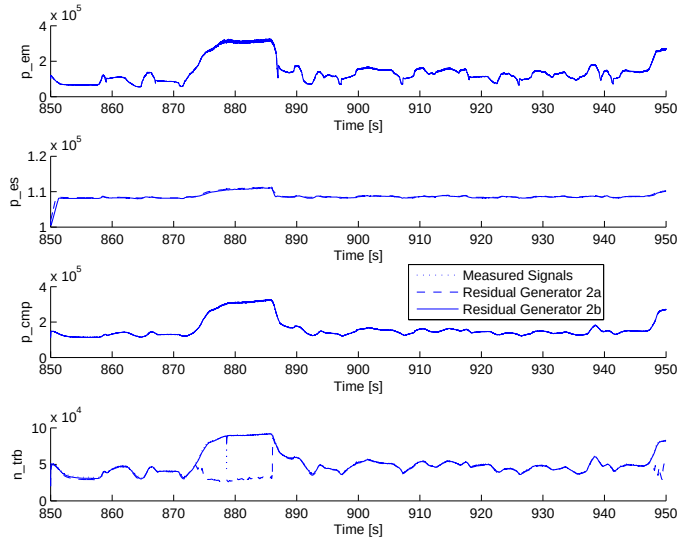



**Figure 10.16.** The four states $\hat{p}_{\mathrm{em}}$, $\hat{p}_{\mathrm{es}}$, $\hat{p}_{\mathrm{cmp}}$ and $\hat{n}_{\mathrm{trb}}$ after the SIMULINK simulation of residual generators 2a (dashed line) and 2b (solid line) using fault-free input data. The fourth state computed by residual generator 2b better follows the measured signal than the state from residual generator 2a.

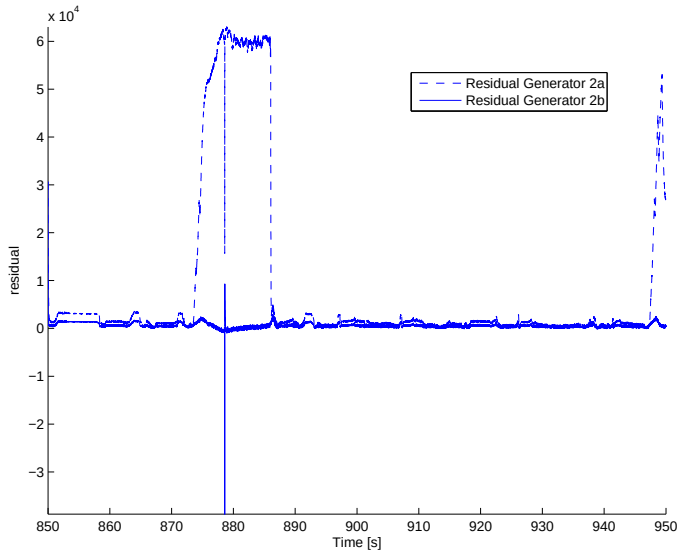



**Figure 10.17.** The residual $r = n_{\mathrm{trb}} - \hat{n}_{\mathrm{trb}}$ from residual generator 2a (dashed line) and 2b (solid line). The residual computed by residual generator 2b is closer to zero and more 'calm' than the residual from residual generator 2a.

# Chapter 11

# Conclusions

## Methods for Residual Generation

- Methods using extended differential equations as residual equations (DRE) are equivalent with methods that does not.
- Methods using integral and derivative causality are equivalent for all static systems.
- Methods using integral and derivative causality are not equivalent for all dynamic systems. Hence, a method using both causality assumptions would likely produce a diagnosis system with better detectability properties than a method assuming one of the causalities.

## The Scania Method

- Some of the systems, assumed to be residual generators, found with the Scania Method are not residual generators according to Definition 3.2, due to instability. Instability of a residual generator is often caused by weak robustness of the system.
- From an unstable residual generator, a stable residual generator can be constructed with the method described in Section 8.2.
- The method for constructing a stable residual generator affects the sensitivity of the residual generator and the detectability properties of the diagnosis system. The properties of the stabilized residual generator highly depends on the parameters used in the design process.

# Bibliography


[1] Armen S. Asratian, Tristan M. J. Denley, and Roland Häggkvist. *Bipartite Graphs and their Applications*. Cambridge University Press, 1998. ISBN 0-521-59345-X.

[2] Mogens Blanke, Michel Kinnaert, Jan Lunze, and Marcel Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer-Verlag, 1 edition, 2003. ISBN 3-540-01056-4.

[3] Henrik Einarsson and Gustav Arrhenius. Automatic design of diagnosis systems using consistency based residuals - optimizing isolation and computational load. Master's thesis, Uppsala University, 2005. UPTEC F 1401-5757 ; 05032.

[4] Lars Eriksson. Structural algorithms for diagnostic system design using simulink models. Master's thesis, Dept. of Electrical Engineering, Linköping University, 2005. LiTH-ISY-EX-3601-2004.

[5] Erik Frisk and Jan Åslund. Lowering orders of derivatives in non-linear residual generation using realization theory. *Automatica*, 41:1799–1807, 2005.

[6] Torkel Glad and Lennart Ljung. *Reglerteknik – Grundläggande teori*. Studentlitteratur, 1981. ISBN 91-44-17892-1.

[7] Torkel Glad and Lennart Ljung. *Control Theory – Multivariable and Nonlinear Methods*. CRC, 1 edition, 2000. ISBN 0-748-40877-0.

[8] T. Kailath, A.H. Sayed, and B. Hassibi. *Linear Estimation*. Prentice Hall, 2000. ISBN 0-13-022464-2.

[9] Thomas Kailath. *Linear Systems*. Prentice Hall, 1980. ISBN 0-13-536961-4.

[10] Hassan K. Khalil. *Nonlinear Systems*. Macmillan Publishing Company, 1992. ISBN 0-02-363541-X.

[11] Kristian Krigsman and John Nilsson. Code generation for efficient real-time execution of diagnostic residual generators. Master's thesis, Dept. of Microelectronics and Information Technology, KTH, 2005. IMIT/LECS-2004-66.

[12] Mattias Krysander. *Design and Analysis of Diagnosis Systems Using Structural Methods*. PhD thesis, Linköpings universitet, June 2006.

[13] 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*, Accepted for publication.

[14] Mattias Nyberg and Erik Frisk. *Model Based Diagnosis of Technical Processes*. Linköping University, 2006.

[15] Belarmino Pulido and Carlos Alonso González. Possible conflicts: A compilation technique for consistency-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 34(5):2192–2206, 2004.

[16] Wilson J. Rugh. *Linear System Theory*. Prentice Hall, 2 edition, 1996. ISBN 0-13-441205-2.

[17] Michael G. Safanov and Michael Athans. Robustness and computational aspects of nonlinear stochastic estimators and regulators. *IEEE Transactions on Automatic Control*, 23(4):717–725, 1978.

[18] M. Staroswiecki and G. Comtet-Varga. Analytical redundancy relations for fault detection and isolation in algebraic dynamic systems. *Automatica*, 37:687–699, 2001.

[19] L. Travé-Massuyès and P. Dague. Common diagnostic framework specification, monet2: Network of excellence on model based systems and qualitative reasoning. http://www.cs.ucc.ie/∼gprovan/CS5205/Readings/mbd_survey03.pdf, 30/1/2003.