Implementation of model-based diagnosis methods on an inverted pendelum

Examensarbete utfört i Fordonssystem
vid Tekniska Högskolan i Linköping
av

Per Nygren

Reg nr: LiTH-ISY-EX-1759

Implementation of model-based diagnosis methods on an inverted pendelum

Examensarbete utfört i Fordonssystem
vid Tekniska Högskolan i Linköping
av

Per Nygren

Reg nr: LiTH-ISY-EX-1759

Supervisor: **Erik Frisk**
Lars Nielsen

Examiner: **Lars Nielsen**

Linköping, November 13, 1996.

Abstract

A diagnosis procedure is an algorithm to detect and locate (isolate) faulty components in a dynamic process.

To achieve diagnosis, redundancy has to be included in the system. This redundancy can be either hardware redundancy or analytical redundancy. Methods based on analytical redundancy need no extra hardware. The redundancy is generated from a process model instead. In this thesis, methods based on analytical redundancy are used.

A mathematical model and a controller are derived for an inverted pendelum. Two diagnosis methods, parity equations and parameter estimation, are implemented on the pendelum. Different faults are then applied to the pendelum. Weights on the rod and on the cart of the pendelum corresponds to component faults, constants added to the output and input signals corresponds to sensor faults and actuator faults respectively. The diagnosis methods are studied in order to see how good the methods are compared with each other and to results in scientific articles using similar methods.

Key Word: Diagnosis, Inverted pendelum, Analytical redundancy, Parity equations, Parameter estimation, Residual generation.

Acknowledgments

I wish to thank my supervisors Erik Frisk and Lars Nielsen and all the people at Fordonssystem for the help and support they have been giving me during this time. Thanks to Erik for the Figures 2.1, 2.2, 2.3 and 2.4.

Notation

Abbreviations

AFD	Actuator Fault Diagnosis.
ARMA	AutoRegressive Moving Average.
ARMAX	AutoRegressive Moving Average with eXternal input.
ARX	AutoRegressive with eXternal input.
BJ	Box-Jenkins.
CFD	Component Fault Detection.
DSP	Digital Signal Processor.
FDI	Fault Detection and Isolation.
FIR	Finite Impulse Respons.
GLR	Generalized Likelihood Ratio.
IFD	Instrumental Fault Diagnosis.
LQ	Linear Quadratic.
LS	Least Squares.
RLS	Recursive Least Squares.
SPRT	Sequential Probability Ratio Test.
SVD	Singular Value Decomposition.
TMR	Triple Modular Redundancy.

Contents

Chapter 1

Introduction

Diagnosis is a procedure to detect and locate faulty components in a dynamic process. Faults and failures in complex automated control systems are, in general, unavoidable facts and they require quick detection, location and identification. A diagnosis scheme is of importance in, for example

- Nuclear plants
- Aeroplanes
- Automotive engines

This is due to increasing demand for higher performance, higher safety and reliability. Different fault detection and isolation techniques have been developed over the recent years.

This report is a study of how two diagnostic methods are to be implemented. The process that is used to study the methods are the inverted pendelum. The fault detection and isolation (FDI) concept consist of two major steps: the residual generation and the residual evaluation. The main focus here will be the residual generation.

1.1 Motives for the thesis

At Fordonssystem diagnosis is a rather new research area. There are at the moment two Ph.D students in diagnosis at Fordonssystem. They are interested in how different methods in diagnosis perform on a process, if the different methods are difficult to implement, what kind of problems different methods can alert and other uncertainties. This has resulted in this thesis where some methods in model based diagnosis are to be implemented on an inverted pendelum.

1.2 Objectives

The objectives with this thesis work is to

- Design and simulate the diagnosis methods in a simulation tool.

- Implement the chosen methods in the lab environment.
- Design experiments to evaluate the methods with respect to robustness, diagnostic time, determine the type of faults that the methods are well suited to diagnose.
- Compare the methods with each other and with experiments in articles.

1.3 Readers guide

We start in Chapter 2 by defining the diagnosis problem and describe some approaches that are found in literature. Those who are familiar with diagnosis can skip this chapter.

Then in Chapter 3, we give a description over the system that is used when implementing the chosen methods, the inverted pendelum. A physical model and a controller for the system are derived. The controller, which is built in Simulink, works on the pendelum with the help of dSPACE. A presentation of dSPACE is therefore given in Chapter 4.

In Chapter 5, the first method, parity equations, applied to the pendelum is presented. First, the theory of the method is given. Then the results after simulation and the results after using the method on the pendelum are reported. The chapter ends with the problems the method gave in this application and what it can give in other applications.

Chapter 6 present the second method, parameter estimation. The chapter is organized as the previous chapter.

Chapter 2

The Fault diagnosis problem

In this chapter the diagnosis problem will be defined and approaches to solve the problem will be presented. This chapter is based on [1].

2.1 Problem formulation

A general diagnosis procedure for a dynamic system consists of several tasks. In literature the following steps are suggested.

- **Fault detection:** Detect when a fault has occured. That is often done with a suitable comparison, for example in parameter estimation, the estimated physical parameters are compared to their nominal values.
- **Fault isolation:** Isolate the fault. Primarely to determine the faults origin but also the fault's type, size and time.

These two tasks are commonly referred to as FDI which sometimes is referred to as diagnosis and the other way around.

The system to be diagnosed often include a control loop which further complicates the problem. A control loop tends to hide or mask a faulty component or sensor making it even more important, in a controlled system, to detect faults. The control loop can also damp the system's signals making it necessary to excite the signals from the system. If the system, as in this thesis, is unstable, we have a control loop and a special test sequences which is used for fault detection thus making methods badly suited for sudden faults.

We speak of *faults* and *failures* in diagnosis. In diagnosis literature there is a distinction between the two and the definition can be written as:

Definition 2.1. *A failure suggests a complete breakdown of a process component while a fault is thought of as an unexpected component change that might be serious or tolerable.*

Obvious fault sources are actuators and sensors where the faults can be a bias or a gain fault. Other examples are component faults, e.g a mass that changes.

Fault diagnosis and fault detection is not a new problem and before model based fault diagnosis, they were accomplished e.g by introducing *hardware redundancy* in the process. A critical component was then duplicated, triplicated (TMR) or even quadrupled and a majority decision rule was then used. Hardware redundancy methods are fast and easy to implement but they have several drawbacks

- Extra hardware can be very expensive
- It introduces more complexity in the system
- The extra hardware is space consuming which can be of great importance, e.g in a space shuttle. Also the components weight sometimes has to be considered.

Instead of using hardware redundancy, *analytical redundancy* can be utilized to reduce, or even avoid, the need for hardware redundancy. The methods examined in this report are founded on analytical redundancy. Analytical redundancy is in principle the relationships that exists between process variables and measured output signals. If an output signal is measured there are information about all variables that influences the output signal in the measurement. If the relationships are known , by quantative or qualitative knowledge, this information can be extracted and the extracted information from different measurements can be checked for consistency against each other.

There are different types of analytical redundancy. Instead of measuring several outputs we compare different output measurements at different times. If the relationship between time series of outputs and inputs are known, we can from this relationship extract fault information. This kind of analytical redundancy is called *temporal redundancy.*

The faults acting upon a system can be divided into three types of faults.

- **Sensor (Instrument) faults:** Faults acting on the sensors
- **Actuator faults:** Faults acting on the actuators
- **Component (System) faults:** A fault acting on the system or the process we wish to diagnose.

A general FDI scheme based on analytical redundancy can be illustrated as in Figure 2.1, an algorithm with measurements and control signals as inputs and a fault decision as output.

It is unrealistic to assume that all signals acting on the process can be measured, therefore an important property of an algorithm is how it reacts on these unknown inputs. It is also unrealistic to assume a perfect model, the modelling errors can be seen as unknown inputs. An algorithm that continue to work satisfactory even when unknown inputs vary is called *robust.* In some approaches we have the possibility to achieve disturbance decoupling, i.e make the isolation decision independant of unmeasured disturbances.

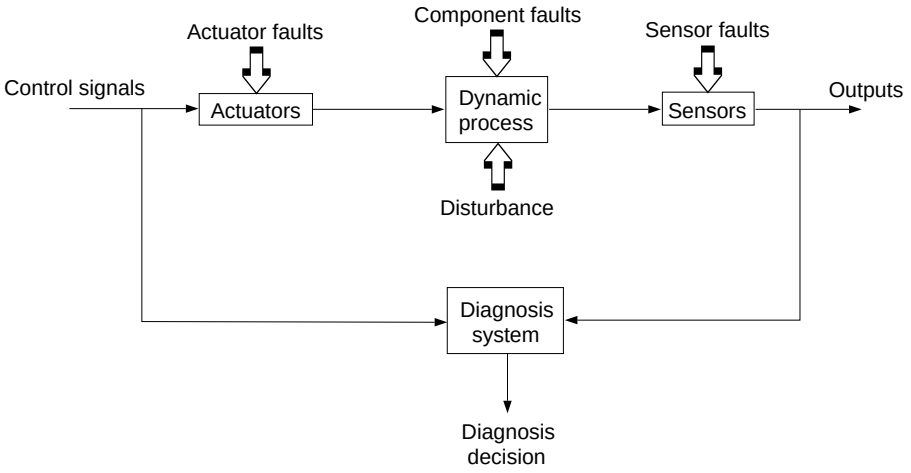

Figure 2.1. *Structure of a diagnosis system.*

2.2 Control engineering approaches to FDI

In control based approaches the diagnosis procedure is explicitly parted into two stages, the residual *generation* stage and the residual *evaluation* stage, as illustrated in Figure 2.2. The residual evaluation can in its simplest form be a thresholding test on the

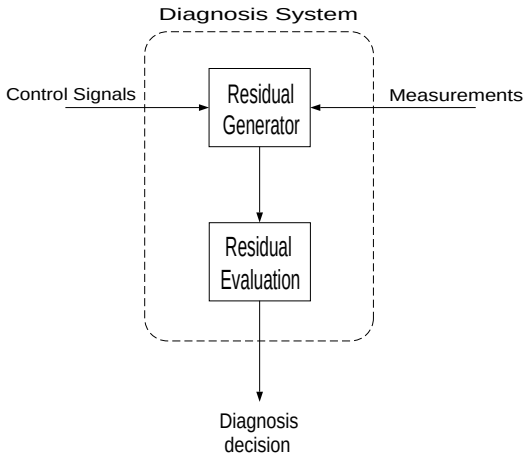

Figure 2.2. *Two stage diagnosis system.*

residual, i.e. a test if $\mid r(t) \mid > Threshold$. More generally the residual evaluation stage consists of a *change detection* test and a *logic inference system* to decide what caused the change. A change here represents a change in normal behavior of the residual.

The residual generation approaches can be divided into three subgroups, *limit & trend checking, signal analysis* and *process model based.*

- **Limit & trend checking:** This approach is the simplest imaginable, testing sensor outputs against predefined limits and/or trends. This approach needs no mathematical model and is therefore simple to use, but it is hard to achieve high performance diagnosis.

- **Signal analysis:** These approaches analyse signals, i.e sensor outputs, to achieve diagnosis. The analysis can be made in the frequency domain, or by using a signal model, e.g an ARMA-model. If fault influence are known to be greater than the input influence in well known frequency bands, a time-frequency distribution method can be used.

- **Process model based residual generation:** These methods are based on a *process* model. The process model based approaches are further parted into two groups, *parameter estimation*, and *geometric approaches*.

Before we can discuss the methods in this section we need to make some definitions. The approaches to be discussed here generates residuals which can be defined as

Definition 2.2 ***[Residual].*** *A residual r(t) is a scalar or vector that is 0 or small in the fault free case and $\neq 0$ when a fault occurs.*

The residual is a vector in the *parity space*. This definition implies that a residual $r(t)$ has to be *independant* of, or at least *insensitive* to, system states and unmeasured disturbances.

We will now concentrate on linear systems because they can be systematically analyzed. A general structure of a linear residual generator, can be discribed as in Figure 2.3. The transfer function from the fault $f(t)$ to the residual $r(t)$ then becomes

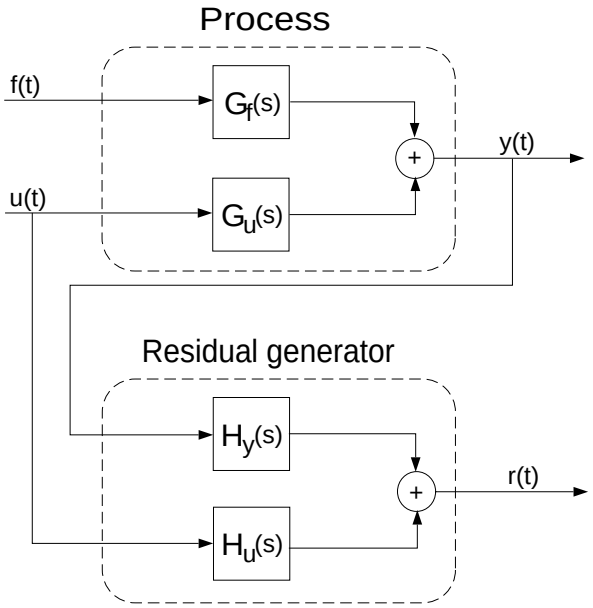

Figure 2.3. *General structure of a linear residual generator.*

$$r(s) = H_y(s)G_f(s)f(s) = G_{rf}(s)f(s)$$

The conditions that has to be fulfilled to be able to detect a fault in the residual has a natural definition. To be able to detect the i:th fault in the i:th column of the response matrix, $[G_{rf}(s)]_i$ has to be nonzero, i.e.

Definition 2.3 ***[Detectability].*** *The i:th fault is detectable in the residual if*

$$[G_{rf}(s)]_i \neq 0$$

This condition is however not enough in some practical situations. Assume that we have two residual generators with structure as in Figure 2.3. When excited to a fault the residuals behave as in Figure 2.4. Here we see that we have a fundamentally different

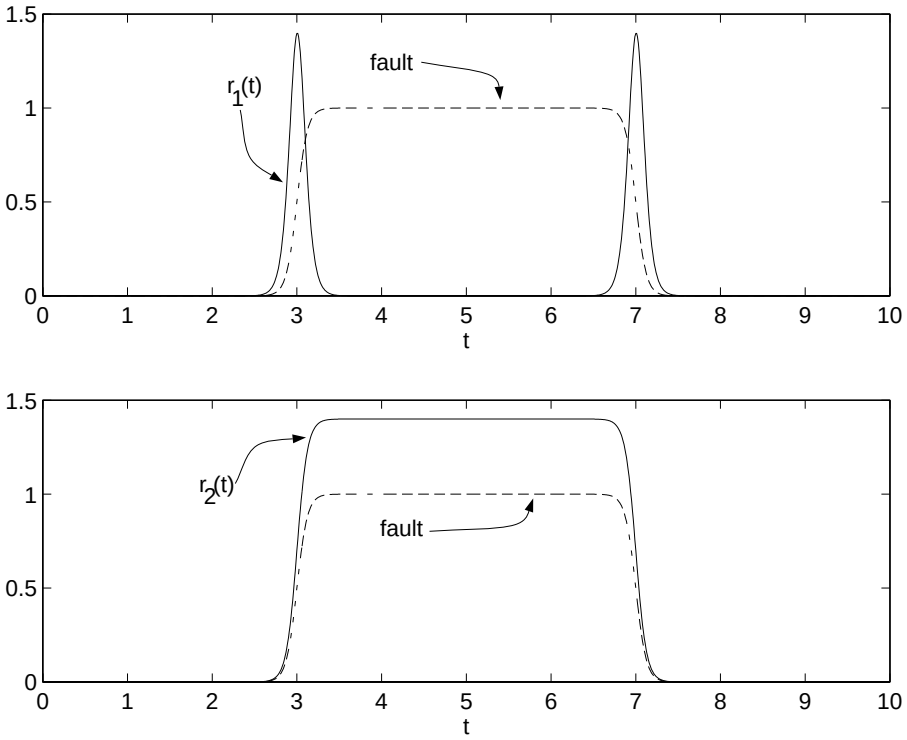

Figure 2.4. *Example residuals.*

behavior between $r_1(t)$ and $r_2(t)$ as $r_1(t)$ only reflects changes on the fault signal and $r_2(t)$ has approximately the same shape as the fault signal. Thus $r_1(t)$ can not be used in a reliable FDI application even though it is clear that $G_{r_1 f}(s) \neq 0$.

The difference between the two residuals in the example are the value of $G_{rf}(0)$. It is clear that residual 1 has $G_{r_1 f}(0) = 0$ while residual 2 has $G_{r_2 f}(0) \neq 0$, This leads to another definition

Definition 2.4 ***[Strong detectability].*** *The i:th fault is said to be strongly detectable if and only if*

$$[G_{rf}(0)]_i \neq 0$$

The above definitions show that it can be of great importance to perform a frequency analysis of the residual generator. If the designed residual generator has a response like $r_1(t)$, an easy solution can be to filter the residual, e.g through a integrating filter.

2.2.1 Model based diagnosis

In this report we will investigate model based diagnosis, i.e a diagnosis procedure that is founded on a model of a system to be diagnosed. Why is there a need for a mathematical model to achieve diagnosis? It is easy to imagine a scheme where important entities of the dynamic process is measured and tested against predefined limits. The model based approach instead performs consistency checks of the process against a model of the process. There are several important advantages with the model based approach.

1. Outputs are compared to their expected value on the basis of process state, therefore the thresholds can be set much tighter and the probability to identify faults in an early stage is increased dramatically.

2. A single fault in the process often propagate to several outputs and therefore causes more than one limit check to fire. This makes it hard to isolate faults without a mathematical model.

3. With a mathematical model of the process the FDI scheme can be made insensitive to unmeasured disturbances, making the FDI scheme feasible in a much wider operating range.

2.2.2 Isolation strategies

If we now have strongly detectable residuals, how can isolation be achieved? Here the method with structured residuals is described.

The idea behind structured residuals is that a bank of residuals is designed making each residual insensitive to different faults or subsets of faults while remaining sensitive to the remaining faults, i.e. if we want to isolate three faults we can design three residuals $r_1(t)$, $r_2(t)$ and $r_3(t)$ to be *insensitive* to one fault each. Then if residuals $r_1(t)$ and $r_3(t)$ fire we can assume that fault 2 has occured.

2.2.3 Geometric approaches

Geometric approaches generate residuals which are vectors. The methods can be divided into open and closed loop approaches. In an open loop approach there are, as the name suggests, no feedback from previously calculated residuals. The idea behind closed loop approaches, i.e. observer based approaches, are to use a state estimator as a residual generator.

2.2.4 Parameter estimation approach

In a parameter estimation approach is important parameters in a process estimated, for example masses, volumes or frictional coefficients, compared to a nominal value. The method can be described in three steps.

1. The data processing step where parameters are estimated.

2. The fault detection step where the comparison to a nominal value is done.

3. The fault classification step where it is tested if a fault is present or not. If so, isolation of the fault source is done.

2.3 Robustness issues

One problem, as was noted earlier, is that unmeasurable signals often act upon the system plus the influence by modelling errors. This makes it hard to keep the false alarm rate at an appropriate level.

If it is known how the uncertainty influences the process, so called *structured* uncertainty, this information can be utilized to actively reduce or even eliminate their influence on the residuals. If it is not known how disturbances act on the system there is little that can be done to achieve any decoupling. We actually don't produce any robustness, at best we can maximize the sensitivity to faults and minimize the sensitivity to disturbances over all operating points.

However it is possible to increase robustness in the fault evaluation stage, i.e. in the threshold selection step, e.g by using adaptive threshold levels or statistical decoupling. This is called passive robustness. It is not likely that one method can solve the entire robustness problem, a likely solution is one where disturbance decoupling is used side by side with passive robustness.

2.4 Model structure

To proceed in the analysis of residual generation approaches we need an analytical model. In this report a state representation of the model are used as

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= h(x(t), u(t)) \end{aligned} \tag{2.1}$$

The linear (time-continuous) state representation

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{2.2}$$

As we have noted earlier we have three general types of faults:

- **Sensor (Instrument) faults:** Modeled as an additive fault to the output signal.
- **Actuator faults:** Modeled as as additive fault to the input signal in the system dynamics.
- **Component (System) faults:** Modeled as entering the system dynamics with any distribution matrix. Here it is seen that actuator faults only are special case of component faults.

There are also uncertainties about the model or unmeasured inputs to the process. If these uncertainties are structured, i.e. it is known how they enter the system dynamics, this information can be incorporated into the model.

In the linear case and model uncertainties are supposed structured, the complete model becomes

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B(u(t) + f_a(t)) + Hf_c(t) + Ed(t) \\ y(t) &= Cx(t) + Du(t) + f_s(t) \end{aligned} \tag{2.3}$$

where $f_a(t)$ denotes actuator faults, $f_c(t)$ component faults, $f_s(t)$ sensor faults and $d(t)$ disturbances acting on the system. H and E is called the distribution matrices for $f_c(t)$ and $d(t)$.

2.5 Approaches in this report

In this report the residual generation stage is emphasized. The two methods, parity equation and parameter estimation, both uses process model based residual generation.

Chapter 3

Modelling the inverted pendelum

In this chapter a model for the system is derived which later is used by both the parity equations method described in Chapter 5 and the parameter estimation method described in Chapter 6. The controller for the unstable system was already made, but designing the controller includes using the model of the system and therefore is the design of the controller using Linear Quadratic control (LQ) shortly described.

When experimenting with the pendelum, dSPACE is used to get the controller, which is implemented in SIMULINK, and the pendelum to work together. The products from dSPACE makes it possible to use constructions, for example controllers, in SIMULINK on real processes. They also makes it possible to study interesting signals, saving them to MATLAB and change parameters during an experiment. In the next chapter the products from dSPACE are further explained.

3.1 System description

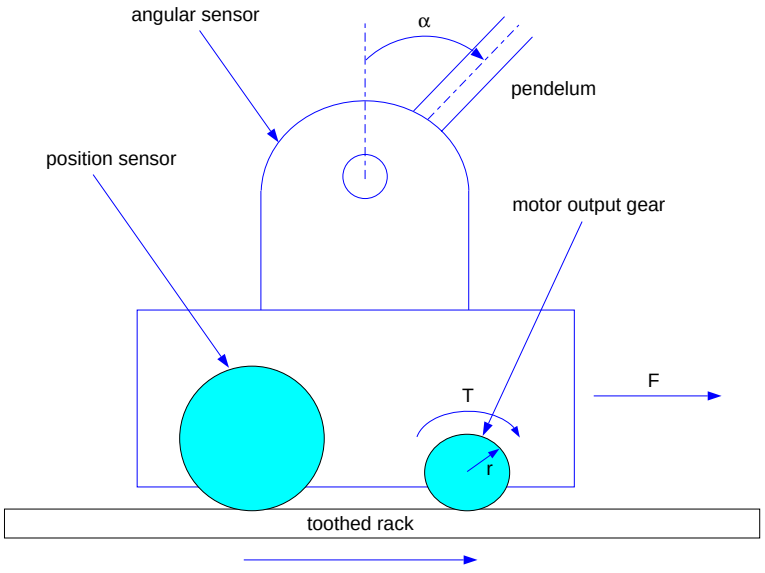

Figure 3.1. *Experimental setup for the inverted pendelum.*

The inverted pendelum, shown in Figure 3.1, consists of a cart which is equipped with a motor and a potentiometer. The motor gives the driving force to the system and the potentiometer is used to measure the cart's position. The motor shaft is connected to one gear while the potentiometer shaft is connected to another larger gear. Both these gears mesh with the toothed rack where the motor makes the cart to move sideways. When the cart moves, the potentiometer shaft turns and the voltage measured from the potentiometer can be calibrated to obtain the track position. A rod is mounted on the cart whose axis of rotation is perpendicular to the direction of motion of the cart. A potentiometer mounted on the axis of rotation allows you to measure the angle of the rod with the vertical.

3.2 The model

The equations of motion for the inverted pendelum are treated in several books, for example [5]. The system is both unstable and non-linear but a linear model can be obtained around an operating point. Here, the mass of the rod is assumed to be concentrated at a point mass m_p at distance l from the axis of rotation. Furthermore it is assumed that the rod is rigid and that friction can be neglected. The simplified model can be seen in Figure 3.2. If these assumptions hold, then the force F_p exerted by the cart on the

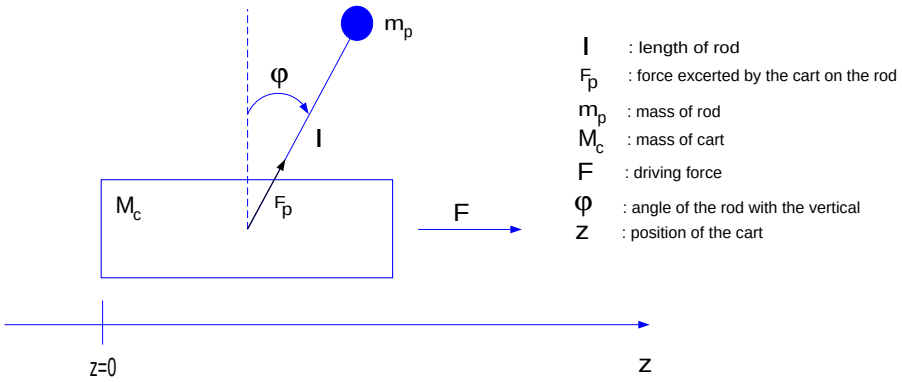

Figure 3.2. *Simplified model of the inverted pendelum.*

pendelum at the pivot must always act along the rod. To derive a mathematical model of the remaining factors of the system, Newton's second law is used for the forces on the pendelum and the cart in the horizontal and vertical direction [2].

Applying Newton's law for the forces on the cart in the horizontal direction gives

$$F - F_p \sin\varphi = M_c \ddot{z} \tag{3.1}$$

The forces on the pendelum in the horizontal direction gives

$$F_p \sin\varphi = m_p \frac{d^2}{dt^2}\left(z + l\sin\varphi\right) = m_p\left(\ddot{z} + l\ddot{\varphi}\cos\varphi - l\dot{\varphi}^2\sin\varphi\right) \tag{3.2}$$

Applying on the forces on the pendelum in the vertical direction gives

$$m_p g - F_p \cos\varphi = m_p \frac{d^2}{dt^2}\left(l - l\cos\varphi\right) = m_p l\left(\ddot{\varphi}\sin\varphi + \dot{\varphi}^2\cos\varphi\right) \tag{3.3}$$

Eliminating F_p from (3.1), (3.2) and (3.3) results in

$$\begin{aligned} F &= (M_c + m_p)\,\ddot{z} + m_p l\left(\ddot{\varphi}\cos\varphi - \dot{\varphi}^2\sin\varphi\right) \\ m_p g\sin\varphi &= m_p\ddot{z}\cos\varphi + m_p l\ddot{\varphi} \end{aligned} \tag{3.4}$$

The linearizing of (3.4) is done around $z = \dot{z} = \varphi = \dot{\varphi} = 0$. This means that higher order terms such as φ^2, $\dot{\varphi}^2$ and $\varphi\dot{\varphi}$ are neglected since they are assumed to be small when φ is close to zero. The $\cos\varphi$ and $\sin\varphi$ terms are linearized using the Taylor series of $\cos\varphi$ and $\sin\varphi$ and neglecting the higher order terms. These approximations are accurate as long as φ is small. The result is

$$\begin{aligned} F &= (M_c + m_p)\,\ddot{z} + m_p l\ddot{\varphi} \\ m_p g\varphi &= m_p\ddot{z} + m_p l\ddot{\varphi} \end{aligned} \tag{3.5}$$

The approximate model of the system is completely discribed by φ, $\dot{\varphi}$, z and $\dot{z}$. To rewrite (3.5) in state space form, the state vector is written as

$$\mathbf{x} = [z \quad \dot{z} \quad \varphi \quad \dot{\varphi}]^T \tag{3.6}$$

The resulting state space description of the system is

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{m_p g}{M_c} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{(M_c+m_p)g}{M_c l} & 0 \end{bmatrix}\mathbf{x}(t) + \begin{bmatrix} 0 \\ \frac{1}{M_c} \\ 0 \\ -\frac{1}{M_c l} \end{bmatrix} F \\ \mathbf{y}(t) &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\mathbf{x}(t) \end{aligned} \tag{3.7}$$

Since the cart is driven by a DC motor the force that acts on the cart must be transformed to the input voltage [5]. This can be done with the help of Figure 3.3. Assuming negliable

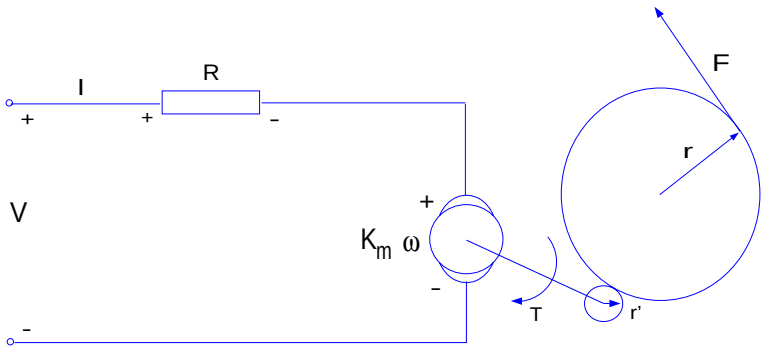

Figure 3.3. *A principle sketch over the motor.*

armature inductance, the electrically equation becomes

$$\begin{aligned} V &= IR + K_m\omega = \\ &= IR + \frac{K_m K_g}{r}\dot{z} \end{aligned} \tag{3.8}$$

where

V	Input voltage (Volt)
I	Armature current (Ampere)
R	Armature resistance (Ohm)
K_m	Motor torque constant (Nm/Ampere)
K_g	Gear ratio of the gearbox (dimensionless)
r	Radius of the output gear (Meter)
ω	Angular velocity of motor shaft (rad/s)

The torque produced at the motor shaft is given by the mechanically equation

$$\tau_m = K_m I \tag{3.9}$$

which, after the gearbox becomes the output torque.

$$\tau = \tau_m K_g = K_m K_g I \tag{3.10}$$

The torque creates a force at the output gear that is

$$F = \frac{\tau}{r} \tag{3.11}$$

Using (3.8), (3.10) and (3.11) the resulting relation between the input voltage and the force on the cart is

$$F = \frac{K_m K_g}{Rr} V - \left(\frac{K_m K_g}{r}\right)^2 \frac{1}{R}\dot{z} \tag{3.12}$$

Substituting (3.12) into the state space equation gives

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{1}{RM_c}\left(\frac{K_m K_g}{r}\right)^2 & -\frac{m_p g}{M_c} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{1}{RM_c l}\left(\frac{K_m K_g}{r}\right)^2 & \frac{(M_c+m_p)g}{M_c l} & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ \frac{K_m K_g}{rRM_c} \\ 0 \\ -\frac{K_m K_g}{rRM_c l} \end{bmatrix} V \\ \mathbf{y}(t) &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}(t) \end{aligned} \tag{3.13}$$

Using the numerical values of the constants in (3.13) gives

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -16.00 & -4.53 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 52.46 & 47.06 & 0 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 3.78 \\ 0 \\ -12.39 \end{bmatrix} V \\ \mathbf{y}(t) &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}(t) \end{aligned} \tag{3.14}$$

3.3 The controller

The method used to design a controller is here called LQ-control. Here, the controller is already given. The method uses a state space model like

$$\begin{aligned} \dot{x} &= Ax + Bu + Nv \\ y &= Mx \end{aligned} \tag{3.15}$$

and results in the state feedback vector L, such that the feedback law

$$u = -Lx \tag{3.16}$$

minimizes the cost function

$$V(L) = E\left[\int_0^\infty \left(y^T(t)Q_1y(t) + u^T(t)Q_2u(t)\right)\, dt\right] \tag{3.17}$$

The matrices Q_1 and Q_2 are cost matrices used to designate a weight or priority to each of the states and inputs. In the case with the inverted pendelum, the control of the angle is more important than the control of the position. There are only priorities on those two states. That is because only two states are measured. The other two states, the velocities of the cart and the angle of the rod, are estimated in SIMULINK with derivative blocks from the measured states. In [5] are the cost matrices that we use given.

$$Q_1 = \begin{bmatrix} 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 4.0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } \quad Q_2 = 0.0003 \tag{3.18}$$

The feedback vector L can be obtained from

$$L = Q_2^{-1}B^TS \tag{3.19}$$

where S is the positive semidefinite symmetrical solution to the Riccati equation

$$SA + A^TS - SBQ_2^{-1}B^TS + M^TQ_1M = 0 \tag{3.20}$$

In this case M is the identity matrix. By this we have got a closed system where the poles are the eigenvalues to the matrix $A - BL$. The matrix calculations can be performed by the MATLAB command `lqr`. The resulting feedback vector becomes

$$L = [28 \quad 31 \quad 137 \quad 12] \tag{3.21}$$

The vector is then fine tuned to the pendelum resulting in

$$L = [30 \quad 33 \quad 110 \quad 20] \tag{3.22}$$

Chapter 4

SIMULINK and dSPACE

In this section a short description is given of how the application to the inverted pendelum is done with a short presentation of SIMULINK followed by a presentation of the products from dSPACE.

4.1 SIMULINK

SIMULINK is a kind of extension to MATLAB which can be used to simulate dynamic systems. A system is built with different blocks such as step functions, sums, gains, integrators, differentiators or transfer functions. Variables defined in MATLAB can be used in blocks. They can also be fetched or sent to MATLAB. When the system have been built it can be simulated, using different types of numerical methods, for example Euler or Runge-Kutta methods. The controller that is used is implemented in SIMULINK and can be found in Appendix B.

4.2 dSPACE

The idea with the products from dSPACE is to use MATLAB and above all SIMULINK in realtime applications. A short manual to it is [10]. The principle of how dSPACE work with MATLAB and SIMULINK can be described with the flowchart in Figure 4.1. The controller is first created in SIMULINK where, as usual, parameters defined in MATLAB can be used. Given a model, there is a toolbox that directly generates C-code corresponding to the model. This toolbox is called Real-Time Workshop. The C-code is now used in different ways by dSPACE via its Real-Time Interface. Partly the C-code is compiled and downloaded to the DSP and partly a number of files are created that makes it easy to extract information from the simulation and change parameters. This is all done automatically. When the code is downloaded it starts running immediately and a number of things can be done. As help there are two tools which can be used in a couple of ways, TRACE and COCKPIT. More features, for example creating your own input signals to the process in MATLAB, are described in [10].

COCKPIT is a virtual instrument panel where it is possible to change parameters in a SIMULINK model. But it is also possible to study different signals practically in

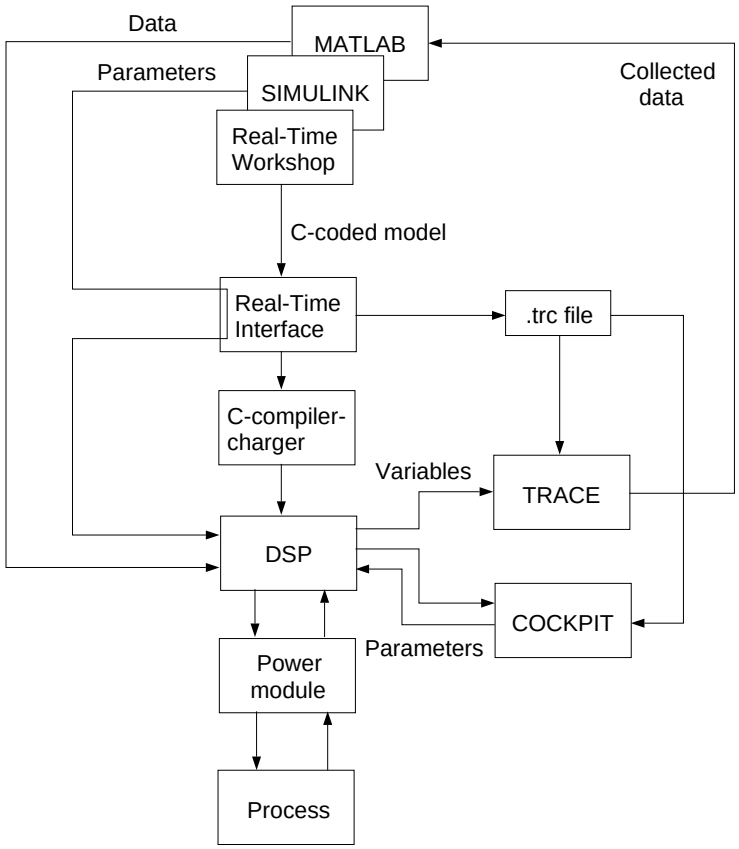

Figure 4.1. *Flowchart of how dSPACE work with* MATLAB *and* SIMULINK.

realtime. There can only be one COCKPIT connected to each DSP, therefore must the DSP be specified when starting the program. The instrument panel can be built with thirteen different kinds of instruments. The limitations in the COCKPIT program are first of all that in SIMULINK blocks where more than numerical values can be chosen, it is not possible to change parameters. It is also not possible to vary parameters in so called masked block in SIMULINK where one can choose more than just numerical values. The Figure 4.2 show the COCKPIT window in the inverted pendelum application.

Some blocks in SIMULINK don't work at all with dSPACE. Blocks that acts like that are those which plot curves. To deal with this problem the TRACE program have been made. The point with TRACE is to have the possibility to study certain signals more in detail and to easily load data to MATLAB. Which DSP we are working with must be specified here too when starting TRACE. The recieved plots are not in realtime. They appear with a delay which is approximately the chosen time interval. The main reason for this is that the communication between the DSP and the computer is done via Ethernet and therefore making it impossible to maintain the chosen sampling rate and at the same time transfer data. The TRACE window is shown in Figure 4.3.

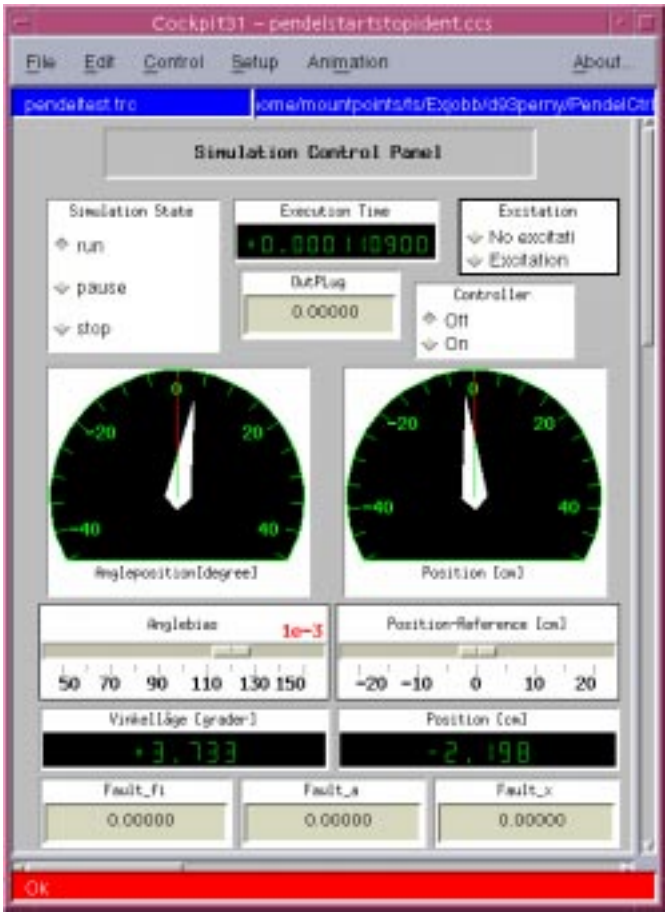

Figure 4.2. *The COCKPIT window.*

Figure 4.3. *The TRACE window.*

Chapter 5

Parity equations method

In this section, we use structured parity equations derived from a state-space model [1] as a diagnosis method on the inverted pendelum.

The strategy of parity equations is an open-loop strategy, i.e. there are no feedback from previously calculated residuals that utilizes what is called temporal redundancy which is a type of analytical redundancy, see Section 2.1.

With temporal redundancy it is possible, given analytical knowledge on the process behavior, to predict how process state and input signals will affect future outputs. Considering a time window, all information about any faults that may have occured during that time are present in the measurement.

Assuming that all signals acting upon a system are measurable, fault detection will be possible. This because the process variables can be extracted from the measured output signals, if the relatonship between process variables and measured output signals are known. But this is not always a realistic situation, therefore we need to make the diagnostic procedure invariant to unmeasurable inputs acting on the system. In order to isolate the faults we make the residuals insensitive to one or several of the other faults in order to distinguish between them. That makes it possible to see which fault that has occured and we have what is called structured parity equations.

5.1 Theoretical background

5.1.1 Fault detection

The first step is to generate the residuals. If we first consider the fault free case, then the discrete-time state-space model given in (2.2) is

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{5.1}$$

Since we are going to utilize temporal redundancy we need an expression for the output based on previous measurements. The output at time $t+1, t+2, \ldots, t+s, s > 0$ then

becomes

$$\begin{aligned} y(t+1) &= CAx(t) + CBu(t) + Du(t+1) \\ y(t+2) &= CA^2x(t) + CABu(t) + CBu(t+1) + Du(t+2) \\ &\vdots \\ y(t+s) &= CA^sx(t) + CA^{s-1}Bu(t) + \ldots + CBu(t+s-1) + Du(t+s) \end{aligned}$$

If we gather the equations in vectors we get

$$\mathbf{Y}(t) = \mathbf{R}x(t-s) + \mathbf{Q}\mathbf{U}(t) \tag{5.2}$$

where

$$\mathbf{Q} = \begin{pmatrix} D & 0 & \ldots & & 0 \\ CB & D & 0 & \ldots & 0 \\ CAB & CB & D & 0 & 0 \\ \vdots & \vdots & & \ddots & \\ CA^{s-1}B & CA^{s-2}B & \ldots & CB & D \end{pmatrix}$$

$$\mathbf{Y}(t) = \begin{pmatrix} y_1(t-s) \\ \vdots \\ y_m(t-s) \\ y_1(t-s+1) \\ \vdots \\ y_1(t) \\ \vdots \\ y_m(t) \end{pmatrix} \quad \mathbf{U}(t) = \begin{pmatrix} u_1(t-s) \\ \vdots \\ u_k(t-s) \\ u_1(t-s+1) \\ \vdots \\ u_1(t) \\ \vdots \\ u_k(t) \end{pmatrix} \quad \mathbf{R} = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^s \end{pmatrix} \tag{5.3}$$

Assuming k inputs and m measurements, the vector $\mathbf{Y}$ is $[(s+1)m]$ long and $\mathbf{U}$ is $[(s+1)k]$ long. Matrices $\mathbf{R}$ and $\mathbf{Q}$ have the dimensions $[(s+1)m \times n]$ and $[[(s+1)m] \times [s+1]k]$ respectively. Note that $y(t)$ and $u(t)$ are vectors and not scalar values.

In equation (5.2), $\mathbf{Y}$, $\mathbf{U}$ and $\mathbf{Q}$ are known. If we multiply with a vector w^T of length $[(s+1)m]$ and move all *known* variables to the left side we get

$$r(t) = w^T(\mathbf{Y}(t) - \mathbf{Q}\mathbf{U}(t)) = w^T\mathbf{R}x(t-s) \tag{5.4}$$

Equation (5.4) will now qualify as a residual (parity relation) if the residual is invariant to state variables, i.e.

$$w^T\mathbf{R}x(t-s) = 0 \quad \forall x \tag{5.5}$$

Given a vector w that satisfies (5.5) we have a residual generator where the left hand side of (5.4) is the computational form and the right hand side is the internal form.

If we now drop the fault free, no disturbance assumption made in (5.1), the residual generator (5.4) will then change to

$$r(t) = w^T(\mathbf{Y}(t) - \mathbf{Q}\mathbf{U}(t)) = w^T(\mathbf{R}x(t-s) + \mathbf{Q_a}\mathbf{F_a}(t) + \mathbf{T_c}\mathbf{F_c}(t) + \mathbf{T_d}\mathbf{D}(t) + \mathbf{S}(t)) \tag{5.6}$$

where

$\mathbf{F_a}$ is a vector of (unknown) actuator faults

$\mathbf{F_c}$ is a vector of (unknown) component faults

$\mathbf{D}$ is a vector of (unknown) disturbances

$\mathbf{S}$ is a vector of (unknown) sensor faults

It can be seen that $\mathbf{T_c}$ and $\mathbf{T_d}$ have the same structure as $\mathbf{Q}$ with B changed to H and E respectively and $D = 0$. $\mathbf{Q_a}$ has also the same structure as $\mathbf{Q}$ with $D = 0$.

If we now want a residual (5.6) to be insensitive to the unknown disturbances or actuator faults we add the additional constraint:

$$w^T \left[\mathbf{R}\ \mathbf{T_d}\ \mathbf{Q_a}\right] = [0\ 0\ 0] \tag{5.7}$$

where in $\mathbf{T_d}$ and $\mathbf{Q_a}$ only the columns in the B and D matrices corresponding to inputs to decouple are left.

If we instead want the residual to be insensitive to sensor faults we make sure that all w_i that appears in front of the sensor whose fault we wish to make the residual insensitive to are set to 0. This implies (s+1) zeros per sensor fault. If we have arranged $\mathbf{Y}(t)$ as in (5.3) and want to make the residual insensitive to faults in the $i : th$ sensor w gets the structure:

$$\begin{aligned} w \quad = \quad & (w_1(t-s), \ldots, w_{i-1}(t-s), 0, w_{i+1}(t-s), \ldots, w_m(t-s), \\ & , \ldots, \ w_1(t), \ldots, w_{i-1}(t), 0, w_{i+1}(t), \ldots, w_m(t))^T \end{aligned}$$

Example 5.1. Suppose

$$x = (x_1 \quad x_2)^T \qquad y = (y_\alpha \quad y_\beta)^T \qquad d = (d_1 \quad d_2)^T$$

If we now assume we have the model

$$\begin{aligned} \dot{x}(t) &= \mathbf{A}x(t) + \mathbf{B}u(t) + \mathbf{T_d}d(t) \\ y(t) &= \mathbf{C}x(t) + \mathbf{D}u(t) + f_s(t) \end{aligned} \tag{5.8}$$

where f_s(t) is a sensor fault. $\mathbf{A}$, $\mathbf{C}$ and $\mathbf{T_d}$ have the dimensions $[2 \times 2]$, and where $\mathbf{B}$ and $\mathbf{D}$ have the dimensions $[2 \times 1]$. The dimensions for $\mathbf{Y}$, $\mathbf{U}$, $\mathbf{R}$ and $\mathbf{Q}$ are then $[6 \times 1]$, $[3 \times 1]$, $[6 \times 2]$ and $[6 \times 3]$ respectively where $\mathbf{Y}$ has the shape

$$\mathbf{Y}(t) = \begin{pmatrix} y_\alpha(t-2) \\ y_\beta(t-2) \\ y_\alpha(t-1) \\ y_\beta(t-1) \\ y_\alpha(t) \\ y_\beta(t) \end{pmatrix} \tag{5.9}$$

Now if we want a residual to be insensitive to sensor faults in sensor α, the residual look like

$$w^T = (0, w_{\beta(t-2)}, 0, w_{\beta(t-1)}, 0, w_{\beta(t)})^T$$

If we instead want a residual to be insensitive to the disturbance d_1, the residual would satisfy the condition

$$w^T \left[\mathbf{R}\ \ [\mathbf{T_d}]_1\right] = [0\ 0] \tag{5.10}$$

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

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

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

Table 5.1. *Example coding sets*

5.1.2 Fault isolation

We now have the knowledge to make a residual insensitive to one or several of the other faults so we can therefore design a *bank* of residuals to achieve isolation. Let us look at an example.

In Table 5.1 three examples are presented and each row represents a residual. A 1 in position j on row i implies that fault f_j affects residual r_i. The different columns in the *coding sets* in Table 5.1 is called the *fault code*. A coding set is a table that describes how different faults affect the residuals.

If for example in coding set III residuals r_1 and r_3 fire while r_2 don't, i.e fault code $(101)^T$, it is probable that fault f_2 has occured.

To detect a fault, no column can have only zeros and to achieve isolation all columns must be unique. If these two requirements are satisfied, the coding set is called *weakly isolating*.

A small fault could trigger some residuals but not some other residuals. To avoid misisolation, the coding set should be constructed so that two columns cannot get identical when ones in a column are replaced by zeros. A coding set that satisfy this requirement is called a *strongly isolating* set.

In Table 5.1, coding set I is *non-isolating*, II is *weakly isolating* and III is *strongly isolating*.

It is not possible to make the residuals insensitive to an arbitrary number of disturbances and faults. The limit is

$$s_y + s_u \leq m - 1 \tag{5.11}$$

where s_y denotes the number of sensor faults that we want to decouple and s_u denotes the number of actuator faults and disturbances we want to decouple. More about that limit is explained in [1].

5.2 Application on the pendelum

When we linearized the model in Section 3.2, we did it at the $x_1 = x_2 = x_3 = x_4 = u = 0$. In order to get the matrix H in (2.2) we must also linearize at $dm_p = dM_c = 0$. The matrix we obtain is

$$\mathbf{H} = \begin{pmatrix} 0 & 0 \\ 0 & -\frac{1.72u(t)}{M_c^2} \\ 0 & 0 \\ \frac{1.72u(t)}{M_c m_p l} & \frac{1.72u(t)}{M_c^2 l} \end{pmatrix} \tag{5.12}$$

w_x	w_φ	w_a	w_{m_p}	w_{M_c}
0	0.8521	-0.0549	15.7830	4.8008
-3.2865	0	-0.0168	0.0006	1.4639
0	-3.5605	1.0000	-10.0001	-3.3294
9.5935	0	0.3050	0.0073	-1.0199
0	5.5651	-0.8352	-25.7829	-6.4689
-8.3093	0	-0.2555	0.0110	-1.9786
0	-3.8567	-1.1099	10.0000	3.7230
1.0000	0	-0.3394	0.0046	1.1340
0	1.0000	1.0000	10.0000	1.2746
1.0000	0	0.3049	0.0004	0.3886

Table 5.2. *Solutions to* $w^T\mathbf{R} = 0$.

Note that the elements are multiplied with the input u(t). We get this intuitively because if the pendelum don't move we cannot see if any component fault, i.e any change in a mass, has occured. This means that when u(t) are zero the residuals will be zero. We can now build the matrices $\mathbf{R}$, $\mathbf{Q}$ and $\mathbf{T_c}$ in order to calculate the residual generators. Residual generators are calculated for sensor faults x and φ, actuator fault a and component faults m_p and M_c.

$$\mathbf{R} = \begin{pmatrix} C \\ CA \\ CA^2 \\ CA^3 \\ CA^4 \end{pmatrix}$$

$$\mathbf{Q} = \begin{pmatrix} D & 0 & 0 & 0 & 0 \\ CB & D & 0 & 0 & 0 \\ CAB & CB & D & 0 & 0 \\ CA^2B & CAB & CB & D & 0 \\ CA^3B & CA^2B & CAB & CB & D \end{pmatrix}$$

$$\mathbf{T_c} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ CH & 0 & 0 & 0 & 0 \\ CAH & CH & 0 & 0 & 0 \\ CA^2H & CAH & CH & 0 & 0 \\ CA^3H & CA^2H & CAH & CH & 0 \end{pmatrix}$$

Finally we can get the vectors w which satisfy the condition (5.7). How it was solved is shown in Appendix A.

This gives us the vector w_x. Similar calculations were made for w_φ, w_a, w_{m_p} and w_{M_c}. That gives us the vectors in Table 5.2. The vectors are then used to get the residuals with (5.6). The residuals then gives us the coding set in Table 5.3. To fulfill the goal of having strongly isolating residuals the coding set was intended to have zeros in the diagonal of Table 5.3. This is not accomplished since there are zeros in the positions (r_φ, f_{m_p}), (r_{m_p}, f_φ) and (r_a, f_{M_c}), (r_{M_c}, f_a) of Table 5.3. The reason for this is because the faults enter the system dynamics in the same way giving a model that have the same

	f_x	f_φ	f_a	f_{m_p}	f_{M_c}
r_x	0	1	1	1	1
r_φ	1	0	1	0	1
r_a	1	1	0	1	0
r_{m_p}	1	0	1	0	1
r_{M_c}	1	1	0	1	0

Table 5.3. *Coding set for the pendelum.*

structure in the corresponding cases. Looking in the case of the zeros in (r_a, f_{M_c}) and (r_{M_c}, f_a) and using (3.5) where we add a mass change, $\triangle M_c$ and an actuator fault, f_a which gives (5.13).

$$\begin{aligned} (M_c + \triangle M_c + m_p)\,\ddot{z} &= F - m_p l\ddot{\varphi} + f_a \\ (M_c + m_p) &= F - m_p l\ddot{\varphi} + \underbrace{f_a - \triangle M_c \ddot{z}}_{\tilde{f}_a} \end{aligned} \tag{5.13}$$

making it impossible to seperate f_a and $\triangle M_c$. Looking in the same case instead at the structure of the model, the result becomes as in (5.14),

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + Pf_{a,c}(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{5.14}$$

where P have the same structure in both cases.

The goal was to have zeros in the diagonal. Instead we now have columns which are identical, f_φ, f_{m_p} and f_a, f_{M_c}. This makes it impossible to isolate these faults. If we look only on sensor and actuator faults we can get a strongly isolating set.

A problem that arose was that it was hard to get good vectors w. The reason was that the matrices $\mathbf{Q}$ and $\mathbf{T_c}$ were badly conditioned which in turn was because of the model's matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$. This was solved by using Singular Value Decomposition (SVD).

5.3 Simulated results

To simulate the system we use SIMULINK. We use the discrete model and the controller which were derived in Section 3.2 and 3.3, and the MATLAB function described in Appendix A. This gives the implementations in Appendix B. The faults we are looking at are bias fault on sensors and actuator and component faults as a change in a mass.

5.3.1 The fault free case

In the fault free case we can see in Figure 5.1 how close to zero all the residuals are. The reason that they are not exactly zero is that the condition (5.5) is not exactly zero.

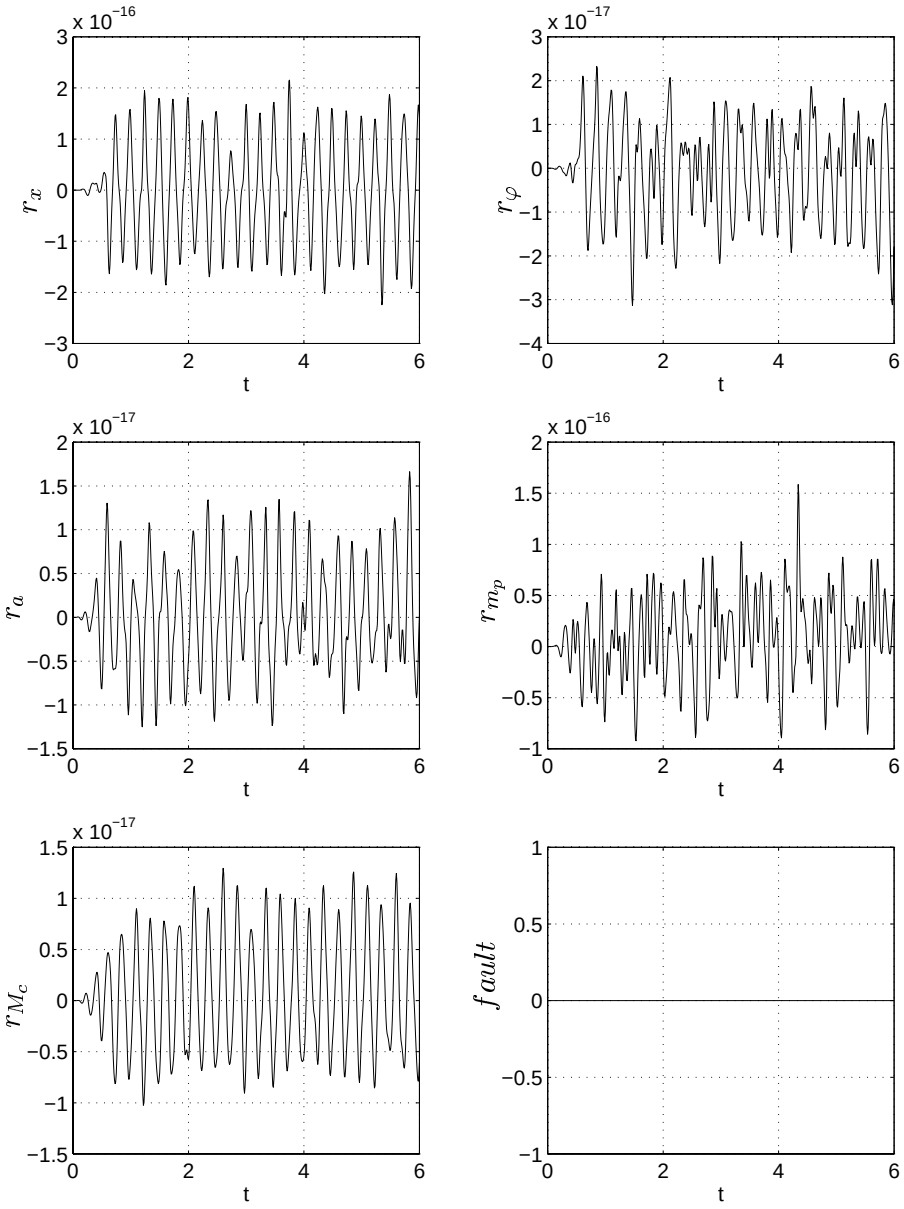

Figure 5.1. *Simulated residuals in a fault free case.*

5.3.2 Fault in position sensor

If we now add a fault to the position sensor with the magnitude of 5 centimeters, we get the residuals in Figure 5.2. The residual r_x is nearly zero as it is expected to be as seen by column 1 in Table 5.3. The other four residuals makes a distinct reaction when the fault occur and then returns to be close to zero. From the definitions in Section 2.2 we find that the faults are not strongly detectable, but detectable, when the residuals only reflects the changes on a fault.

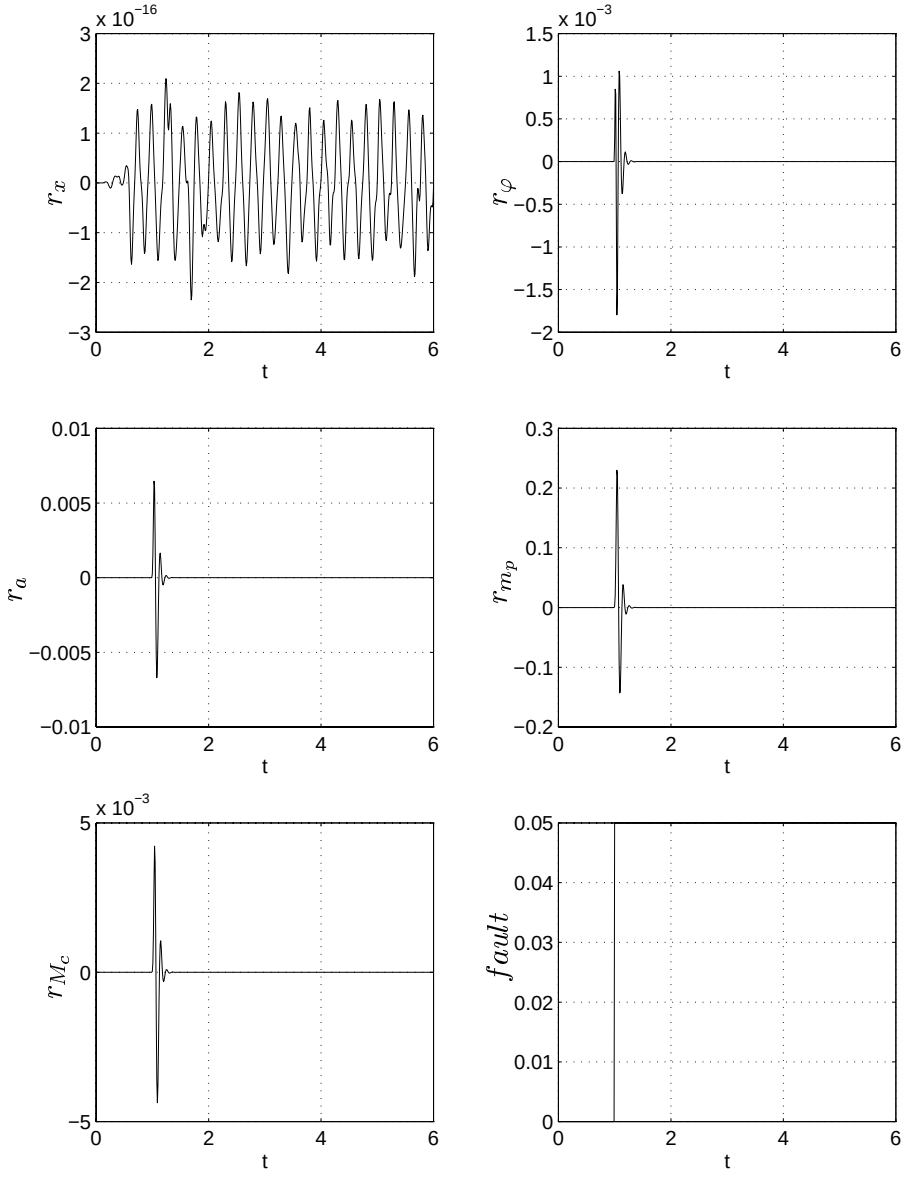

Figure 5.2. *Simulated residuals when a fault in the position sensor occur at t=1.*

5.3.3 Fault in angle sensor

When we instead add a fault to the angle sensor with the magnitude 0.05 radians, or approximatly 3 degrees, the residuals work as expected in this case too, as we can see in Figure 5.3. The residual for the angle r_φ is close to zero and the others react when the fault occur as seen by column 2 in Table 5.3. In this case the residuals don't return so close to zero, which is a little bit better. In fact, the residual for the rod's mass, r_{m_p}, doesn't return at all to zero which is how we want a residual to react. Again from the definitions in Section 2.2 we find that the faults are not strongly detectable, but detectable, when the residuals only reflects the changes on a fault. The residual r_{m_p} is though strongly detectable. Here, the residuals are smaller than before.

Comparing with the coding set in Table 5.3 we find that the residual r_{m_p} react which

it shouldn't do at all. This is not what was expected.

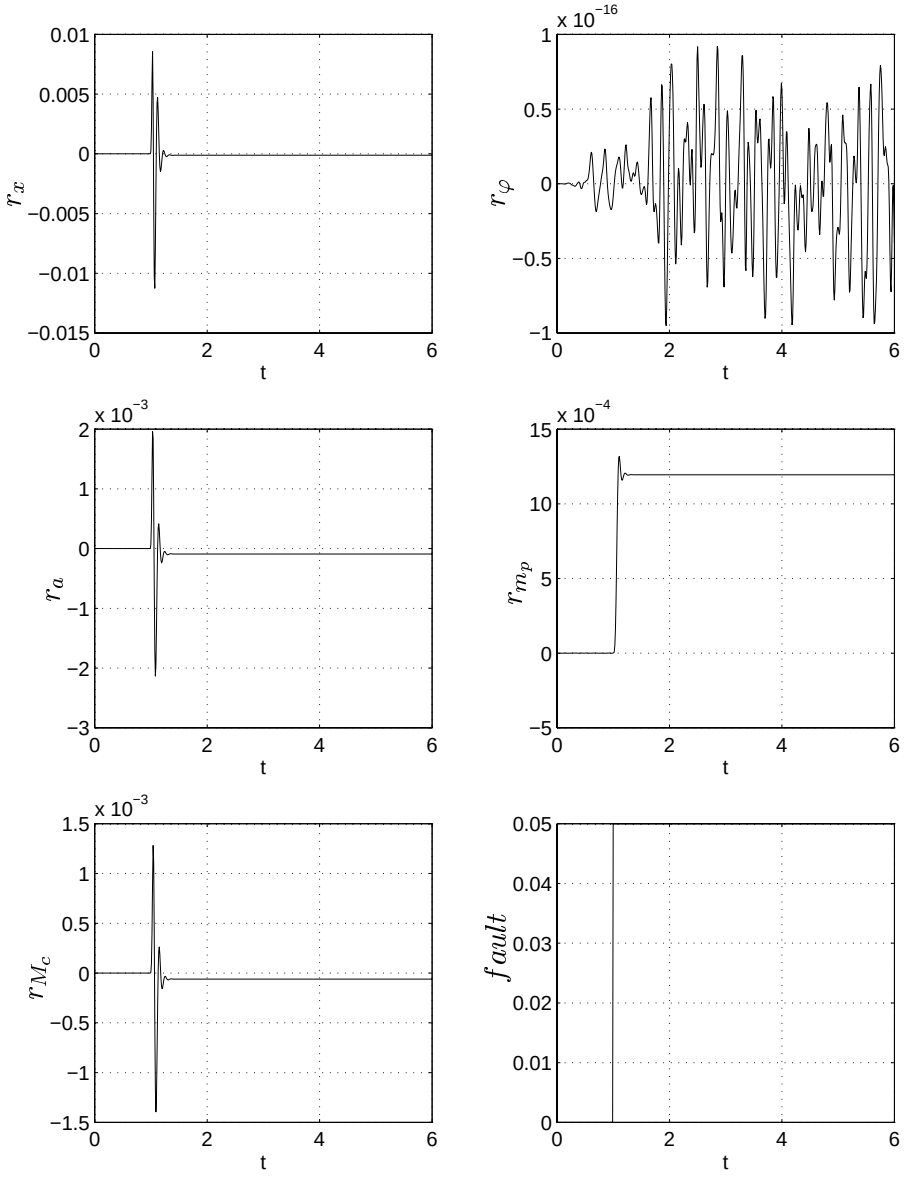

Figure 5.3. *Simulated residuals when a fault in the angle sensor occur at t=1.*

5.3.4 Fault in actuator

One fault was added to the actuator and had the magnitude of 1 Volt, see Figure 5.4. The residuals worked as expected here too. Those residuals that should react do that while the residual for the actuator is still. Here, the two residuals for the masses, the rod and the cart, don't return against zero. The residual for the mass of the cart, r_{M_c}, is very small which affect the fault isolation on the real process. On the real process there is a risk that the fault information will be buried in noise.

Comparing with the coding set here we find that the residual r_{M_c} react which it

shouldn't do. This is also not what was expected.

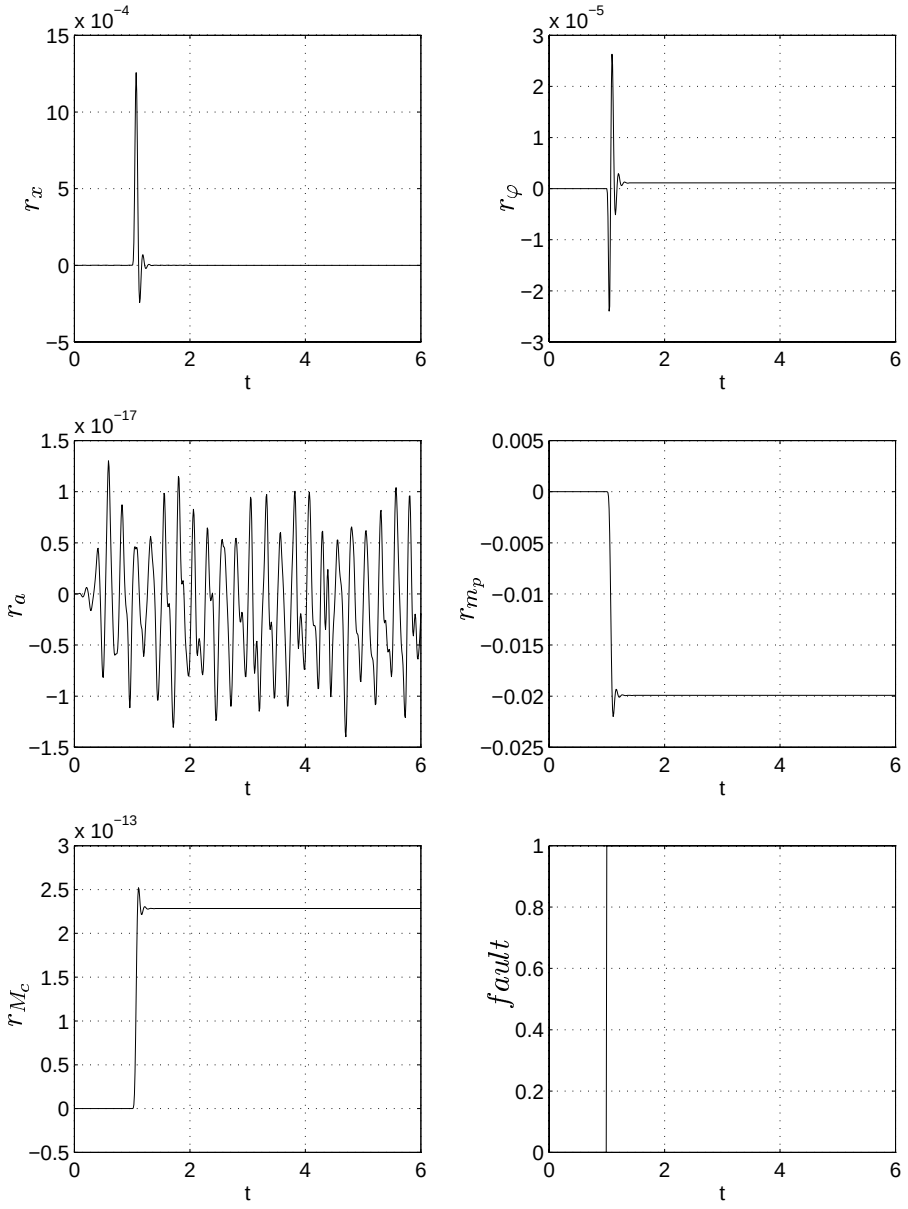

Figure 5.4. *Simulated results when a fault in the actuator occur at t=1.*

5.3.5 Component faults

When changing a mass in the model and simulating, the results didn't became as expected. No residuals gave any reaction, in spite of the large mass changes, 88 % for the mass of the cart and 48 % for the mass of the rod. This was not expected. A reason for this could be the input signal $u(t)$ in matrix H in (2.3). It makes it possible to rewrite the model so that the matrices B and H can be put together to one matrix B' making

the model to become like (5.15).

$$\begin{aligned} \dot{x}(t) &= Ax(t) + B'u(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{5.15}$$

Comparing with the coding set we find that the residuals for component faults are the same as the residuals r_φ and r_a making it impossible to isolate these faults.

5.3.6 Tests

Since the residuals for the masses M_c and m_p didn't seemed to work for component faults, we instead looked at how our sensor and actuator residuals would react when a sensor or actuator fault occured and a mass had been changed. The result were that the residuals variance increased and some residuals, especially during an actuator fault, didn't behave as we wanted. Also the significance between a fault and a fault free situation decreased. This means that the method is not especially robust.

The number of previous time measurements, s, has in [1] been chosen to the same as the system order. An experiment was made where s was increased, from four to six. What happened was that the residuals didn't change any of their reactions but their magnitudes did change. Since its always possible to normalize the residuals, increasing the number of time measurements is not something that can be used to improve the results in this case. No further investigation has been done because of this.

5.4 Experimental results

In order to perform the experiment, we add to the signal that we want to have a fault, a step in SIMULINK. The step is then controlled in dSPACE so that it can be active when we want it to be. We then log the input and output signals under a time period and invoke the step during this period. The logged data are then used by a MATLAB script that calculates the residuals.

There are always difference between a simulation and an experiment and that is the difference between the process and the model and measurement noise. In this case there is also no disturbance in the simulations. During the experiments though, there are noise which we have tried to filter with LP-filters that through experimentation have their cutoff frequency at 10 Hz. Here, we will also see that when the input signal is zero the residuals will be zero. In this case there will be another difference between simulated and experimented results. The experiment with an actuator fault didn't work which in the simulated case it did.

5.4.1 The fault free case

The residuals in the fault free case are shown in Figure 5.5. Here we see that the residuals includes noise compared to the simulated ones. Now, they are much more insecure. From the results here we arbitrary include thresholds on the residuals, which are the dotted lines in the Figures 5.5, 5.6, 5.7 and 5.8,

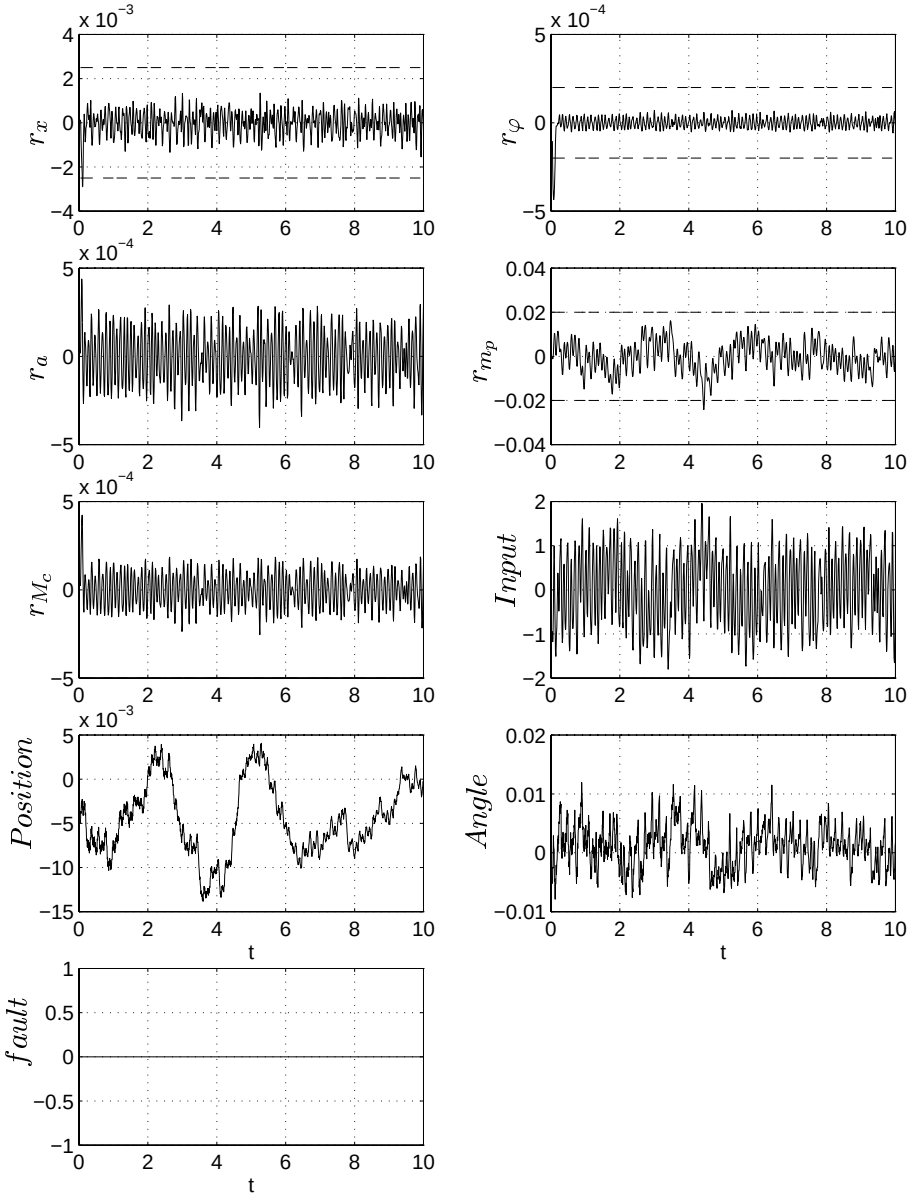

Figure 5.5. *Residuals in a fault free case.*

5.4.2 Fault in position sensor

When we add a fault to the position sensor, which results in Figure 5.6, we can see that all the residuals react. Even the residual r_x react a little bit. This is not something it should. This is because the condition $w^T \mathbf{R} x(\text{t-s}) = 0$ is fulfilled for the model but the real process' R differs from the model's. If the residual r_x can be improved, for example by a better filter, then localisation of position fault are possible in this case.

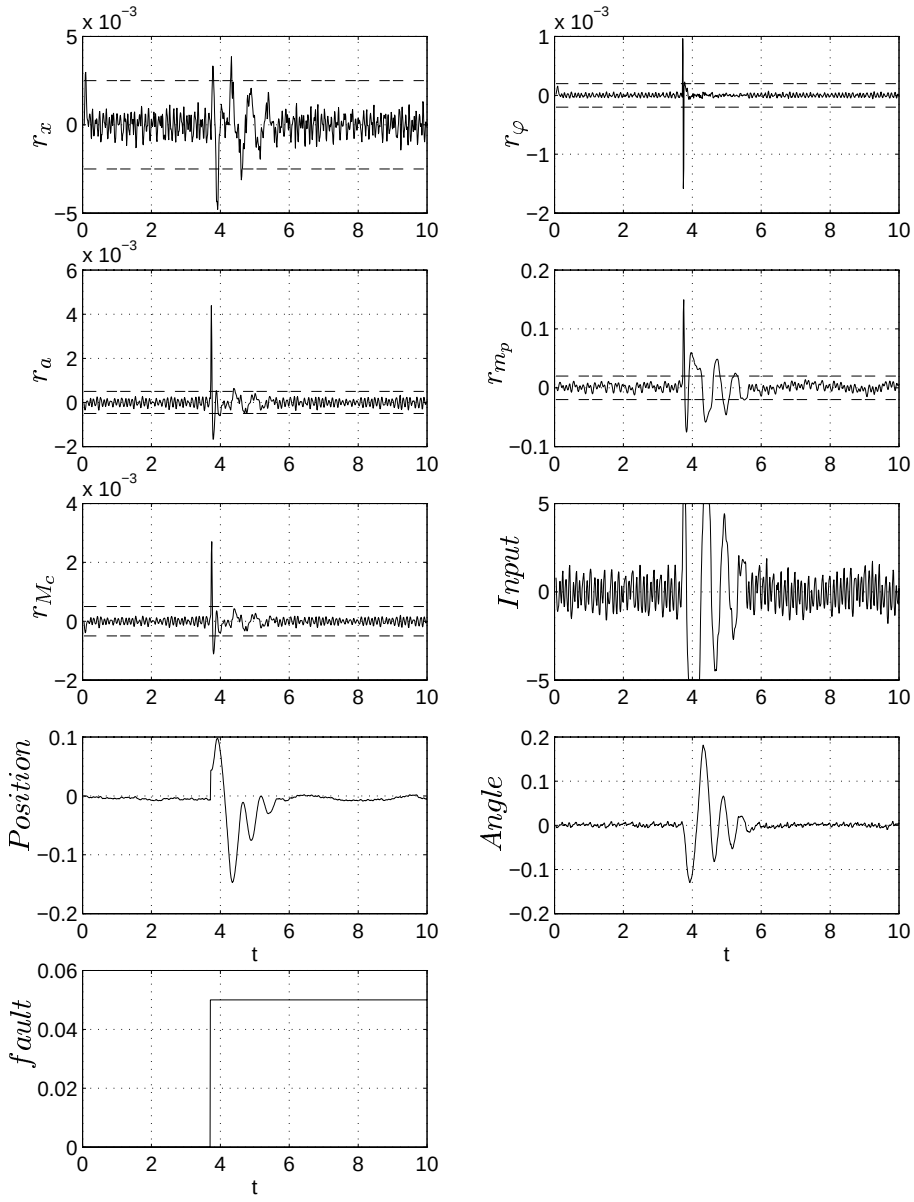

Figure 5.6. *Residuals when a fault in the position sensor occur.*

5.4.3 Fault in angle sensor

Adding a fault to the angle sensor results in Figure 5.7. Here we can see that the residual r_φ also reacts at the fault. But its reaction is less than what the residual r_x did when a fault occured on that sensor. The condition (5.5) is then better fulfilled for an angle fault than for a position fault.The other residuals reactions are smaller than compared to the result on the fault in the position. Still, it is possible to locate a fault in the angle sensor in this case. A reflection to be made is that in this experiment, we could see on the output signal for φ, named *Angle*, a change the size of the fault, see Figure 5.7. If we observe the experiment with a fault in the position sensor, there is no such change in the output signal for the position, named *Position*, see Figure 5.6. This is because

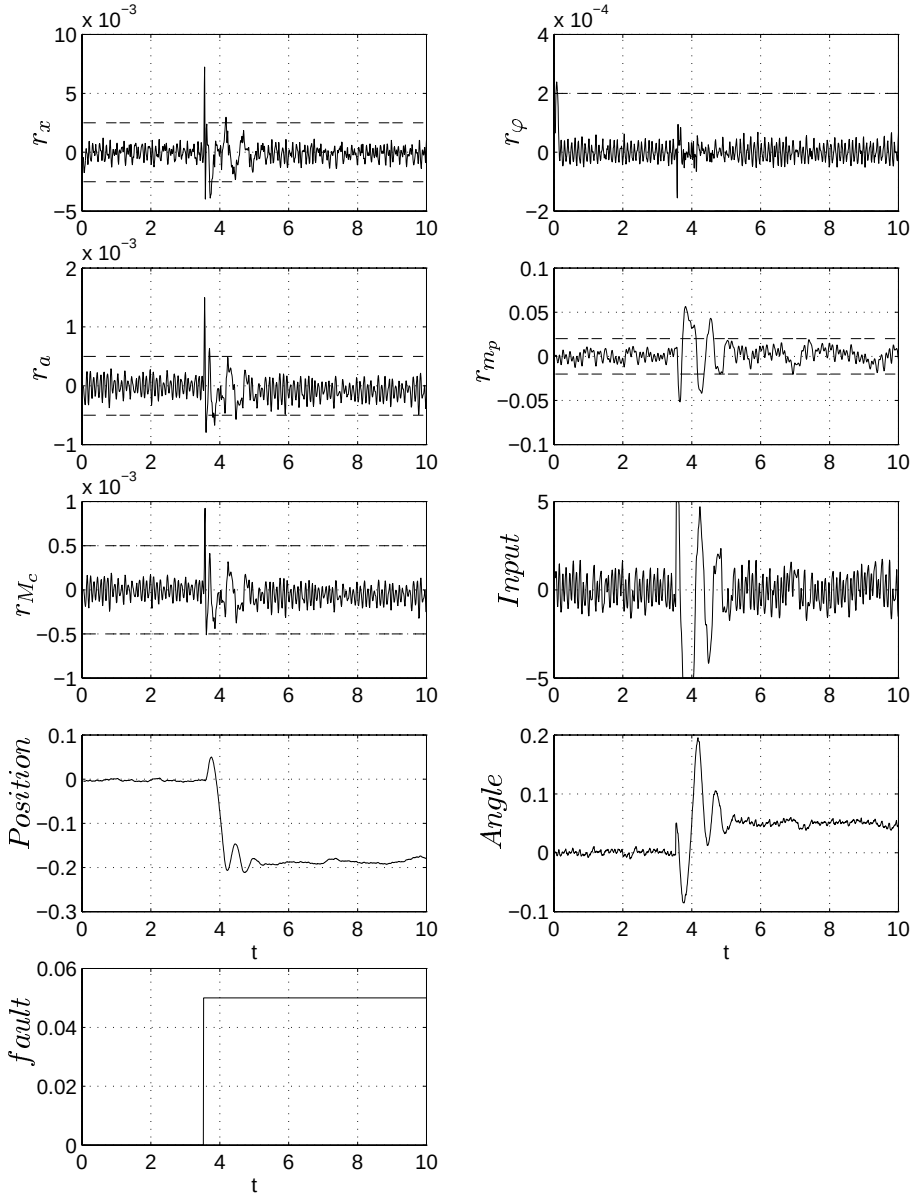

Figure 5.7. *Residuals when a fault in the angle sensor occur.*

Position have no trouble keeping its reference, but the *Angle* must keep the rod vertical to control it causing this to show the fault.

5.4.4 Fault in actuator

When we add a fault to the actuator we notice in Figure 5.8 that the residuals don't work at all as expected. There are no reaction at all from any of the residuals, except a very small one from the residual for the pendelum's mass. In this case it is not possible to locate a fault in the actuator.

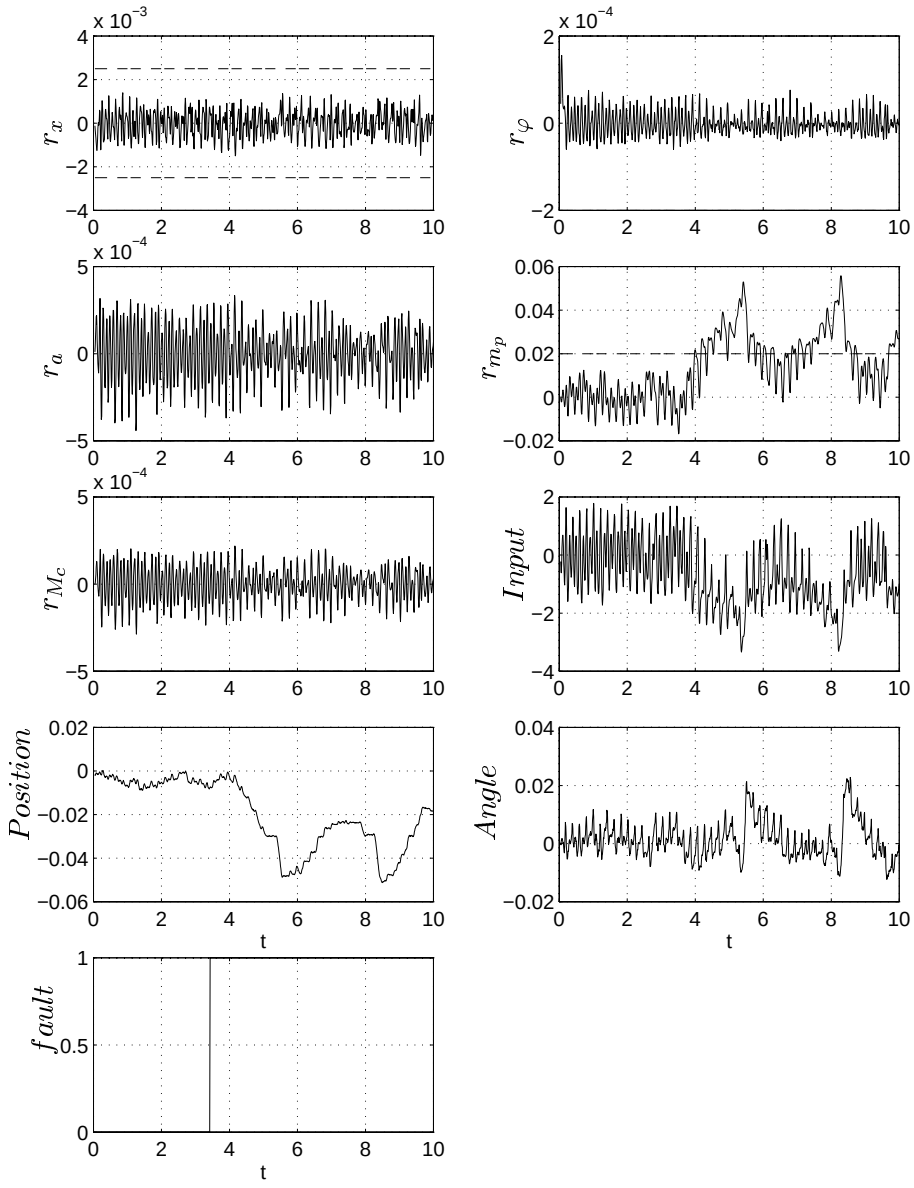

Figure 5.8. *Residuals when a fault in the actuator occur.*

5.4.5 Component faults

Adding a mass to the pendelum or on the cart didn't made the residuals to work as we wanted either. The conclusion must be that it is not possible to locate any fault on this process.

5.4.6 Tests

Since the residuals for the masses M_c and m_p didn't seemed to work for component faults, we instead looked at how our sensor and actuator residuals would react to a change in a mass when a sensor or actuator fault also occured. The result were that the

residuals variance increased and some residuals, especially for an actuator fault, didn't work. Also the significans between a fault and a fault free situation decreased.

In order to improve the residuals an attempt were made where the sampling period was changed to a slower period to see if it made any difference, which in this case it didn't. Instead the residuals became worse probably because the signals became less substantial when we changed the period.

Another attempt to improve the residuals were an experiment where the controller was sampled at one sample period and the residual generator at another. This got, on a sensor fault, all the residuals to react.

The same experiment with changing the number of previous time measurements, see Section 5.3.6, was made in this case too. The result became the same as before. The residuals reactions didn't change but their magnitudes on the reactions did.

5.5 Problems

The first problem that arose were that there was a difference between the model and the real process. That made it a bit difficult during simulations since the controller didn't work perfectly with the model. It also made the results from simulations and experiments to differentiate.

The next problem was to achieve good vectors w. In this case the matrics $\mathbf{Q}$ and $\mathbf{T_c}$ were badly conditioned, which was due to the model's matrices. This was solved by using Singular Value Decomposition (SVD).

During the simulations the residuals were discovered to be weakly detectable. A possible solution that was never tried here is to filter the residuals, e.g through an integrating filter.

The biggest problem was that the method didn't work during simulations. Just look at the component faults where nothing happened. The coding set that was derived didn't match the actual results.

5.6 Comparison with articles

Since no articles have been found where this method has been applied to an inverted pendelum, comparisons against observermethods are done instead since the methods are related.

Comparing with [7] we find that there two of three sensor fault and the only actuator fault be detected and located. The sensor fault that can't be located is a sensor for the velocity of the cart. Here that sensor doesn't exist but the two sensor fault could be detected and located as in the article. There is one difference and that is that the method in the article is more robust towards parameter variations than the parity equation method.

Comparing with [9] we find that the two observer based methods in that article can detect and locate three sensor fault. The third sensor is, as in the previous comparison,

a sensor for the velocity of the cart. Parity equation method could as noted before detect and locate the other two faults.

5.7 Conclusions

The conclusions of this method would be that

- It is a method that is easy to understand.
- The model that is used to determine the necessary matrices must be precise. Here, we could use the model to design a controller, but the model couldn't be used to design the residuals.
- If there are any numerical problems to calculate the vectors w, then the only thing we can do is to use SVD to get vectors that can be used.
- Sensor and actuator faults could be detected and isolated during simulations. Only sensor faults could be detected when the method was used on the process. But component fault could not be detected in either cases.
- The coding set that was derived didn't match the actual results.

Chapter 6

Parameter estimation method

In this section we use parameter estimation as a diagnosis method on the inverted pendelum [6, 7]. This is another method to utilize the analytical redundancy.

A parameter estimation method is based on the idea that faults are related to changes of the system parameters. Fault detection and isolation can then be performed by a suitable comparison of the estimated physical parameters to their nominal values. But when discretizing the continuous-time model, the close relationship between the physical parameters and those of the continuous-time model is lost. Therefore, fault localisation is less intuitive in discrete-time case than in the continuous-time case and most of these methods will therefore only perform fault detection. Some of the techniques combine state space methods with identification techniques for localisation.

Here we will use a method for fault detection and localisation based on discrete-time identification. This is because the signals that is received from dSPACE to SIMULINK and MATLAB are discrete. For this method, the relations between the physical parameters and those of the discrete-time model need not be known. The principle is described in Figure 6.1.

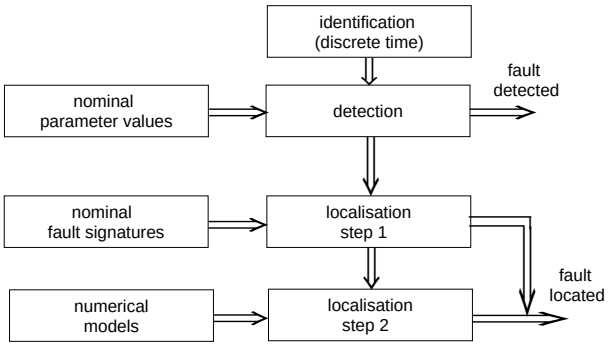

Figure 6.1. *Principle of FDI based on discrete-time identification.*

For identification, in order to obtain reliable models, the input signals must fulfil the condition of being persistently exciting, i.e the signals have to excite all system modes during the identification procedure. If the input signals aren't persistently exciting they must be enhanced in some way. This can be done for example by using additional input signals. This could degrate the performance of the system. A solution would be to use

a test sequence. Then the method would not be suited to detect rapidly acting faults like sudden sensor or component break downs.

For application of FDI methods based on parameter estimation, it has to be distinguished between supervision of open loop and closed loop systems. When identification has to be performed in a closed loop, the signals are filtered by the controller and hence the condition of persistent excitation may not be fulfilled.

Compared to the number of Component Fault Detection (CFD) methods, only a few methods deal with instrument fault detection and localisation. In [8] a method is reported, performing fault detection and localisation of slowly acting sensor gain and sensor bias faults in closed loop systems. However, here we will only look at component faults because the method is well suited for it.

6.1 Theoretical background

6.1.1 Identification

For system identification, linear regression model structures are very useful in describing basic linear and nonlinear systems [3]. In order to estimate the nonmeasurable model parameters we write the model as

$$y(t) = \varphi^T(t)\theta \tag{6.1}$$

where $\varphi(t)$ consists of measurable inputs and outputs in a discrete model and output deratives in a continuous model. θ are the model parameters to be estimated.

Example 6.1. For an ordinary linear differential equation

$$\begin{aligned} &y(t) + a_1\frac{dy(t)}{dt} + a_2\frac{d^2y(t)}{dt^2} + \ldots + a_n\frac{d^ny(t)}{dt^n} = \\ &= b_0u(t) + b_1\frac{du(t)}{dt} + b_2\frac{d^2u(t)}{dt^2} + \ldots + b_m\frac{d^mu(t)}{dt^m} \end{aligned}$$

we get

$$\begin{aligned} \varphi(t) &= \left[-\frac{dy(t)}{dt} \; -\frac{d^2y(t)}{dt^2} \; \ldots \; -\frac{d^ny(t)}{dt^n}\right. \\ &\qquad \left. u(t) \; \frac{du(t)}{dt} \; \frac{d^2u(t)}{dt^2} \; \ldots \; \frac{d^mu(t)}{dt^m}\right]^T \\ \theta &= [a_1 \; a_2 \; \ldots \; a_n \; b_0 \; b_1 \; \ldots \; b_m]^T \end{aligned}$$

Note that θ is the *model* parameters, not the *physical* parameters. In a continuous time model, θ could have been written as a function of the physical parameters p as

$$\theta = f(c) \tag{6.2}$$

But when using discrete time models, the relation between the discrete time model parameters and the physical parameters becomes complex making it difficult to use the relation for analytical fault localisation.

When the model structure is determined we use the measurable input and output signals in order to estimate the parameter vector θ. This can be done with, for example, the Least Squares method (LS) where we minimize the quadratic estimation error

$$V_N(\theta) = \sum_{i=0}^{N} \Big(y(i) - \varphi^T(i)\theta\Big)^2$$

and achieve the solution

$$\hat{\theta} = \Big[\varphi(t)^T \varphi(t)\Big]^{-1} \varphi^T y$$

If we have time varying parameters, we can instead choose the Recursive Least Squares method (RLS) with forgetting factor which eliminates that problem.

6.1.2 Identification in a closed loop

Because of the system's unstable character we have to identify the system in a closed loop. This is troublesome since the controller will interfere in the identification, or in the worst case it will be the controller that is identified. To succeed with an identification experiment we must determine if the experiment is informative enough. In [3] it is stated that we can retrieve a correct description of the true system with a prediction-error method (ARX, ARMAX or BJ) if we have an informative data set. In general, nonlinear or time-varying or noisy or complex (high-order) controllers yield experiments that are informative enough. A more exact condition dealing with time-varying controllers is to let the input signal be given by

$$u(t) = F_i(q)y(t) + K_i(q)w(t) \qquad i = 1, 2, \dots, r$$

Here F_i and K_i are linear filters that are changed during the experiment between r different ones. With an extra input signal $w_j(t)$ that passes through filters without any zeros on the unit circle, there will always be an informative data set. Comparing with Figure 6.2, we see that w(t) can be considered as an additional extra input signal w_1 (measurable), as noise w_2 in the controller (unmeasurable), as set point changes (reference signal) in the controller w_3, or as a combination of these effects. Here though, we don't have a time-varying controller.

One condition that *must* be fulfilled in order to have an informative data set is to have an input signal that is persistenly exciting. With a persistenly exciting signal we excite all of the system modes. Such a signal has its main energy in frequencies that is important for the system. If we look at the Bode plot of our system in Figure 6.3, we see in the Gain plot that it has a hump where most of the information lies. After the hump there is not much information. We chose to let the crossover frequency, ω_c, be where the curve is three decibel lower than where the curve cross the initial gain, i.e when the curve is approximatly -51 dB. Hence we want an input signal that has its energy beneath the crossover frequency that is approximatly 63 rad/s.

Looking at the Bode plot gives us more than just how the input signal should look like. It also gives the sampling interval, T_s. In [3, 4] it is said that about ten times the

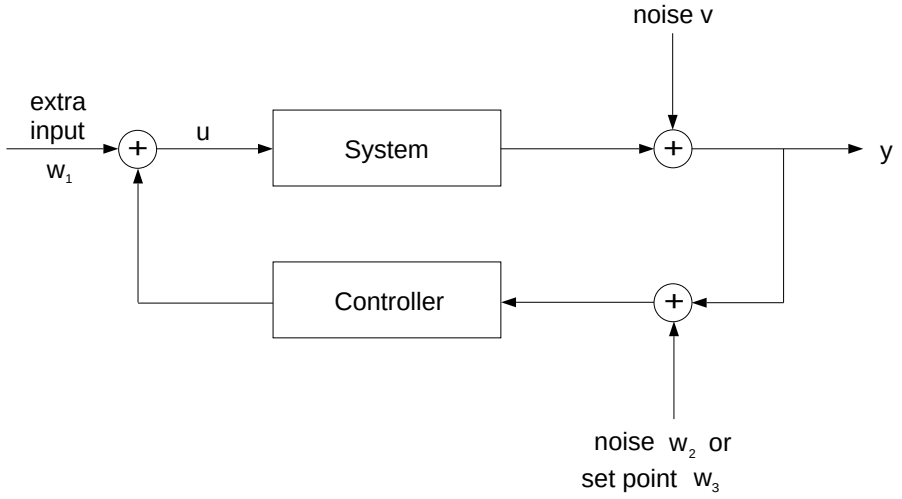

Figure 6.2. *Block diagram of a typical feedback system.*

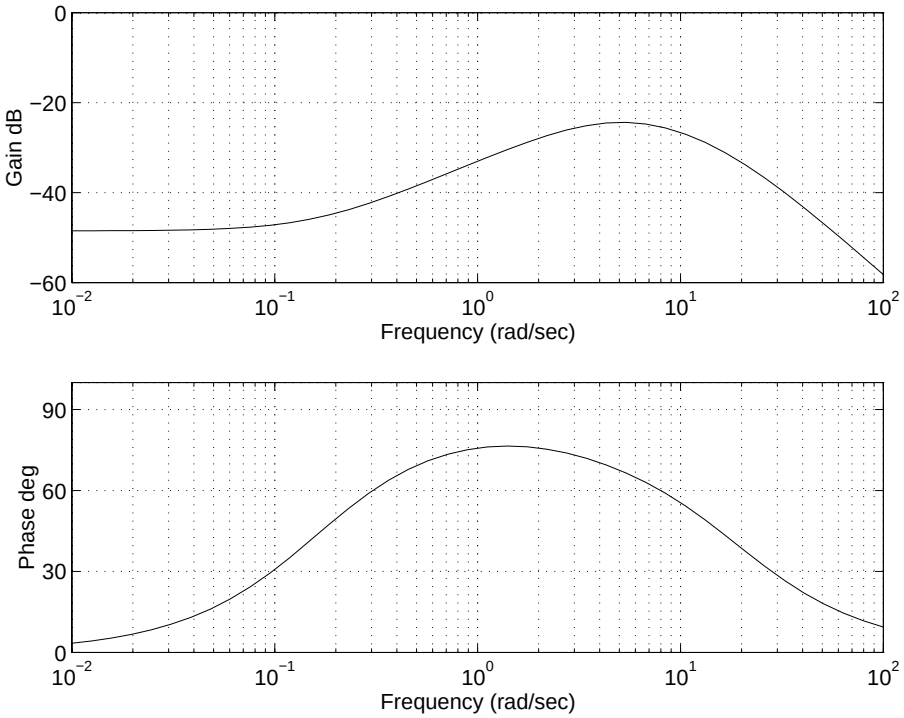

Figure 6.3. *Bode plot of the inverted pendelum.*

bandwith, ω_b of the system is a good choice of sampling interval. A rough estimate of the bandwith is to let it be equal to the crossover frequency. If we do that we get a bandwith that is 63 rad/s which will give a sampling interval that is approximatly 10 ms since $10\omega_b = \frac{2\pi}{T_s}$.

Some specific techniques for identifying systems in a closed loop have been developed. For example, stating that y and u in Figure 6.2 are outputs of another system, driven by disturbances and extra inputs (joint input–output identification). But in [3] we are adviced to apply prediction-error methods in a direct fashion, without taking special

measures to cope with the feedback. If they fail, no other approach will succed.

If we don't allow any extra input signals, we can get informative experiments if we shift between different linear controllers. In [3] it is said that it can be sufficient to use two controllers

$$u(t) = F_1(q)y(t) \quad \text{and} \quad u(t) = F_2(q)y(t)$$

subject to

$$\left[F_1(e^{i\omega}) - F_2(e^{i\omega})\right] \neq 0 \quad \forall\omega$$

This is a useful way of acheiving informative experiments when the requirement of good control is stringent.

In [6] a third method is used which also is used here. The extra input signal, w_1 in Figure 6.2, is used as the identification input. Then the whole system is identified (the pendelum and the controller).

6.1.3 Fault detection and isolation

For fault detection we compare the obtained parameter estimates to their nominal values which allows the detection of a change in the system. When we check for each parameter if its value increases or decreases, we obtain the fault signature. In [6] has Wald's Sequential Probability Ratio Test (SPRT) been chosen for the fault detection.

Wald's sequential probability ratio test examine from information which of two hypothesis are true. Two thresholds determines when the value of the test is good enough to support one of the two hypothesis. If the value is between the thresholds no conclusion can be made and more information is needed. To perform a *sequential test* of H_0 versus H_a, we gather individual observations one at a time, and assess whether the accumulated information is enough to choose H_0 before H_a or viceversa. This is done in a series of seperate steps:

Step 0: Begin by setting two constants A and B such that $0 < \text{A} < \text{B}$.

Step 1: Study the observation X_1. Compute the probability ratio $\Lambda_1 = \frac{f_1(x_1)}{f_0(x_1)}$. Since very large values of this ratio support H_a, specify rejection of H_0 if Λ_1 equals or exceeds $\text{B} > 0$, i.e

if $\Lambda_1 \geq \text{B}$, reject H_0

Alternatively, since very small values of this ratio support H_0, specify acceptance of H_0 if Λ_1 equals or drops below $\text{A} > 0$, i.e

if $\Lambda_1 \leq \text{A}$, accept H_0

An important feature of the sequential approach is the allowance for an indeterminate outcome, i.e no conclusion when not enough substantive information is available either to accept or reject H_0. This occurs if $\text{A} < \Lambda_1 < \text{B}$:

if $\text{A} < \Lambda_1 < \text{B}$, continue gather observations $\Rightarrow$ study X_2

Step 2: Study X_2. Compute the probability ratio $\Lambda_2 = \frac{f_1(x_1,x_2)}{f_0(x_1,x_2)}$. As in Step 1, if $\Lambda_2 \geq B$, reject H_0, while if $\Lambda_2 \leq A$, accept H_0. If

A $< \Lambda_2 <$ B, continue gather observations and study X_3.

$\vdots$

Step N_L: Study X_{N_L}. Compute the probability ratio test $\Lambda_{N_L} = \frac{f_1(x_1,x_2,\dots,x_{N_L})}{f_0(x_1,x_2,\dots,x_{N_L})}$. As in Step 1, if $\Lambda_{N_L} \geq$ B, reject H_0, while if $\Lambda_{N_L} \leq$ A, accept H_0. If A $< \Lambda_{N_L} <$ B, continue gather observations and study X_{N_L+1}.

This method is known as a *Sequential Probability Ratio Test (SPRT)*, due to Wald [11]. Notice that in the common setting where the individual and independant observations are gathered from $f_0(x)$ or $f_1(x)$, the probability ratios take the form

$$\Lambda_N = \frac{f_1(x_1, x_2, \dots, x_N)}{f_0(x_1, x_2, \dots, x_N)} = \frac{\prod_{i=1}^N f_1(x_i)}{\prod_{i=1}^N f_0(x_i)} \tag{6.3}$$

Then, e.g, the continuance condition A $< \Lambda <$ B is equivalent to

$$\log A < \log \left(\prod_{i=1}^N \frac{f_1(x_i)}{f_0(x_i)} \right) < \log B \tag{6.4}$$

or

$$\log A < \sum_{i=1}^N \log \frac{f_1(x_i)}{f_0(x_i)} < \log B \tag{6.5}$$

Figure 6.4 shows how the SPRT test can look like. As long as the value of the test is between $\log B$ and $\log A$ then no decision can be made and the test continues until the value of the test exceeds or drops below the thresholds and the test stops.

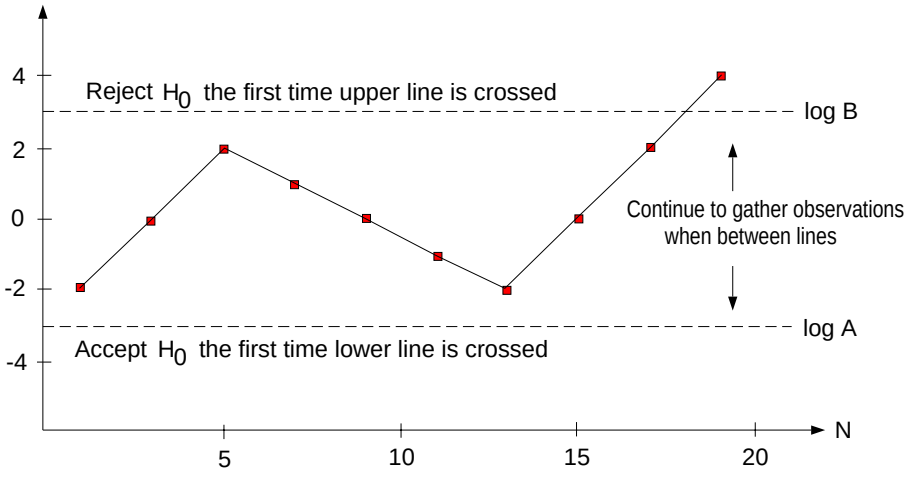

Figure 6.4. *Idealized schematic of the effect of SPRT.*

Here the difference between the estimated parameters and their nominal values is considered,

$$\triangle\theta_{i,k} = \hat{\theta}_{i,k} - \theta_{0,i}$$

where i is the i:th parameter, k is the k:th estimation and 0 means that it is the nominal value. Based on the last N_L estimates, a test on one of the following hypotheses is made:

$$H_0: \quad \theta_{i,\kappa} = \theta_{0,i} \quad \kappa = k - N_L, \dots, k$$

$$H_1: \quad \begin{cases} \theta_{i,\kappa} = \theta_{0,i} & \kappa = k - N_L, \dots, r-1 \\ \theta_{i,\kappa} \neq \theta_{0,i} & \kappa = r, \dots, k \end{cases}$$

The hypothesis H_0 states that all the N_L estimated values of parameter i are the same as the nominal value of that parameter, and the hypothesis H_1 states that some of the N_L estimated values one parameter are not the same as the nominal value. To obtain the fault signature we test the hypothesis for increase and decrease in the parameters seperatly.

In this thesis are the density functions Gaussian and we let the no-fault hypothesis H_0 have the mean value zero and a variance σ^2, and the fault hypothesis H_1 have the mean value h_{min} and the same variance σ^2. Then for k observations the density functions becomes

$$\begin{aligned} f_1 &= e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{k}(\triangle\theta(i)-h_{min})^2} \\ f_0 &= e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{k}\triangle\theta(i)^2} \end{aligned} \tag{6.6}$$

The probability ratio will then take the form

$$\begin{aligned} \Lambda(k) &= \frac{f_1}{f_0} = \frac{e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{k}(\triangle\theta(i)-h_{min})^2}}{e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{k}\triangle\theta(i)^2}} = \\ &= e^{-\frac{h_{min}}{2\sigma^2}\sum_{i=1}^{k}(h_{min}-2\triangle\theta(i))} \end{aligned} \tag{6.7}$$

Taking the logarithm of $\Lambda(k)$ in (6.7) and transforming it to a recurrent test will give

$$\begin{aligned} \ln\Lambda(k) &= -\frac{h_{min}}{2\sigma^2}\sum_{i=1}^{k}(h_{min}-2\triangle\theta(i)) = \\ &= \frac{h_{min}}{\sigma^2}\sum_{i=1}^{k}\left(\triangle\theta(i)-\frac{h_{min}}{2}\right) \end{aligned} \tag{6.8}$$

The following recurrent test can then be formulated:

$$\begin{aligned} \ln\Lambda^{+}_{\triangle\theta_i}(k) &= \ln\Lambda^{+}_{\triangle\theta_i}(k-1) + \frac{|h_{i,min}|}{\sigma_i^2}\left(\triangle\theta_i(k) - \frac{|h_{i,min}|}{2}\right) \\ \ln\Lambda^{-}_{\triangle\theta_i}(k) &= \ln\Lambda^{-}_{\triangle\theta_i}(k-1) - \frac{|h_{i,min}|}{\sigma_i^2}\left(\triangle\theta_i(k) + \frac{|h_{i,min}|}{2}\right) \end{aligned}$$

where $h_{i,min}$ is the parameter change which states the minimum value that has to be detected for the i:th parameter and σ_i is the standard deviation of the i:th parameter. We check the magnitude of both $\ln\Lambda^{+}_{\triangle\theta_i}(k)$ and $\ln\Lambda^{-}_{\triangle\theta_i}(k)$ by comparing them to two limits, A and B, who express the risk of an undetected fault and of a false alarm respectively. Then the decisions will be the following

$$\begin{array}{ll} |\ln\Lambda^{\pm}_{\triangle\theta_i}(k)| > \mathrm{A} & \text{the decision is made in favour of } H_1 \\ |\ln\Lambda^{\pm}_{\triangle\theta_i}(k)| < \mathrm{B} & \text{the decision is made in favour of } H_0 \\ \mathrm{B} \leq |\ln\Lambda^{\pm}_{\triangle\theta_i}(k)| \leq \mathrm{A} & \text{no decision can be made} \end{array} \tag{6.9}$$

The fault signature is then given by the decisions from each parameter where each decision can be one of the three types. Either in favour of H_0 or in favour of H_1 or neither. If we call the states

$$\begin{aligned} H_1 &: \text{ The decision is made in favour of } H_1 \\ H_0 &: \text{ The decision is made in favour of } H_0 \\ X &: \text{ No decision can be made} \end{aligned}$$

then the fault signatures could look like the example in Table 6.1. The fault signature in the table are grouped in three pairs which signifies three parameters where the pairs signifies the states that $\ln \Lambda^{+}_{\triangle\theta_i}$ and $\ln \Lambda^{-}_{\triangle\theta_i}$ have.

Fault in:	**Fault signature**
M	H_1H_0 H_1H_1 H_0H_1
l	H_0H_0 H_1X H_1H_1

Table 6.1. *Example of table of faults and fault sigantures.*

The fault isolation phase is divided into two steps. In the first step we compare the obtained fault signature to the reference or nominal signature of each possible fault. These nominal fault signatures can be obtained through simulation or experimentation. If the fault signature does not allow this localisation, because some faults have the same nominal fault signature, the second step is used. In the second step we use parameter variation functions where only the remaining hypothesis have to be checked. The principle is that the variations of the model parameters θ_i can be expressed as a function of the faults p_j,

$$\theta_i = \theta_{0,i} + f_{i,j}\left(\triangle p_j\right)$$

where i means the i:th parameter and j means the j:th type of fault. The relations between the faults and the discrete-time parameters are often too complex or not known. Therefore, by means of simulation or experimentation we approximate them with interpolation polynomials $\hat{f}_{i,j}\left(\triangle p_j\right)$. In general, the functions of faults are not symmetric for a positive or negative value of a parameter p_j, i.e

$$f_{i,j}\left(\triangle p_j\right) \neq f_{i,j}\left(-\triangle p_j\right)$$

That means the approximation polynomials $\hat{f}^{+}_{i,j}\left(\triangle p_j\right)$, for an increasing p_j, and $\hat{f}^{-}_{i,j}\left(\triangle p_j\right)$ for a decreasing p_j, are not identical.

Example 6.2. The parameter variation functions for two parameters of the denominator with a change of the pendelum mass m as a fault ($-0.5m \leq \triangle m \leq 0.5m$) could look like this:

$$\begin{aligned} \hat{f}^{+}_{a1,m} &= -0.0073\triangle m + 0.0262\triangle m^2 \\ \hat{f}^{-}_{a1,m} &= 0.0013\triangle m + 0.1400\triangle m^2 \end{aligned}$$

$$\begin{aligned} \hat{f}^{+}_{a2,m} &= 0.0456\triangle m - 0.0612\triangle m^2 \\ \hat{f}^{-}_{a2,m} &= -0.0384\triangle m - 0.3324\triangle m^2 \end{aligned}$$

For the following fault localisation procedure based on parameter variation functions, it is assumed that only one fault occur at a time. If changes in the model parameters $\triangle\theta_i$ are detected, and using the parameter variation functions, we estimate all possible fault magnitudes $\hat{p}_{j,\theta_i}$ for each considered model parameter. For the fault that has occurred, the interpolation polynomials belonging to this fault give approximately the same estimation value while the hypothesis for the other faults will give fault magnitudes that are quite different.

Example 6.3. Here we have two faults, p_1 and p_2, where each fault has three interpolation polynomials due to that it's three parameters(θ) which are estimated. Assuming fault p_1 has occurred, we see in Figure 6.5 that all interpolation polynomials give approximately the same estimation value $\hat{p}_1$. When we check the hypothesis for the second

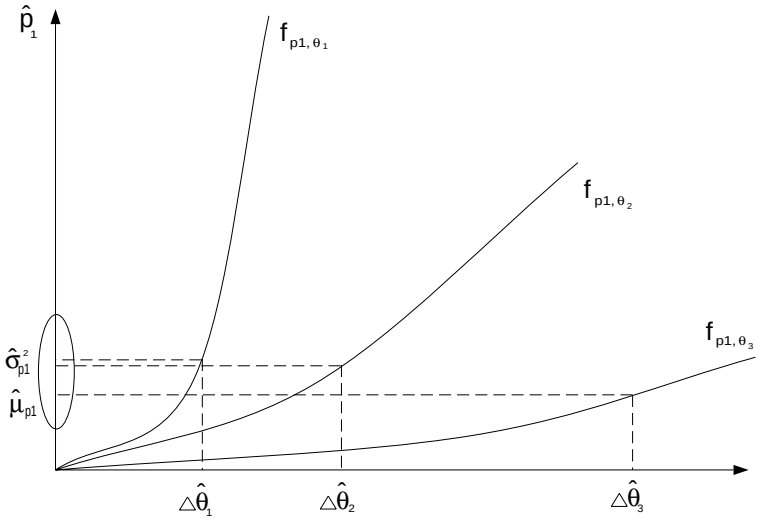

Figure 6.5. *Fault localisation. Fault p_1 has occured, checking the hypothesis of fault p_1.*

fault, the estimated fault magnitudes $\hat{p}_j$ will generally all be quite different from each other, because occuring faults do not belong to this set of interpolation polynomials, see Figure 6.6. Therefore, the variance of $\hat{p}_2$ will be greater than in the first case.

For the pendelum the fault polynomials for the two faults are shown in Figures 6.7 and 6.8. The x-axis corresponds to the deviation from the parameters nominal values while the y-axis corresponds to the fault. The fault polynomials are determined by first deriving the model when it is applied for a known fault. From the model the denominator is studied. It give three parameters which values changes depending on the magnitude of a fault. In Appendix A is the MATLAB code to generate the polynomials. If the obtained fault signature is not known, a change in the system behaviour is detected, but fault localisation is not. There are two possible situations for unknown fault signatures. The first case is when the occurring fault is not known or its reference signature was not introduced. The second one is due to a fault magnitude which is too small. When the fault magnitude is too small, a possible situation is that changes may only be detected in a few parameters.

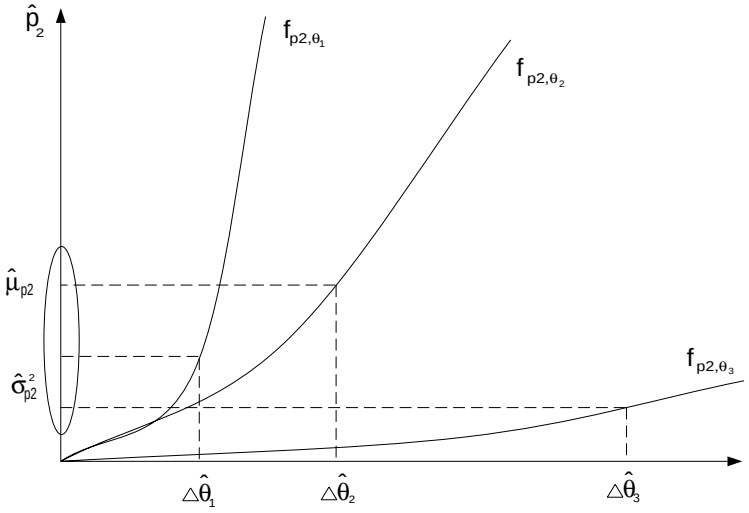

Figure 6.6. *Fault localisation. Fault p_1 has occured, checking the hypothesis of fault p_2.*

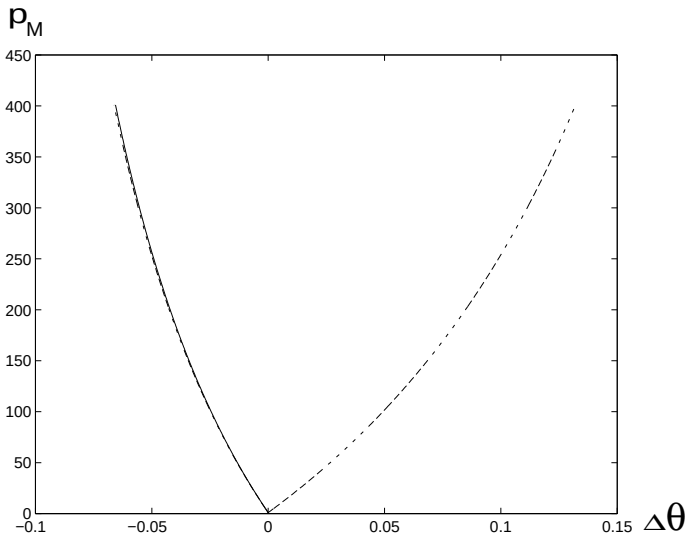

Figure 6.7. *Fault polynomials for the parameters for a fault in M_c.*

6.2 Simulated results

Starting with trying to create a good input signal, the position's reference was changed by hand. Studying the signal's spectrum in Figure 6.9, it shows that most of the signal's energy lies in the lowest frequency interval. That is what we know how the signal should be due to the bode plot in Figure 6.3. Deriving a model from the "handmade" signal and comparing the pole-zero plot of that model to the mathematical model's pole-zero plot shows that at the righthand side, near 1, the two plots are fairly alike. Since the sampling rate is high and the faster sample rate a system has the closer to 1 on the real axis the poles will lie, that area is the most important. Figure 6.10 and Figure 6.11 show the two zero-pole plots.

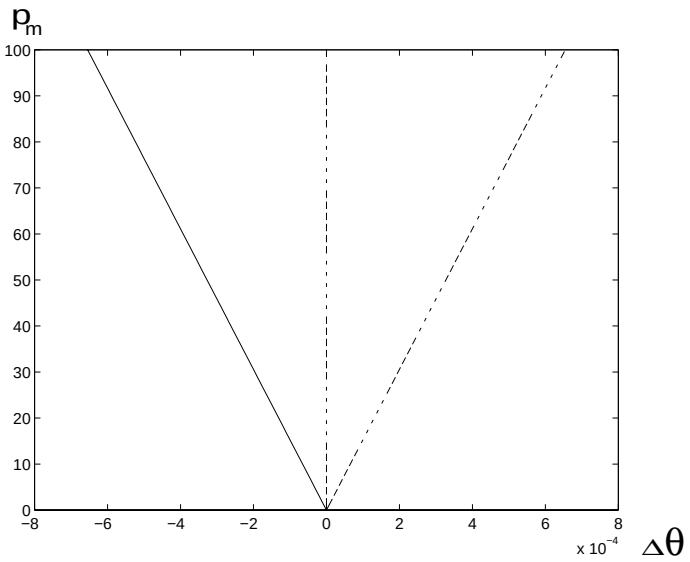

Figure 6.8. *Fault polynomials for the parameters for a fault in m_p.*

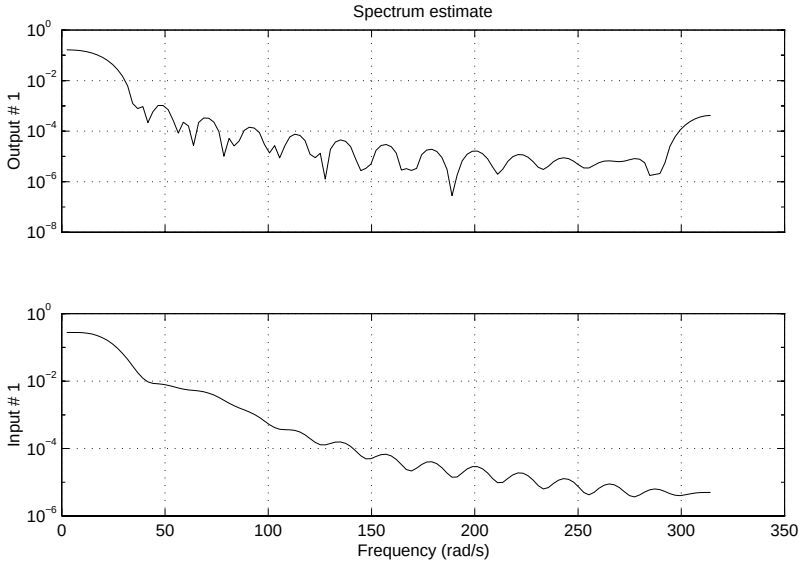

Figure 6.9. *Spectrum of the "handmade" signal.*

Instead of changing the position reference by hand bandlimited white noise was filtered through a fourth order butterworthfilter. The filter had a cutoff frequency of 4 Hz. Studying the spectrum for the signals in Figure 6.12, we find that they are not good in the low frequency interval. If instead as in [6] the reference signal is used as input signal and the output signal is amplified, the spectrum in Figure 6.13 becomes similar to the "handmade" signal. Looking at the pole-zero plot of the new model from the signals in Figure 6.14 shows a plot that is different than the earlier. Instead of having all

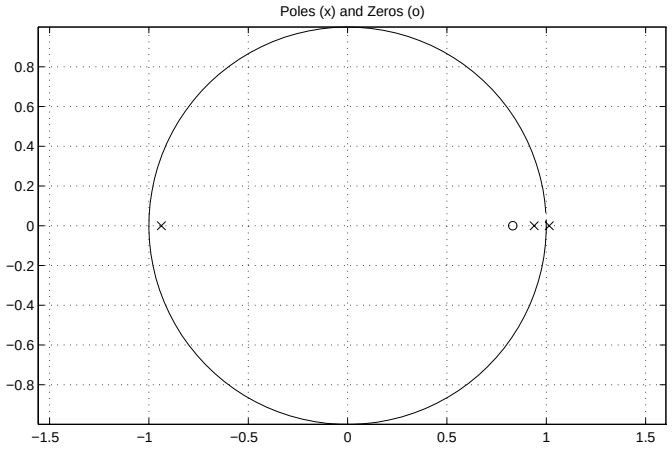

Figure 6.10. *Zero-pole plot of model identified with the "handmade" signal.*

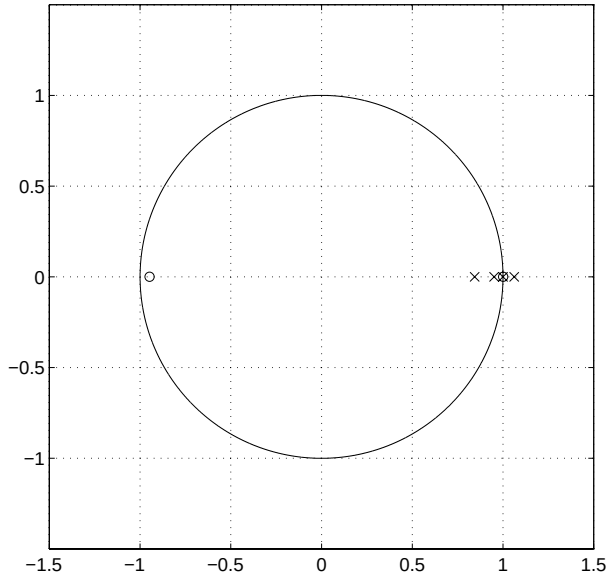

Figure 6.11. *Zero-pole plot of model derived in Chapter 3.*

the poles and zeros on the real axis, two poles are complexconjugated. Since the whole system with both the pendelum and the controller are identified it should give a plot that is different compared to the mathematical model.

The parameters were then determined from the signals using an ARX-model with its parameters set to three poles, two zeros and one time delay. Three simulations were done, each with a different seed to the bandlimited white noise, and the θ vector

$$\theta = [a_1 \quad a_2 \quad a_3 \quad b_0 \quad b_1 \quad b_2]^T \tag{6.10}$$

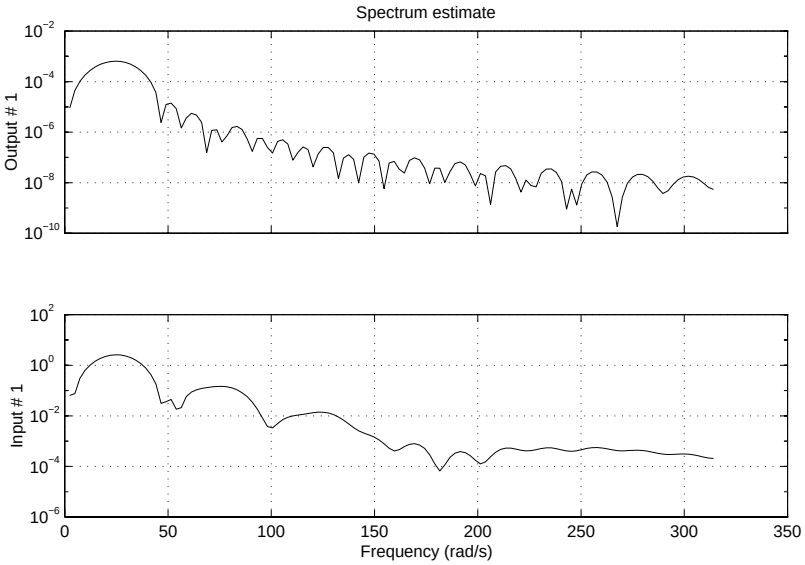

Figure 6.12. *Spectrum for input signal u(t) and output signal y(t).*

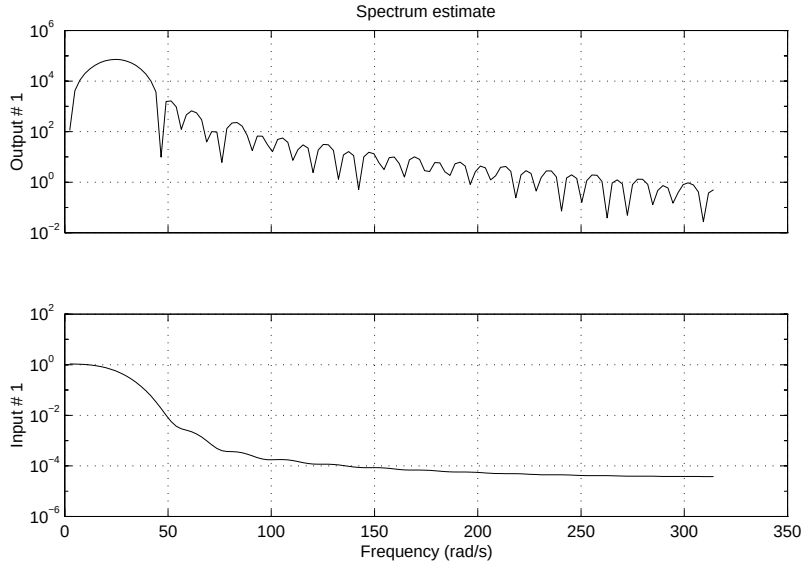

Figure 6.13. *Spectrum for input signal w(t) and output signal y(t).*

was determined. The result in the three cases when no faults been applied became

$$\begin{aligned} H_1(q) &= \frac{b_0 + b_1 q^{-1} + b_2 q^{-2}}{1.0000 - 2.5002 q^{-1} + 2.0975 q^{-2} - 0.5711 q^{-3}} \\ H_2(q) &= \frac{b_0 + b_1 q^{-1} + b_2 q^{-2}}{1.0000 - 2.5001 q^{-1} + 2.0973 q^{-2} - 0.5748 q^{-3}} \\ H_3(q) &= \frac{b_0 + b_1 q^{-1} + b_2 q^{-2}}{1.0000 - 2.5181 q^{-1} + 2.1294 q^{-2} - 0.5802 q^{-3}} \end{aligned} \tag{6.11}$$

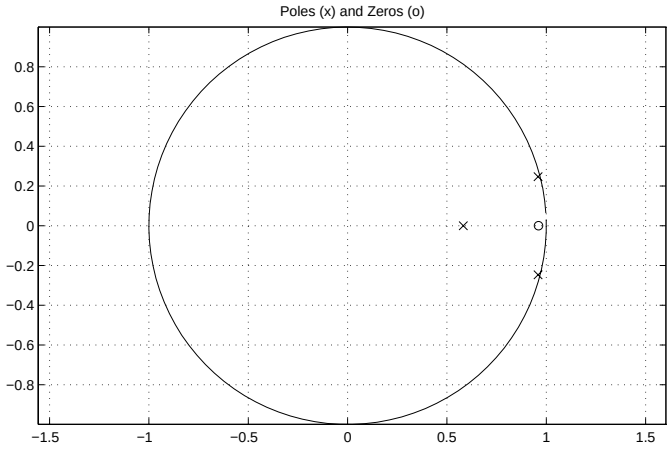

Figure 6.14. *Pole-zero plot of system with the pendelum and the controller.*

where

$$H(q) \quad = \quad \frac{b_0 + b_1 q^{-1} + b_2 q^{-2}}{1.0000 + a_1 q^{-1} + a_2 q^{-2} + a_3 q^{-3}} \tag{6.12}$$

The zeros b_i are here disregarded since their standard deviation are larger than the actual values and because the poles have a larger influence over the system than the zeros. The nominal values are determined by taking the mean value from the three estimations of each parameter. To derive the thresholds A and B the SPRT test was done on the three simulations ($N_L = 3$) of a fault free case and of a case with a fault. The results became the ones in Table 6.2. Studying the results from the test the thresholds

Fault		θ_1	θ_2	θ_3
Fault free	$\ln \Lambda^+_{\triangle\theta_i}$	0.00004	0.00001	0.000006
	$\ln \Lambda^-_{\triangle\theta_i}$	0.00003	0.000002	0.00003
$m_p = 230$ gr	$\ln \Lambda^+_{\triangle\theta_i}$	0.000017	0.00001	0.0008
	$\ln \Lambda^-_{\triangle\theta_i}$	0.000017	0.000002	0.0008
$m_p = 310$ gr	$\ln \Lambda^+_{\triangle\theta_i}$	0.0007	0.0005	0.0016
	$\ln \Lambda^-_{\triangle\theta_i}$	0.0007	0.0005	0.0015
$M_c = 555$ gr	$\ln \Lambda^+_{\triangle\theta_i}$	0.0161	0.0096	0.0132
	$\ln \Lambda^-_{\triangle\theta_i}$	0.0161	0.0096	0.0132
$M_c = 855$ gr	$\ln \Lambda^+_{\triangle\theta_i}$	0.0450	0.0270	0.0357
	$\ln \Lambda^-_{\triangle\theta_i}$	0.0450	0.0270	0.0357

Table 6.2. *Results from SPRT.*

were determined ad hoc which gave

$$\begin{aligned} \triangle\theta_1 = \hat{a}_1 - a_{1,0}: &\quad 0 < B = 0.0001 < A = 0.001 \\ \triangle\theta_2 = \hat{a}_2 - a_{2,0}: &\quad 0 < B = 0.0001 < A = 0.001 \\ \triangle\theta_3 = \hat{a}_3 - a_{3,0}: &\quad 0 < B = 0.001 < A = 0.01 \end{aligned} \tag{6.13}$$

Simulations with different faults in the two masses, M_c and m_p, were then done to get the fault signatures. The fault signatures is what the three parameters have for state from (6.9). Using the same terminology as in the previous section the fault signatures then becomes the ones in Table 6.3. Since it is two different fault signatures there is no

Fault in:	**Fault signature**
Fault free	H_0H_0 H_0H_0 H_0H_0
M_c	H_1H_1 H_1H_1 H_1H_1
m_p	XX XX H_1H_1

Table 6.3. *Table of faults and fault signatures from simulations.*

need to use parameter variation functions in the fault isolation part. A problem though is when m_p grows. Then in this case the X:s in the fault signature change to H_1 making the fault signatures to be alike. This is, for example, the case when $m_p = 1000$ grams but then such large faults cannot be applied without the process breaking down.

6.3 Experimental results

The experiments were done by logging the signals, w_1, and y from Figure 6.2, when different masses were added to the cart and to the rod. Three different measurements for each mass were gathered to work with. For the cart, starting at the original mass of the cart, 0.455 kg, two masses of 200 gram each were added, giving a mass to the cart of 0.455 kg, 0.655 kg and 0.855 kg. On the rod, masses of 20 grams were added to the original mass of the rod 0.210 kg giving a mass to the rod of 0.210 kg, 0.230 kg and 0.270 kg. The masses were applied using rubber band.

The thresholds in (6.13) from the previous section are here used to determine which fault that has occured. The SPRT test is used on the logged signals and they give the results in Table 6.4. As we can see the fault signatures are different depending on the

Fault in:		**Fault signature**
Fault free		H_0H_0 H_0H_0 H_0H_0
M_c:	0.655 kg	H_1H_1 H_1H_1 H_1H_1
	0.855 kg	H_1H_1 H_1H_1 XX
m_p:	0.230 kg	H_1H_1 H_1H_1 H_1H_1
	0.270 kg	H_1H_1 H_1H_1 H_1H_1

Table 6.4. *Table of faults and fault signatures from experimental signals using thresholds derived from simulations.*

magnitude of the fault, i.e fault isolation is not possible in this situation.

Since fault signatures could be obtained through simulation or experimentation, we instead of using the thresholds derived from simulations, derive thresholds from experiments in a fault free case. The thresholds change to

$$\begin{aligned} 0 < B = 0.0001 < A = 0.01 \\ 0 < B = 0.0001 < A = 0.01 \\ 0 < B = 0.001 < A = 0.01 \end{aligned} \tag{6.14}$$

where the threshold A gets a larger value which is due to more disturbed signals. The fault signatures, using the SPRT test, then becomes as in Table 6.5. Again the fault

Fault in:		Fault signature
Fault free		H_0H_0 H_0H_0 H_0H_0
M_c:	0.655 kg	H_1H_1 XX H_1H_1
	0.855 kg	H_1H_1 XX H_0H_0
m_p:	0.230 kg	XX H_1H_1 H_1H_1
	0.270 kg	H_1H_1 H_1H_1 H_1H_1

Table 6.5. *Table of faults and fault signatures from experimental signals using thresholds derived from experiment.*

signatures differs depending on the magnitude of the fault and fault isolation is not accomplished. The difference between Table 6.3 and the Tables 6.4 and 6.5 can be explained by the model fault that exists while the difference between Table 6.4 and Table 6.5 can be explained by the change in thresholds.

A solution in order to have fault detection could be to increase N_L and use more estimations. Another solution is to only look at two parameters or one parameter which is possible in this case. If we use the two most stabile parameters, i.e we don't use the two rightmost symbol in the fault signature, and use the thresholds derived from experiment then the fault signatures would become as in Table 6.6. Now the fault signatures still

Fault in:		Fault signature
Fault free		H_0H_0 H_0H_0
M_c:	0.655 kg	H_1H_1 XX
	0.855 kg	H_1H_1 XX
m_p:	0.230 kg	XX H_1H_1
	0.270 kg	H_1H_1 H_1H_1

Table 6.6. *Table of faults and fault signatures from experimental signals using thresholds derived from experiment and using two parameters for the signatures.*

differs but $m_p = 0.230$ kg is a small fault. That means that fault detection is possible, and fault isolation is possible if the fault is big enough. Since the two signatures are different from eachother when faults are large it makes fault isolation possible.

6.4 Problems

The problems that occured had really not anything to do with the diagnosis method itself. Instead they came during the identification of the system's parameters. At first, attempts were made to only identify the pendelum, but a satisfying input signal couldn't be generated. The filter that the bandlimited white noise went through was changed from a butterworthfilter to a FIR-filter, but that didn't help. Another attempt was to shift between two different controllers as described in Section 6.1.2, but that didn't succeed either. Instead the whole system with the pendelum and the controller was identified.

At the experiments the major problem was to apply the different masses on the cart and on the rod. The final solution, to use rubber band went though very well.

6.5 Comparison with articles

When comparing with articles [6, 7], the first observation is that in the articles it is another type of inverted pendelum. On the pendelum in the articles it is possible to directly measure three of the systems four states (position of the cart, velocity of the cart and the angle of the rod). The fourth state (anglular velocity) has then been estimated by an observer in order to compensate the effects of non-linear friction. In this thesis the effects of the friction have been assumed to be negligable. The difference between the systems with its controllers are that in this thesis there is no observer to estimate unmeasurable variables. Instead the two unmeasured variables (velocity of the cart and angular velocity) are here estimated with differential blocks in SIMULINK.

Also the ARX model is different from that used in [6]. It appears that a model with four poles and four zeros have been used there. When using the ARX model here the LS method is used. In the articles have instead the RLS method been used.

Comparing the results with [6] it is found that the results thats been achieved are similar. In the article a discrete identification has been made, a component fault has been applied to the pendelum and only the fault signature has been studied which are what's been done here too.

Comparing the results with [7] we find in that article that both sensor faults and component faults are investigated. For sensor faults the method in [8] is used and for component faults the method in [6] is used. Three different kinds of component faults could be detected and isolated compared to the two faults that could be detected and isolated here. It is a fault in the rod's length that can't be changed in this case. The faults that occured both in the article and here are slowly occuring faults because the process has to become excited.

6.6 Conclusions

The conclusions of this method applied to this process would be that

- The friction may not be neglected. If it is not neglected then the model will be enhanced and a better result could be produced.
- It is difficult to validate the parameters. If stabile parameters are derived, the poles aren't necessarly in the positions that they are expected to be.
- In discrete time, large changes in components can give small changes in the model. If, for example, you change a mass with a large value, the change will still be large in the poles of a continous time. But transforming the model to discrete time will make the poles to move towards eachother and move towards 1 on the real axis. How much is depending on the sample rate.
- It is difficult to make an identification in a closed loop.

- Identifying both the controller and the pendelum is easier than to just identify the pendelum. There is then no problems what the feedback loop can cause to the identification.

Chapter 7

Conclusions and extensions

7.1 Conclusions

In this report two model-based diagnosis schemes were studied. Model-based diagnosis is a procedure that uses a mathematical model of the process to be diagnosed.

The mathematical model used in this report is a four state nonlinear and unstable model that is linearized by assuming no friction, the rod is rigid and small angles. To control the process a controller is used which is derived from using LQ-method.

After studing articles of FDI methods applied on the inverted pendelum, two methods were chosen for analysing and implementing them on the inverted pendelum. The two methods were

- Parity equations method
- Parameter estimation method

Both methods utilizes analytical redundancy. Parity equation method uses an open loop strategy. It succeeded in detecting and locating sensor faults on the process but not to detect and locate actuator and component faults on the process, see Table 7.1. The derived residuals were detectable and not strongly detectable. During simulations the residuals didn't worked as was expected from the coding set which was surprising. The mathematical model was for this method not accurate enough.

	Sensor fault	**Actuator fault**	**Component fault**
Theory	Worked	Worked	Didn't worked
Simulations	Worked	Worked	Didn't worked
Experimental	Worked	Didn't worked	Didn't worked

Table 7.1. *Table of how the parity equation method worked.*

Parameter estimation were only used to detect and locate component faults, which it succeeded with when not all estimated parameters were used in the fault signature. To detect a fault when using this diagnosis method a statistical method is used, here it was the Sequential Probability Ratio Test (SPRT).

Trying to compare the two methods against eachother is difficult since the methods are best suited for different faults. Parity equations is though rather simple while parameter estimation could be difficult when identification must be performed in a closed loop.

7.2 Extensions

This report has shown how two different diagnosis method are implemented on an inverted pendelum and one is compared to articles that uses the same method. Some things though that can be improved are

- Improve the mathematical model.
- Investigate the controller.
- Residual evaluation.

Improve the mathematical model

If the mathematical model is improved both methods will become more accurate. One way of improving the model could be not to neglect the effect of the friction.

Investigate the controller

The controller that was used was not derived in this thesis, it was already given. Since the controller plays a part in the parameter estimation method and we don't know if the controller is good or bad it should at least be investigated. For example, two states (velocity of cart and angle) in the model were estimated with differential blocks in SIMULINK. Using differential blocks can give bad estimations. This can for example be avoided by using an observer instead.

Residual evaluation

The residual evaluation in the parity equation method are thresholds that has been derived ad hoc. It can be improved if the evaluation instead uses some statistical method as Generalized Likelihood Ratio (GLR) or approaches based on fuzzy logic.

References

[1] Erik Frisk. *Model-based fault diagnosis applied to a SI-engine.* Master's thesis, Reg nr: LiTH-ISY-EX-1679, Vehicular Systems, Linköping University, 1996.

[2] Marc van der Zon. *Analys av regleringen av en inverterad pendel.* Master's thesis, Reg nr: LiTH-ISY-EX-1658, Reglerteknik, Linköping University, 1995.

[3] Lennart Ljung. *System Identification: theory for the user.* Prentice-Hall, 1987, ISBN 0-13-881640-9.

[4] T. Glad and L. Ljung. *Modellbygge och simulering.* Studentlitteratur, 1991, ISBN 91-44-31871-5.

[5] Quanser Consulting. *INVERTED PENDELUM, Experiment and Solution.* Quanser Consulting, Canada, 1991

[6] Th. Sproesser, G. L. Gissinger, A. Kopf and K. Kroschel. *Fault detection in closed loop systems using identification methods applied on an inverted pendelum.* pages 973–978, Tooldiag 1993, Toulouse, France.

[7] B. Köppen-Seliger, Th. Sproesser, P. M. Frank and G. L. Gissinger. *Comparison of FDI-methods at the inverted pendelum.* In IFAC Fault Detection, Supervision and Safety for Technical Processes, pages 545–549, Espoo, Finland, 1994.

[8] Th. Sproesser, G. L. Gissinger. *Identification based Sensor Fault Detection and Localisation. Application to an Inverted Pendelum.* In IFAC Fault Detection, Supervision and Safety for Technical Processes, pages 555–560, Espoo, Finland, 1994.

[9] B. Köppen, P. M. Frank. *Application of Observer-Based Fault Detection Schemes to the Inverted Pendelum.* pages 979–986, Tooldiag 1993, Toulouse, France.

[10] H. Fortell. *Användarmanual till dSPACE.* 1996.

[11] W. W. Piegorsch, W. J. Padgett. *Notes on Sequential Analysis for a Course in Mathematical Statistics.* Technical Report No. 180, Department of Statistics, University of South Carolina, 62L10-4, February, 1995, http://www.stat.sc.edu/rsrch/techrep/.

Appendix A: Matlab script

MATLAB script used in simulations

```
s = 4;
[N, m] = size(y);
ytmp = y;

Y = zeros((s+1)*m, N-s);
for i=0:s
  Y((s-i)*m+1:((s-i)+1)*m,:) = ytmp(s+1-i:N-i,:)';
end;

[N, p] = size(u);
utmp = u;

U = zeros((s+1)*p, N-s);
for i=0:s
  U((s-i)*p+1:((s-i)+1)*p,:) = utmp(s+1-i:N-i,:)';
end;

[I,J] = butter(2,0.1);

r_x  = filter(I,J,(w_x'*(Y-Q*U)))';
r_fi = filter(I,J,(w_fi'*(Y-Q*U)))';
r_a  = filter(I,J,(w_a'*(Y-Q*U)))';
r_m  = filter(I,J,(w_m'*(Y-Q*U)))';
r_M  = filter(I,J,(w_M'*(Y-Q*U)))';

resT = t(s+1:length(t));
l = ones(997,1);

figure;
subplot(5,2,1),plot(resT,r_x,'-',resT,2.5e-3*l,'y--',resT,-2.5e-3*l,'y--'),
               grid, zoom on, ylabel('res_x');
subplot(5,2,2),plot(resT,r_fi,'-',resT,2e-4*l,'y--',resT,-2e-4*l,'y--'),
               grid, zoom on, ylabel('res_fi');
subplot(5,2,3),plot(resT,r_a,'-',resT,0.5e-3*l,'y--',resT,-0.5e-3*l,'y--'),
               grid, zoom on, ylabel('res_a');
subplot(5,2,4),plot(resT,r_m,'-',resT,0.02*l,'y--',resT,-0.02*l,'y--'),
               grid, zoom on, ylabel('res_m');
subplot(5,2,5),plot(resT,r_M,'-',resT,0.0005*l,'y--',resT,-0.0005*l,'y--'),
               grid, zoom on, ylabel('res_M');
subplot(5,2,6),plot( resT, u(s+1:length(u)) ),
               grid, zoom on, ylabel('In-u');
subplot(5,2,7),plot( resT, y(s+1:length(y),1) ),
```

```
              grid, zoom on, ylabel('Out-x'), xlabel('t');
subplot(5,2,8),plot( resT, y(s+1:length(y),2) ),
              grid, zoom on, ylabel('Out-fi'), xlabel('t');
subplot(5,2,9),plot( resT, f(s+1:length(f)) ),
              grid, zoom on, ylabel('fel'), xlabel('t');
orient tall;

clear ytmp utmp N p m i s l I J;
```

MATLAB code to generate residuals

```
 Q1=[D;C*G;C*F*G;C*F*F*G;C*F*F*F*G];
 Q2=[zeros(size(D));D;C*G;C*F*G;C*F*F*G];
 Q3=[zeros(size([D;C*G]));D;C*G;C*F*G];
 Q4=[zeros(size([D;C*G;C*F*G]));D;C*G];
 Q5=[zeros(size([D;C*G;C*F*G;C*F*F*G]));D];
 Q=[Q1,Q2,Q3,Q4,Q5];

 T1=[D;C*K(:,1);C*F*K(:,1);C*F*F*K(:,1);C*F*F*F*K(:,1)];
 T2=[zeros(size(D));D;C*K(:,1);C*F*K(:,1);C*F*F*K(:,1)];
 T3=[zeros(size([D;C*K(:,1)]));D;C*K(:,1);C*F*K(:,1)];
 T4=[zeros(size([D;C*K(:,1);C*F*K(:,1)]));D;C*K(:,1)];
 T5=[zeros(size([D;C*K(:,1);C*F*K(:,1);C*F*F*K(:,1)]));D];
 Tm=[T1,T2,T3,T4,T5];

 T1=[D;C*K(:,2);C*F*K(:,2);C*F*F*K(:,2);C*F*F*F*K(:,2)];
 T2=[zeros(size(D));D;C*K(:,2);C*F*K(:,2);C*F*F*K(:,2)];
 T3=[zeros(size([D;C*K(:,2)]));D;C*K(:,2);C*F*K(:,2)];
 T4=[zeros(size([D;C*K(:,2);C*F*K(:,2)]));D;C*K(:,2)];
 T5=[zeros(size([D;C*K(:,2);C*F*K(:,2);C*F*F*K(:,2)]));D];
 TM=[T1,T2,T3,T4,T5];

 R=[C;C*F;C*F*F;C*F*F*F;C*F*F*F*F];


 Z=R([1,3,5,7,9],:)';
 rank(Z)
 w_tmp=Z(:,[1,2,3,4])\(-Z(:,5))
 w_x=[w_tmp(1);0;w_tmp(2);0;w_tmp(3);0;w_tmp(4);0;1;0]
```

MATLAB code to generate fault polynomials

```
a1 = 1:1:400;
a2 = 1:1:400;
a3 = 1:1:400;
i = 1;
```

```
M = 0.455;
A = [0,1,0,0;0,-7.28/M,-m*g/M,0;0,0,0,1;0,7.28/(M*l),(M+m)*g/(M*l),0];
B = [0;1.72/M;0;-1.72/(M*l)];
[Ad,Bd] = c2d(A,B,0.01);
[num,den] = ss2tf(Ad,Bd,C(2,:),D(2,:));
poles = roots(den);
pp = [poles(1);poles(3);poles(4)];
p = poly(pp);
ff1 = p(2);
ff2 = p(3);
ff3 = p(4);
a1(i) = 0;
a2(i) = 0;
a3(i) = 0;
i = i+1;
for M = 0.455:0.001:0.855,
   A = [0,1,0,0;0,-7.28/M,-m*g/M,0;0,0,0,1;0,7.28/(M*l),(M+m)*g/(M*l),0];
   B = [0;1.72/M;0;-1.72/(M*l)];
   [Ad,Bd] = c2d(A,B,0.01);
   [num,den] = ss2tf(Ad,Bd,C(2,:),D(2,:));
   poles = roots(den);
   pp = [poles(1);poles(3);poles(4)];
   p = poly(pp);
   a1(i) = p(2)-ff1;
   a2(i) = p(3)-ff2;
   a3(i) = p(4)-ff3;
   i = i+1;
end
```

Appendix B: Simulink implementations

Controller

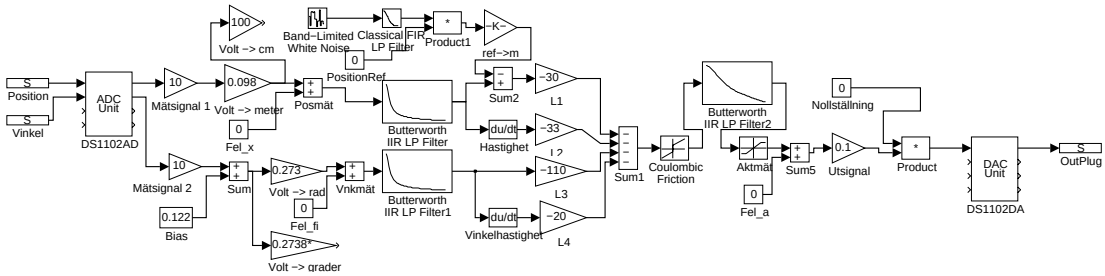

Figure B.1. *Controller used at experiments on the real process.*

Simulation of parity equations

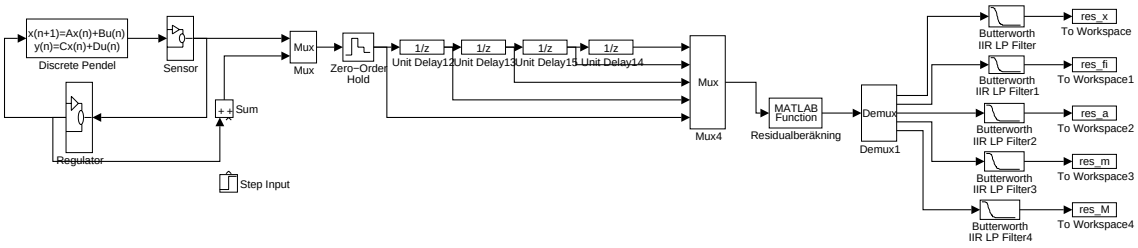

Figure B.2. SIMULINK *system to simulate parity equations.*

The sensor in the previous system

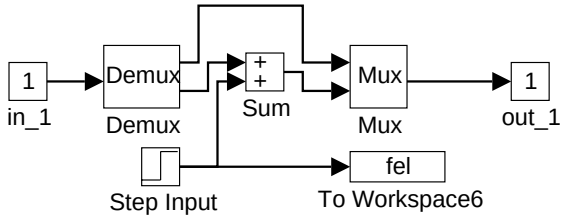

Figure B.3. *A sensor and how its faults are implemented.*