Linköping University | Department of Electrical Engineering

Master's thesis, 30 ECTS | Electrical Engineering

2021 | LiTH-ISY-EX--21/5415-SE

# Evaluation of model-based fault diagnosis combining physical insights and neural networks applied to an exhaust gas treatment system case study

**Björn Kleman**
**Henrik Lindgren**

Supervisor : Daniel Jung
Examiner : Erik Frisk

External supervisor : Håkan Warnquist

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

**Abstract**

Fault diagnosis can be used to early detect faults in a technical system, which means that workshop service can be planned before a component is fully degraded. Fault diagnosis helps with avoiding downtime, accidents and can be used to reduce emissions for certain applications. Traditionally, however, diagnosis systems have been designed using ad hoc methods and a lot of system knowledge. Model-based diagnosis is a systematic way of designing diagnosis systems that is modular and offers high performance. A model-based diagnosis system can be designed by making use of mathematical models that are otherwise used for simulation and control applications. A downside of model-based diagnosis is the modeling effort needed when no accurate models are available, which can take a large amount of time. This has motivated the use of data-driven diagnosis. Data-driven methods do not require as much system knowledge and modeling effort though they require large amounts of data and data from faults that can be hard to gather. Hybrid fault diagnosis methods combining models and training data can take advantage of both approaches decreasing the amount of time needed for modeling and does not require data from faults.

In this thesis work a combined data-driven and model-based fault diagnosis system has been developed and evaluated for the exhaust treatment system in a heavy-duty diesel engine truck. The diagnosis system combines physical insights and neural networks to detect and isolate faults for the exhaust treatment system. This diagnosis system is compared with another system developed during this thesis using only model-based methods. Experiments have been done by using data from a heavy-duty truck from Scania.

The results show the effectiveness of both methods in an industrial setting. It is shown how model-based approaches can be used to improve diagnostic performance. The hybrid method is showed to be an efficient way of developing a diagnosis system. Some downsides are highlighted such as the performance of the system developed using data-driven and model-based methods depending on the quality of the training data. Future work regarding the modularity and transferability of the hybrid method can be done for further evaluation.

# Acknowledgments

This thesis is has been a joint collaboration between Scania CV AB and the division of Vehicular Systems, Department of Electrical Engineering at Linköping University, during the spring of 2021.

We would like to extend our thanks to our supervisor from Scania, Håkan Warnquist, for his help and guidance throughout this thesis project. We would also like to thank Niclas Lindström and Kurre Källkvist for providing system knowledge and support. We want to thank Magnus Wadstrand and Fredrik Andersson for their help with the practical aspects of the thesis regarding measurements and diagnosis as well.

With special thanks to our supervisor from Linköping University, Daniel Jung, for his dedication and support during the project.

*Linköping, June 2021*
*Henrik Lindgren*
*Björn Kleman*

# Contents

# Abbreviations and Nomenclature

## Abbreviations

**CUSUM** Cumulative Sum

**DC** Duty Cycle

**DEF** Diesel Exhaust Fluid

**ECU** Electronic Control Unit

**FDT** Fault Diagnosis Toolbox

**FSM** Fault Signature Matrix

**MSE** Mean Squared Error

**MSO** Minimal Structurally Overdetermined

**NN** Neural Network

**PWM** Pulse Width Modulation

**ReLU** Rectified Linear Unit

**RNN** Recurrent Neural Network

**UDS** Urea Dosing System

## Nomenclature

| | |
|---|---|
| $\beta$ | Bulk modulus |
| $\epsilon_p$ | Pump displacement setting |
| $\eta$ | Fluid viscosity |
| $\eta_{lr}$ | Learning rate |
| $\eta_{volp}$ | Pump volumetric efficiency |
| $\hat{y}$ | Predicted value |
| $\nu$ | CUSUM tuning parameter |
| $\rho$ | Density |
| $A$ | Cross-sectional area |
| $C_q$ | Flow coefficient |
| $C_v$ | Pump laminar leakage loss |
| $D_p$ | Pump displacement |
| $e$ | Equation |

| | |
|---|---|
| $f$ | Fault |
| $J$ | Threshold |
| $L$ | Loss function |
| $n_p$ | Pump speed |
| $p$ | Pressure |
| $q$ | Fluid flow |
| $r$ | Residual |
| $T$ | Test quantity |
| $V$ | Volume |
| $w$ | Weight |
| $y$ | Measured value |

# Chapter 1

# Introduction

Fault diagnosis and system monitoring is an important part of the functionality of modern vehicles to assure reliability, avoid unexpected component failures and reduce environmental impact by tracking system degradation. Designing diagnosis systems requires the development of fault detectors to detect when a fault occurs in the system and isolate which component that is starting to fail. This Master's thesis project considers fault diagnosis of the exhaust treatment system in a heavy-duty diesel engine truck. The purpose of this project is to design and evaluate both a model-based diagnosis system and a hybrid diagnosis system using a combined data-driven and model-based method for diagnosability analysis, sensor placement, residual selection and generation. Since model-based and data-driven hybrid diagnosis is a promising method, Scania wants to compare its feasibility to the standard model-based diagnosis.

Model-based diagnosis is a fault diagnosis approach that detects abnormal system behavior by comparing sensor data and predictions using a physically-based model of the system. One advantage is that faults are isolated using model analysis which makes it possible to localize faults that have not been observed before. Model-based analysis can be done early during the system development phase which can give useful insights when designing the system.

The main disadvantage with model-based methods is that it requires a reliable and accurate model of the system, which can be a time-consuming task to produce [1]. To combat this issue, data-driven fault diagnosis methods can be used. General structure data-driven methods require training data that captures the system's behavior and fault modes. However, collecting enough representative training data from all relevant fault classes can be difficult, which will result in miss-classifications when classifying fault scenarios not represented in training data [2], [3].

Most heavy duty vehicles today are powered by internal combustion engines and they all release pollution out to the air. Legislation to reduce emissions are becoming stricter and the industry needs to keep up with the demands [4]. To reduce toxic emissions from internal combustion engines they are all required to use exhaust gas treatment systems. Diesel engines especially release large amounts of emissions if left untreated. This is partly because of their higher combustion temperatures. The exhaust gas treatment system for a diesel engine consists of multiple parts working together to reduce particles, hydrocarbons, CO and NOx gases. The exhaust gas treatment system of a modern Scania truck is shown in Figure 1.1 where the input of exhaust gases is in the pipe in the top left of the figure. The gases pass through multiple filters and catalysts following the arrow in grey until they end up in the bottom left exhaust pipe. The system looked at in this thesis is the Diesel Exhaust Fluid (DEF) injection part of the exhaust treatment system also seen in Figure 1.1. It includes sensors, a tank, hoses, a pump and the DEF dosage unit (Evaporator in the figure). This system is used to reduce

NOx gases created in the combustion process, where a failing component results in the wrong amount of DEF injected into the exhaust. The main focus of this thesis will be on how the pressure builds up in the system.

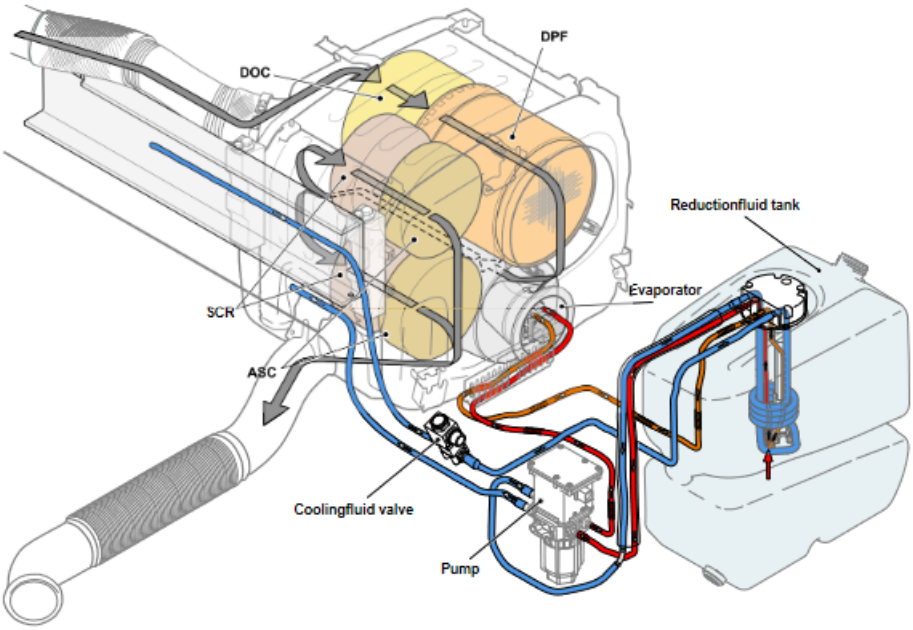



Figure 1.1: System overview of the Scania truck exhaust gas treatment system. *©Scania CV AB 2021, Reprinted with permission.*

Designing diagnosis systems is a non-trivial task and the complexity increases with the size and complexity of the system. There are several questions and properties that arise while constructing diagnosis systems that need to be addressed, such as:

- With the available set of sensors, in which components is it possible to detect faults and how accurately can the faults be isolated?
- Where are the best locations to add additional sensors to improve fault isolation?
- How to identify model candidates for designing fault detectors?
- How to handle and classify the root cause of unknown faults?

Fortunately, there are well-defined tools and methods available that can be used for designing diagnosis systems that can address these questions. In this thesis, structural methods [5] will be used for analysis with the help of the Fault Diagnostics Toolbox [6].

Developing high fidelity mathematical models for a set of components can be a time consuming process. In this work, standard mechanical models of the different system components will be used in the analysis and diagnosis system design. Then, a hybrid modeling approach, as presented in [7], combining causal structural model information and training data for residual generation will be used. This method generates residuals by using data-driven grey-box recurrent neural networks (RNN) where the structure of the neural network is based on a structural model of the system. A structural model describes the relationship between equations and variables in a system model without including analytical expressions. The mentioned advantages

of the proposed grey-box RNN approach is that it only requires data from the fault-free mode and no high fidelity model is needed. By using an RNN based on the physical insights of the system, the results in [7] indicate that it would be possible to detect the root cause of unknown faults.

The two methods, the model-based method and the hybrid method, evaluated in this thesis use different ways of generating residuals. The hybrid method utilizes neural networks with structural information and the model-based method uses a mathematical model of the system. The neural networks only use structural information while the model-based method uses analytical relations derived from physical insights.

Systematic methods can speed up the design process of diagnosis systems. Hybrid fault diagnosis methods combining models and training data can take advantage of both approaches. One aspect of this thesis is the comparison of the two different methods, where the main comparisons between the methods will be in the implementation, validation process and the ability to detect faults. There will also be modeling work and data collection performed in order to validate the system model and residuals.

## 1.1 Problem Formulation

The purpose of this thesis is to look at the design process of a diagnosis system, creating a hybrid diagnosis system for analysis and evaluation. As a baseline of how good the hybrid diagnosis system performs, another diagnosis system will be developed using only model-based techniques. The two methods will be compared both in terms of performance and the process of creating the diagnosis systems.

To accomplish the goals and satisfy the purpose addressed above, this thesis aims to address the following problem statement:

1. Use model-based diagnosis for analysis and design of a diagnosis system for the DEF injection subsystem and evaluate the results using real data from an experimental test bench and a heavy duty truck.
2. Use model-based and data-driven hybrid diagnosis for analysis and design of a diagnosis system for the DEF injection subsystem and evaluate the results using real data from an experimental test bench and a heavy duty truck.
3. Evaluate the performance and the development process of the hybrid method diagnosis system and compare to the standard model-based method.

## 1.2 Delimitations

The modeling process can take a large amount of time and effort since all physical systems can be modeled very meticulously. Since this Master's thesis project's focus is on the design process and evaluation of the diagnosis system, not just modeling, there needs to be a limit to how much time will be spent on the modeling process. In this project standard models from literature will be used to model the different components of the system.

## 1.3 Method

The implementation and evaluation work of the thesis was an iterative process where for each iteration the diagnosis systems and evaluation process is extended.

Before the implementation and evaluation process of the thesis, a literature pre-study was conducted to get familiar with the topic of the thesis. Several reports regarding model-based fault diagnosis, data-driven fault diagnosis, structural modeling and grey-box modeling are reviewed in Section 1.4. The main focus of the pre-study is to study and understand the report [7], which first proposed the method used in this thesis.

As well as doing a literature pre-study the first part of the thesis was used to understand the urea dosing system (UDS) by talking to experts at Scania. After the pre-study, data was collected for model validation and evaluation. The data was collected from a test bench and from a heavy-duty truck during fault-free and fault conditions. This data was then processed and used for modeling both physically-based and neural network models, and construction of the respective diagnosis systems. The last part of the process was the evaluation and comparison of both diagnosis methods. The system is modeled and validated for both methods using the collected data.

## 1.4 Related research

This thesis is based on the report [7], where the author provides theory and a case study on an internal combustion engine test bench, where a method of hybrid fault diagnosis to automatically generate residuals is used. The mentioned report's use of grey-box RNNs, enables the ability to combine physical insights from model-based diagnosis methods with data-driven models' ability to describe the system and classify different faults. The structure of the neural networks is determined using structural analysis to help facilitate the generation of residuals. It is shown how structural models can be an effective approach for designing grey-box RNN for residual generation. To improve classification, cumulative sum (CUSUM) tests can be used. The classification performance depends on the training data. The collected data must be representative of the system in fault free operation and it needs to capture different operating modes. The trained neural networks are sensitive to all faults in the case study, however, they are overall better at detecting faults that are strongly affecting the predicted sensor output.

A similar study was done in [8]. Isolation and localization of unknown faults using neural network-based residuals is performed on a simulated non-linear two tank system. It was assumed in the report that only a general model that describes the system is available. Using nominal training data and neural networks that are based on the system structure proves to be a feasible solution to reducing development time while still making use of the physical properties of the system.

Combining model-based and data-driven techniques for fault diagnosis has been tested and written about in several reports. In [3], a hybrid diagnosis method is developed and evaluated. The proposed method combines model-based fault isolation with Support Vector Data Description classifiers. In a case study, it is shown that the hybrid method can solve the problem of limited training data and it can improve fault isolation accuracy.

A demonstration of data-driven diagnosis conducted on an evaporator for a beet sugar factory using real data was performed in [9]. Structural information is used to design a grey-box model of the system using state space neural networks (ssNN). It is found that by using an ssNN they can obtain similar or even better results than a simulation model manually derived by an expert.

Using data-driven methods such as neural networks can be an effective way of modeling a system. In [10] the goal was to model the dynamics of a wind turbine using a neural network based model. Since not all states can be measured an observer was used together with neural network-based system identification. A residual-based fault detection algorithm was evaluated for simulated data. The identification and fault detection method was proven effective.

In the report [11], modeling of an industrial process is examined and compared using two methods, physically-based modeling and data-driven grey-box modeling using RNN:s. The report found that the physical model results in higher modeling and computational effort and requires more knowledge about the system. However, an advantage is that it can adapt to small changes and is very robust. The data-driven grey-box model however is not as robust and requires large amounts of data. The authors of the report argue that there are still major advantages to using data-driven grey-box modeling. It requires low modeling effort and is flexible for adaptations which can save a lot of development time.

Structural analysis is an efficient method that can be used when designing fault diagnosis systems and it will be used in this thesis. There are several resources for structural modeling and analysis. In [12] a model-based diagnosis algorithm for Polymer Electrolyte Membrane Fuel Cell (PEMFC) systems was developed and studied. The model was analyzed using structural analysis to identify how the physical variables correlated to each other. By using casual computation analysis, the maximum theoretical fault isolability that could be achieved with the minimal number of sensors was identified. It was shown that the algorithm could detect and isolate almost all faults using a minimal number of sensors.

In [13] and [14], methods to decide where to place sensors were presented. The methods are based on structural analysis and can be used to find the minimal number of sensors to achieve certain fault diagnosis requirements. Similar methods for diagnostic purposes were used in [15] to decide how to select a minimal sensor set for battery packs.

A useful tool for designing and analyzing advanced fault diagnostics is the Matlab toolbox called Fault Diagnosis Toolbox (FDT) [6]. The system in this thesis is not a large system, but the toolbox and the techniques used in it are scalable for large systems as well. The toolbox includes tools for modeling, fault diagnosability analysis, sensor selection, residual generator analysis, test selection, and code generation. All of which were used in this thesis.

The exhaust gas treatment system that will be used as a case study for this thesis to evaluate the diagnosis methods has previously been modeled in a thesis in cooperation with Scania. The proposed model has been validated with experimental data and will be used as a starting point for this thesis [16]. The model in this thesis work will not be the same, the older model included two dosage units and modeled flows as states through the Navier stokes equations. This thesis will not consider flows as states and there is only one dosage unit. The modeling of filters and pressures before the pump will also be looked at for increased diagnosis performance, which the older model does not consider.

## 1.5 Outline

The report is organized as follows:

- **Chapter 1: Introduction**
  This chapter includes the purpose, goal, problem formulation, related research, approach and delimitations.
- **Chapter 2: Model-based diagnosis**
  With this chapter, the purpose is to present and describe theory related to model-based diagnosis.
- **Chapter 3: Data-driven modeling**
  This chapter presents the theory needed to understand the data-driven methods used in the thesis work.
- **Chapter 4: System Description**
  Here the goal is to present and describe the exhaust gas treatment system, its function and how it is modeled.
- **Chapter 5: Diagnosis system design process**
  Chapter 5 describes the diagnosis system design process. Including the methods used for the modeling, residual generation, data collection, training, etc.
- **Chapter 6: Results**
  This chapter will present the results for the diagnosis system. It for instance includes figures for the fault detection performance and residual validation.
- **Chapter 7: Discussion**
  This chapter includes analysis and discussion of the results and method.
- **Chapter 8: Conclusion and Future Work**
  This final chapter will present the conclusions and recommendations for future development.

## 1.6 Work distribution

The work of this project was divided evenly between both thesis workers. Both authors contributed equal amounts to the design process and report writing. In the later stages of the work, Henrik Lindgren was more responsible for evaluating the model-based residuals and Björn Kleman evaluated the neural network-based residuals.

# Chapter 2

# Model-based diagnosis

This chapter introduces the necessary theory used in this thesis related to model-based diagnosis.

## 2.1 Model-based diagnosis

The meaning of diagnosis is to, from observations and knowledge of a system, decide if there is a fault present and also to identify where and what the fault is. Model-based diagnosis is a diagnosis technique utilizing models to diagnose systems by comparing observations of the systems' actual behavior with the behavior predicted from the model. The model will predict the fault-free behavior and if the model prediction is not consistent with sensor data ideally it implies that there is a fault present in the system. Comparing models of different parts of the system to different observations enables isolation of faults as well [1].

### 2.1.1 Residual generation for fault detection

A residual generator, $r_k(z)$, is defined as a function of sensor and actuator signals $z$, that is ideally equal to zero when no faults are present. The residual is said to be sensitive to a fault, $f_i$, if $f_i \neq 0$ implies that $r_k \neq 0$. If the residual is insensitive to a certain fault $f_i$, the fault is said to be decoupled from that particular residual [1].

A fault, $f_i$, is detectable if when $f_i \neq 0$ (which implies that $r_k(z) \neq 0$) the observations from that mode are distinguishable from observations made during nominal conditions. Detectability is a system property that can be interpreted as the ability to design a residual generator that is sensitive to the fault. Fault detection is complicated by sensor noise and model inaccuracies.

A residual generator using analytical redundancy, as illustrated in Figure 2.1, is constructed by using a model of the system to predict what the system would output given the same input signals as the system. The predicted value is then subtracted from the system output which results in the residual $r(t)$.

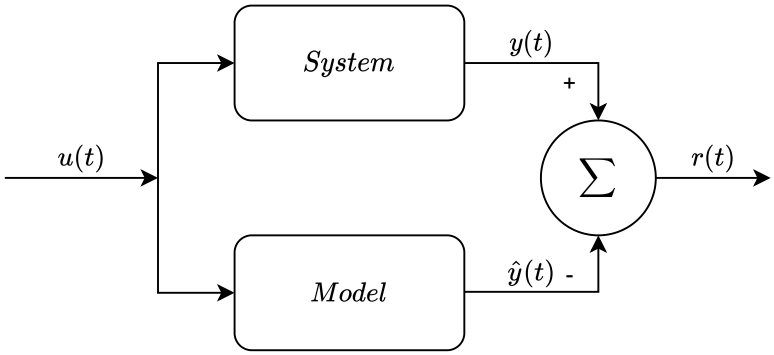



Figure 2.1: An example of an implementation of a residual generator. Where $u(t)$ is the control signal for the system, $y(t)$ is the measured values from the system, $\hat{y}(t)$ is the predicted value from the model and $r(t)$ is the residual

**Example 1** *An example of the signals in Figure 2.1 is shown in Figure 2.2. Where the control input is a unit step resulting in the step in both modeled and measured value. The measured values include white noise and $\hat{y}$ is the modeled value of $y$. The residual signal is then calculated by subtracting $\hat{y}$ from $y$. At 60 seconds a fault is implemented which results in the measured value of the system decreasing to 0.5 while the predicted value stays at 1. This difference in predicted and measured value results in an increase in the output of the residual, seen in the lower subplot at 60 seconds.*

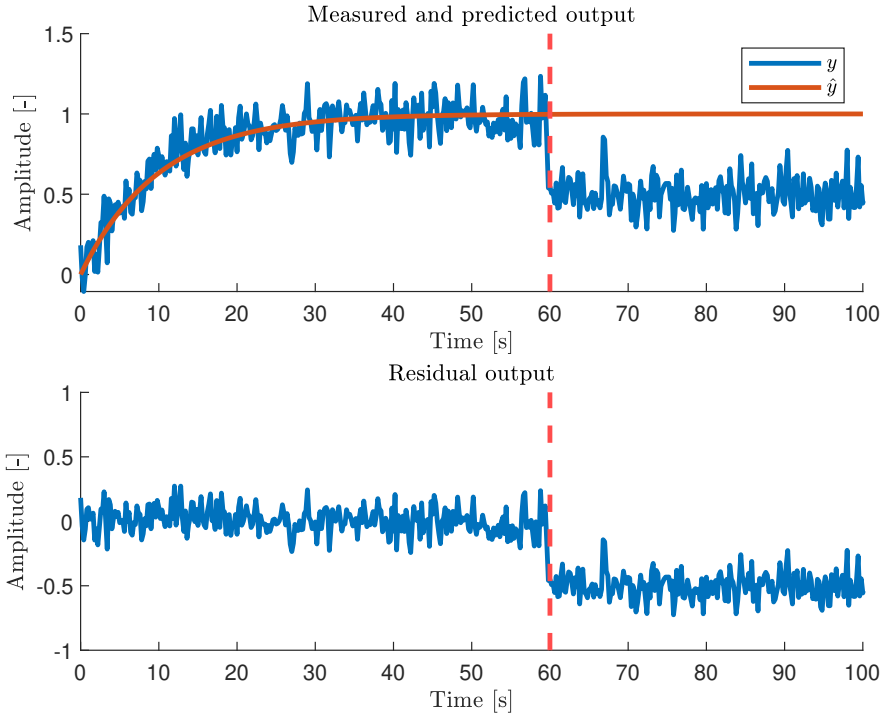



Figure 2.2: Example of measuring and predicting a systems output in the upper subplot where the orange line is the predicted value and the blue line is the measured value. In the lower subplot is the residual created by subtracting $\hat{y}$ from $y$. A red dashed line at 60 seconds in both plots indicates when a faults enters the system.

### 2.1.2 Fault Signature Matrix

Given a set of residuals and faults, a Fault Signature Matrix (FSM) can be used to illustrate the fault sensitivity of the system. The FSM is a matrix with different residuals in each row, $i$, and faults in each column, $j$. An X at $(i, j)$ means that $r_i$ is sensitive to $f_j$. Given a set of triggered residuals, a set of fault candidates can be determined [17]. The FSM is derived by analyzing which part of a model is used in each residual. The FSM only illustrates the best-case results, since it only uses structural information. See Table 2.1 for an example of a FSM.

Table 2.1: An example of a fault signature matrix with 4 faults and 4 residuals.

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ |
|---|---|---|---|---|
| $r_1$ | X | X | | |
| $r_2$ | X | | X | X |
| $r_3$ | | X | X | X |
| $r_4$ | | X | X | |

### 2.1.3 Isolability

An important property of a diagnosis system is not only detectability i.e. the ability to detect faults but the ability to isolate faults from each other. The isolability of a diagnostic system enables it to specify where in a system a fault occurs. The isolability of a fault is defined as: $f_i$ is isolable from $f_j$ if there exists a residual that is sensitive to $f_i$ but not to $f_j$ [1].

The isolability properties of a system can be visualized through an isolability matrix. It is a matrix with the fault modes on both axes. An $X$ on row $i$ and column $j$ means that the fault, $f_i$ is not isolable from $f_j$ [1]. This means that for full isolability, there will be a diagonal line of $X$:s. Using the same example as in Table 2.1, the resulting isolability is shown in Table 2.2.

Table 2.2: The isolability matrix for the example in Table 2.1

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ |
|---|---|---|---|---|
| $f_1$ | X | | | |
| $f_2$ | | X | | |
| $f_3$ | | | X | |
| $f_4$ | | | X | X |

The fault $f_3$ is not isolable from $f_4$ because there is no residual in Table 2.1 that is sensitive to $f_4$ without being sensitive to $f_3$.

### 2.1.4 Diagnostic tests based on residuals

Diagnostic tests can be created in several different ways but this thesis will be focused on two of them, thresholded residuals and CUSUM tests. Test quantities are model validity measures [1] and since the goal of creating a diagnosis system is to measure model validity test quantities can be used for this. For each diagnostic test $\delta_i$ the substatement $S_i$ is defined as

$$S_i = \begin{cases} S_i^1 & \text{if } |T_i(z(t))| \geq J \\ S_i^0 & \text{if } |T_i(z(t))| < J, \end{cases} \tag{2.1}$$

where $S_i(t)$ defines the behavior at time point $t$ based on the model and measurements $z(t)$. $S_i^1$ implies that a fault has been detected in a residual. The possible faults depend on what the

residual is sensitive to, and that can be seen in an FSM such as Table 2.1. Whereas $S_i^0$ implies that either there is no fault or the fault is to small to be detected from the residual. The parameter $J$ is the threshold for distinguishing between the cases $S_i^1$ and $S_i^0$. When selecting $J$ there is a trade-off between missed detection rate and false alarm rate. Lowering $J$ increases the detection rate of faults but is at the cost of increasing the risk of false alarms. If instead, $J$ is increased, false alarms will also decrease but it also increases the risk of missed detections.

$S_i^1$ implies faulty behavior and $S_i^0$ fault-free or fault is not yet detected. The test quantity $T_i(t)$ should be low if the data matches the model, and large otherwise [1]. Therefore the test quantity $T_i(t)$ can be seen as a measure of the validity of the models compared to the measurements.

An overview of a diagnosis test $\delta_i$ is shown in Figure 2.3, where the residual generator is a function of a model and measurement. In this thesis, the test quantity is either the residual by itself or CUSUM test because they are some of the simplest yet effective methods. Other variations exist but will not be discussed. A threshold $J_i$ is selected to satisfy the requirement on false alarms since residuals are usually noisy due to measurement noise and modeling errors so that $T_i(t) < J_i$ in the fault free case.

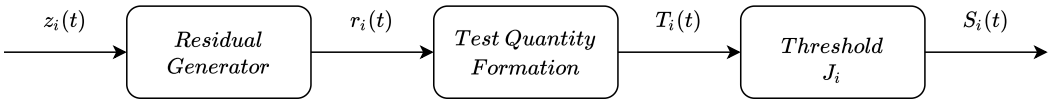



Figure 2.3: Diagnostic test structure.

#### 2.1.4.1 Residual filtering

Residuals may need to be filtered before a reliable decision can be made since the signal can be noisy. This avoids unnecessary false alarms. A simple low pass filter can be enough [1].

For a residual $r(t)$, the filtered signal can be described as

$$r_{filt}(t) = LP(r(t)), \tag{2.2}$$

where $LP()$ is the low-pass filter function applied on the residual $r(t)$ and the diagnostic test is still performed as in Figure 2.3 after filtering.

**Example 2** *An example of an implemented residual with a threshold can be seen in Figure 2.4. The residual includes some noise but is centered around zero in the nominal case. There is a threshold implemented at 0.3 for the non-filtered residual so that an alarm would be output, i.e. a fault would be detected if the residual crosses 0.3 and the residual does not cross in the nominal case. At 60 seconds a fault is simulated which results in the residual deviating from zero and crossing the threshold. Decreasing the threshold for the residual will increase the risk of false alarms i.e. the residual could cross the threshold in the fault free case.*

*To improve the detection rate and decrease the risk of false alarms the residual is filtered as seen in the lower plot of Figure 2.4. As shown in the figure the threshold can be reduced with the filtered residual.*

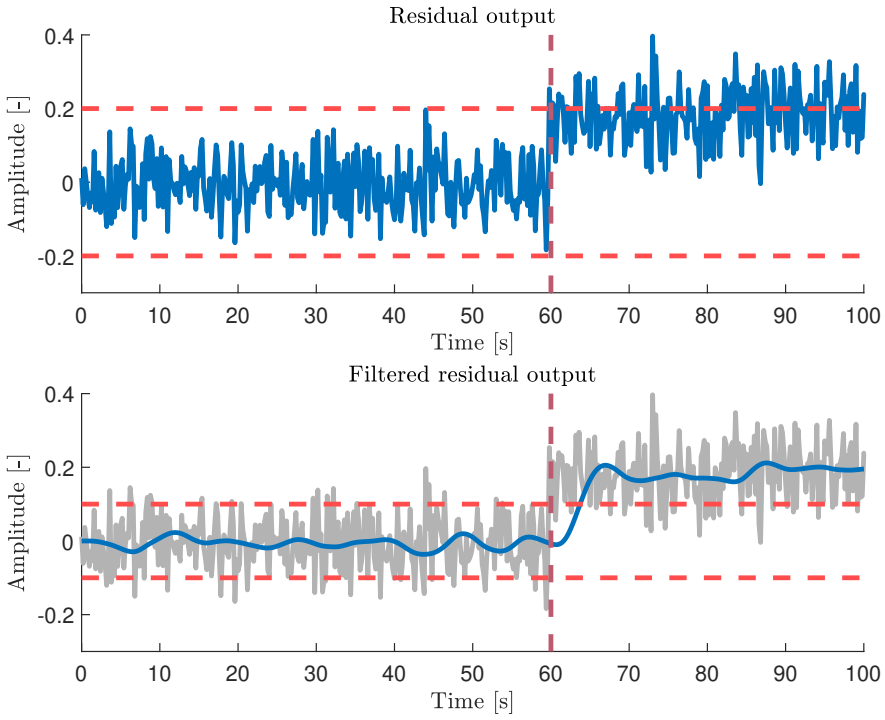



Figure 2.4: Example of a residual with a fault implemented at 60s marked by the dashed purple line. In the lower subplot the blue line is the filtered residual. In the upper subplot a threshold is implemented at ±0.2 marked by the dashed red line. While in the lower subplot the threshold can be reduced to ±0.1 without increasing the false alarms and better detection of the fault.

#### 2.1.4.2 CUSUM test quantity

The cumulative sum (CUSUM) algorithm is an alternative way to calculate a test quantity [1]. The benefit to this approach is that since it reacts to changes over time, it can detect small changes without causing a large false alarm rate if allowed a longer detection time. In the previously described case, the only way to detect smaller faults is by lowering $J_i$, which will cause false alarms due to sensor noise.

One example of how a CUSUM test can be implemented is

$$T_i(t) = \max(0, T_i(t-1) + |r(t)| - \nu), \quad T_i(0) = 0, \tag{2.3}$$

where the term $|r(t)|$ from (2.3) should be smaller than the tuning parameter, $\nu$, in the fault free case so that $T_i(t)$ is zero. When a fault occurs, $|r(t)|$ will exceed $\nu$ and then its impact on the residual output will be integrated over time by $T_i(t)$.

**Example 3** *An example of an implemented CUSUM test with its corresponding residual is shown in Figure 2.5. Using this CUSUM test the fault can be detected without increasing the risk of false alarms just as with the use of a filter. In this figure, the CUSUM test is used with $\nu$ set at 0.1. At 60 seconds the CUSUM function integrates the fault over time resulting in the detection of a change in the mean of the residual. Faults that are hard to detect with just a residual are now more easily detectable and the effect of the fault over time is shown.*

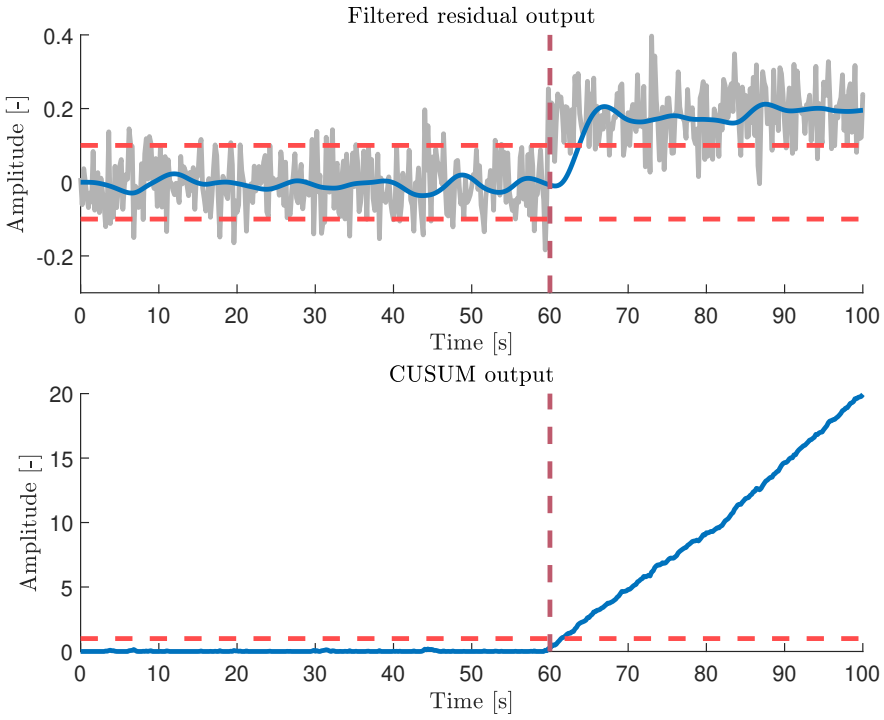



Figure 2.5: Example of a CUSUM test in the lower subplot implemented on the residual colored in grey in the upper subplot. The blue line is the residual in the upper plot and the CUSUM value in the lower plot, the two red dashed lines are $\pm\nu$ set at $\pm 0.1$. The time of the fault is again marked by the dashed purple line at 60s. A threshold for the CUSUM test is implemented at 1 marked by the dashed red line in the lower plot.

## 2.2 Fault diagnosis analysis of complex systems using structural methods

A structural model is a bi-partite graph describing the relationship between equations and variables in a model. Structural analysis uses structural models which do not need parameter values, which means that they can be used early in a design process. Structural methods are useful to see which faults are detectable and isolable when developing diagnosis systems. Analyzing large scale non-linear systems is a non-trivial task and structural methods can help make this task more efficient. A drawback is that it only describes the best case results since the analysis is based on structural information only [5].

The structural model contains no information about parameter values or analytical expressions and can be represented by an incidence matrix, where variables are divided into known, unknown and fault signals [5]. For each row, there is an equation, $e_i$, and for each column, there is a variable $x_j$. An $X$ in position $(i, j)$ means that the variable $x_j$ is included in the equation $e_i$ as illustrated in the following example.

**Example 4** *Consider the simple electrical circuit in Figure 2.6. It consists of a voltage source, resistor and a capacitor.*

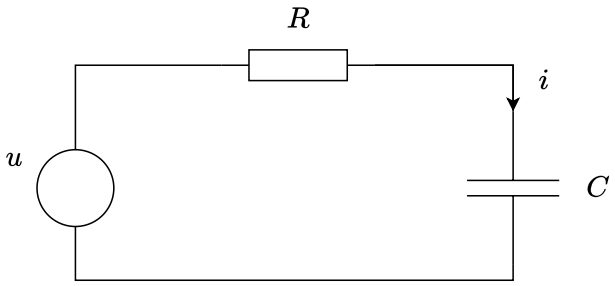



Figure 2.6: A simple electrical circuit.

*Consider a sensor measuring the current is included, $y_i$. Faults are possible in the resistance, $f_R$, and in the sensor, $f_{y_i}$. Including these parameters the circuit can then be described by the following equations, with the voltage over the capacitor and resistor denoted as $u_c$ and $u_R$:*

$$\begin{aligned} e_1 &: u_R = (R + f_R)\, i \\ e_2 &: C\dot{u}_c = i \\ e_3 &: y_i = i + f_{y_i} \\ e_4 &: u = u_R + u_c \\ e_5 &: \dot{u}_c = \frac{d}{dt} u_c \end{aligned} \tag{2.4}$$

*The known variables for this system are the voltage $u$ and the sensor signal $y_i$. The known parameter values are $R$ and $C$ and the unknown variables are $u_R$, $u_c$, $\dot{u}_c$ and $i$. The faults $f_R$ and $f_{y_i}$ are marked in red. Given the variable definitions above, the structural representation of this equation system looks like following:*

| | $u_R$ | $u_c$ | $\dot{u}_c$ | $i$ | $u$ | $y_i$ | $f_R$ | $f_{y_i}$ |
|---|---|---|---|---|---|---|---|---|
| $e_1$ | X | | | X | | | X | |
| $e_2$ | | | X | X | | | | |
| $e_3$ | | | | X | | X | | X |
| $e_4$ | X | X | | | X | | | |
| $e_5$ | | X | X | | | | | |

### 2.2.1 Dulmage-Mendelsohn decomposition

A structural model can be reorganized into a Dulmage–Mendelsohn canonical decomposition, by reordering of rows and columns, [18] as shown in Figure 2.7.

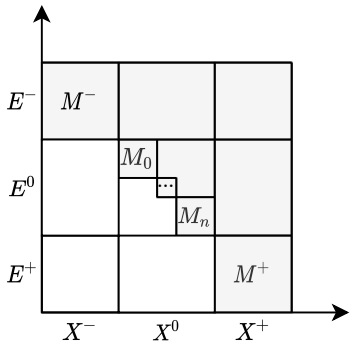



Figure 2.7: General structure of a Dulmage-Mendelsohn canonical decomposition.

The Dulmage-Mendelsohn decomposition divides the model into three parts: an under-determined part, $M^-$, an exactly determined part, $M_0$, and an over-determined part, $M^+$. The over-determined part is the part that contains redundancy, meaning that there are more equations than unknown variables. Redundancy is essential for fault diagnosis and to create residual generators.

In the overdetermined part of the model there are subsets of equations called minimal structurally overdetermined (MSO) sets [5]. An MSO set has redundancy 1, meaning that there is one more equation than unknown variables. MSO sets are the minimal equation sets that can be used to construct residual generators. They are sensitive to few faults which is useful for fault isolation [19].

A fault is defined as structurally detectable if it is included in an overdetermined set. A fault, $f_i$ is structurally isolable from another fault, $f_j$, if the equation including $f_i$ remains in the overdetermined part when the equation containing $f_j$ is removed [5], i.e if $f_i$ is to be isolable it have to be possible to design a residual that is sensitive to $f_i$ but not $f_j$.

Unknown faults can be structurally detected and isolated by including them as generic faults in equations even though there is not a known fault there. If a residual deviates from zero, one of the equations used to derive the residual generator is no longer correct. A fault signal that is non-zero can be interpreted as the nominal equation no longer being valid. This means that when a unknown fault enters the system an equation representing a certain component will no longer be valid.

## 2.3 Sequential residual generation

Solving an equation set requires that the number of equations equals the number of unknown variables to be solved. If the equation set consists of more equations than unknown variables, the redundant equations can be used for residual generation. Any equation in the MSO set can be used as the residual equation, i.e the equation used for detecting inconsistencies between observations and model predictions of the MSO set. When this equation is removed from the MSO set, there remains an equal amounts of unknown variables and equations. The remaining equation system is solved using a matching algorithm [5]. The matching algorithm describes in which order the unknown variables should be solved.

Depending on the computational order, the state variables in the dynamic equations can be computed by either differentiation or integration. The way that the states are computed is defined as causality. If all states are computed using integration the computation have integral causality. If they are computed by only using differentiation the computation have derivative causality. If the computation is done by both integrating and differentiating states it is defined as mixed causality. Integral causality is less sensitive to measurement noise compared to using derivative causality, but it can have stability issues. Derivative causality can be very sensitive to noise, but has less stability issues. Generally integral causality is preferred due to the noise sensitivity of derivative causality [19]. Residuals with integral causality can be written in state-space form which is required when generating residuals with the grey-box RNN method [7]. This will be presented in more detail in Section 3.3.

The way that a set of equations is solved can be illustrated by a computational graph. A computational graph is a directed graph showing the computation order and how variables are connected.

**Example 5** *Using the same system as in Example 4 with $e_4$ as the residual equation the computational graph would look like this:*

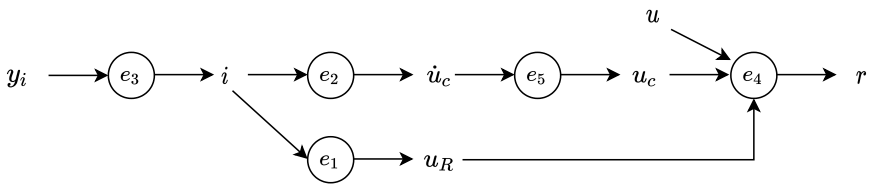



Figure 2.8: Computational graph for the electrical circuit in Example 4 using $e_4$ as the residual equation.

*Figure 2.8 is a computational graph with integral causality since there is only integration used and no differentiation.*

*The analytical expression for the residual, based on the solution path of the computational graph, is:*

$$r = u - u_C - R\, y_i, \tag{2.5}$$

*where $u_C$ is calculated by integrating the right-hand side of the following equation:*

$$\dot{u}_C = \frac{y_i}{C} \tag{2.6}$$

# Chapter 3

# Data-driven modeling

This chapter introduces the necessary theory and techniques used in this thesis related to data-driven modeling. Data-driven modeling is a modeling technique that utilizes data instead of system knowledge to model functions/systems. The benefits of data-driven modeling is that it is a way of modeling without the need of explicit information of the physical properties of the function/system. The downside is the need of representative data and often not able to generalize to system behavior that the model is not trained on.

## 3.1 Artificial Neural Networks

Neural networks are a data-driven modeling technique and among other things used to approximate linear or nonlinear functions. According to the universal approximation theorem, a neural network can represent any arbitrarily complex function given enough data and the right structure of the network [20]. This makes it a very powerful modeling technique. It is in a sense a black-box, since when creating and training a neural network all that is needed is input and output data of the system to be modeled. This can be done without any knowledge of how the system actually works. Although knowledge about the system will help to design and train the neural network, it is not needed as there are general neural networks available.

An artificial neural network (ANN) is as its name suggests a modeling technique inspired by biological neural networks. An ANN, also more broadly called neural network (NN), is essentially a mathematical model that tries to simulate the structure and functions of biological neural networks [20]. A NN consists of a collection of nodes called artificial neurons, which are simple mathematical functions that simulate the function of biological neurons. Every node scales all the inputs to that node with a weight so that every input is scaled by individual weights. After that, all the weighted inputs plus any bias are summed together. The last function of an artificial neuron is an activation function, also called transfer function, which scales the output. The activation function is used to normalize the output of neurons and as its name suggests it decides how ”active” a neuron is.

Figure 3.1 shows the basic structure of an artificial neuron. Where $x_1$ to $x_n$ are the outputs from other nodes, $b$ is the bias term and $y$ is the output of the node.

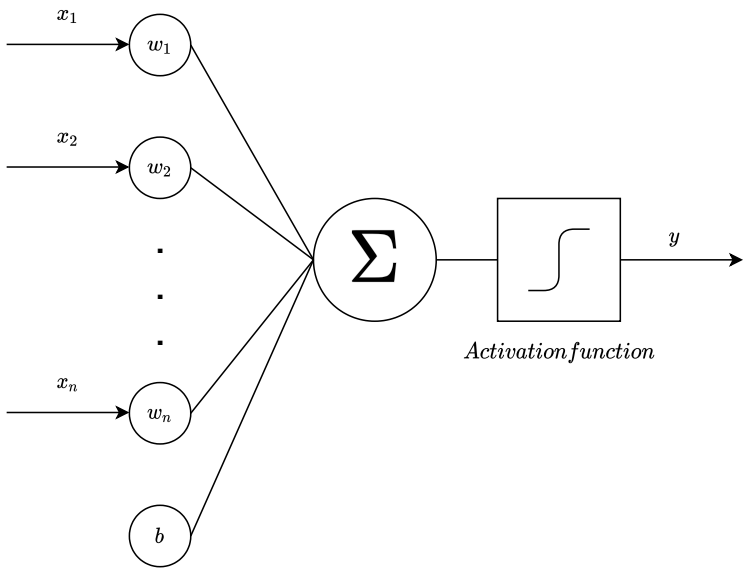



Figure 3.1: Structure of an artificial neuron.

There are various activation functions to choose from, historically the sigmoid function

$$f_{sigmoid}(x) = \frac{1}{1 + e^{-x}}, \tag{3.1}$$

has been a tool of choice for incorporating non-linearities in neural networks. In this function $x$ is the input to the function and $x$ includes all inputs to the neuron summed plus bias terms.

However, more recently the activation function called Rectified Linear Unit (ReLU),

$$f_{ReLU}(x) = \max(x, 0), \tag{3.2}$$

has gained popularity. In ReLU $x$ is the input to the function and the function removes negative values by outputting the maximum of the input and 0. For the NNs in this thesis, ReLU was used.

A single neuron is quite simple, not that impressive and not capable of doing very much. However, when connected in a network, the result can be extremely complex and capable of modeling, in theory, any function. Examples of this are biological neural networks, complex and very capable. The behavior of neural networks is determined by relationships between neurons. These relationships are decided by weights, bias, activation and connections between the neurons. In biological neural networks, neurons can adapt over time. This adaptation is known to be key properties in how functions such as memory and learning work. The learning of neural networks is described in detail in Section 3.1.1.

Neural networks consist of a collection of neurons that all work in the same simple way as described above and illustrated in Figure 3.1. These neurons are interconnected in a structured way that is standardized to help with easier, faster and more efficient problem solving. A general neural network structure is illustrated in Figure 3.2. A NN consists of an input layer, output layer and n hidden layers. $u_1$ to $u_i$ are inputs to the neural network and represent the features of the data input to the NN. They are connected to neurons in the first layer of the neural network, called the input layer. Since this example is of a fully connected network each neuron in the input layer is then connected to the next layer $h_1$ called the hidden layer, and so on through all hidden layers $h_1$ to $h_n$. The last layer of the neural network is called the

output layer which is connected to the last hidden layer $h_n$. The output layer $y_1$ to $y_j$ is what the neural network outputs and depends on what is being estimated if it is a classification or regression [20]. The amount of hidden layers is called the depth of the neural network, a network with a large number of layers is called a deep neural network. The amount of neurons in the individual hidden layers is usually referred to as the width of the network.

The width and depth, activated neurons, etc. all depend on the problem to be solved, different topographies fit different problems. With an increase in width and depth of a NN the complexity of the network increases, more data for training will be needed and take a longer time to train. The benefit is an increase in the ability to approximate more complex functions. The topography is up to the designer of the NN. Where width, depth, cost function, optimization function, activation function, learning rate, normalization of data, etc. has to be chosen. Different problems require different structures, which normally is determined through experiments.

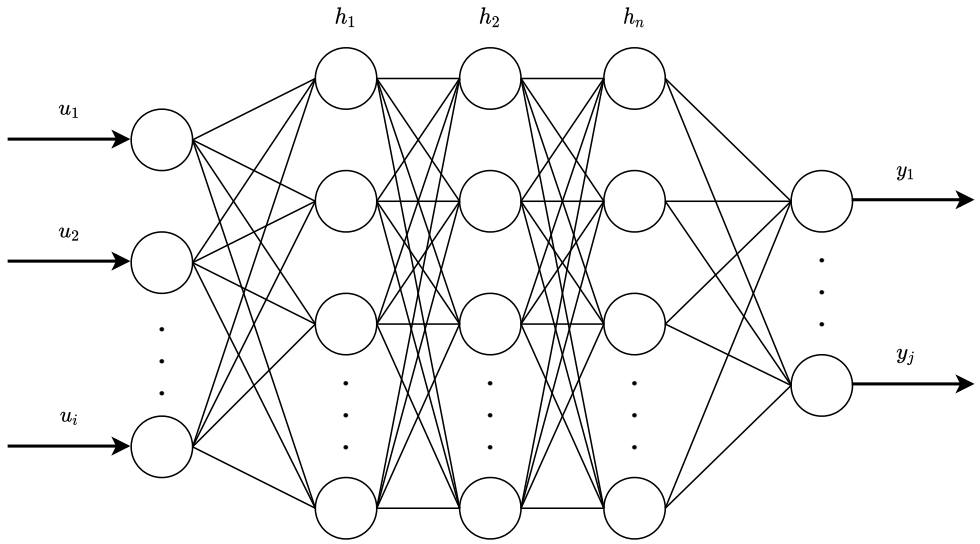



Figure 3.2: General structure of an artificial neural network.

### 3.1.1 Training

After the design and structure of a NN is complete the next step is the optimizing of the parameters in the NN. The optimization process to fit parameters of a NN to model a function is called training of the NN.

#### 3.1.1.1 Training Data

Training data is a set of data that contains examples of relevant system behavior and is used for parameter estimation or a learning process to fit parameters or weights.

For neural network-based models it is important that the training data contains as many relevant working points as possible. Data-driven models such as NN:s models the relations in training data, meaning that it is hard for a NN to extrapolate what would happen in a region that it is not trained on. This makes it extra important to include working points in the training data for a NN compared to parameter estimation for a grey-box model.

#### 3.1.1.2 Loss function

The training of a neural network is done by minimizing a loss function. There are different loss functions that can be used for regression. A common one is the mean squared error (MSE) loss function. It is defined as following, where $y$ is the measured value and $\hat{y}$ is the model prediction:

$$L_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.3}$$

#### 3.1.1.3 Optimization algorithms

There are many different optimization algorithms that can be used to minimize the loss function. A simple and commonly used method is gradient descent. The gradient of the loss function is computed for a certain set of weights, $w$, and multiplied with a learning rate, $\eta_{lr}$. This product is then subtracted from the old weights to get new weights, see the equation below.

$$w_{t+1} = w_t - \eta_{lr}\nabla_w L(w_t) \tag{3.4}$$

This is then repeated until the loss function converges towards a minimum. The learning rate is a hyperparameter that needs to be chosen appropriately.

There are more advanced and efficient algorithms that can be used. For this thesis, an optimization algorithm called Adam has been used. Adam is an adaptive learning rate algorithm, meaning that the learning rate is gradually updated during the training. See [21] for more information.

## 3.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN) is a type of NN that is useful when dealing with sequential data like time-series data. Recurrent neural networks use information from previous inputs to compute the current input and output [20].

Recurrent neural networks use hidden states, $h$, that contain information about the previous inputs and outputs. The hidden state is computed for each time step and used for the computation of the hidden state in the next time step [20]. See Figure 3.3 for an illustration of an RNN.

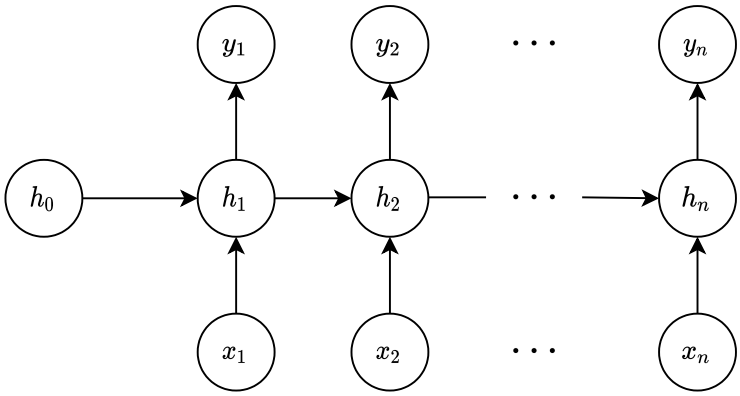



Figure 3.3: General structure of a recurrent neural network.

## 3.3 Design of RNN using structural models

The method of designing the RNN:s using the structural models is described in [7] and will be summarized here. To be able to use an RNN for residual generation, the residual must have a computational graph with integral causality. This makes it possible to write the residual on state-space form as

$$\begin{aligned} \dot{x} &= \bar{g}(x,u), \\ r &= y - h(x,u), \end{aligned} \tag{3.5}$$

where the functions $\bar{g}(x,u)$ and $h(x,u)$ are non-linear functions modeled with a general neural network structure. To utilize information from the structural model, the input arguments ($x$ and $u$) to $\bar{g}(x,u)$ are determined by backtracking from each state derivative, $\dot{x}$, using the computational graph of the MSO set until an input signal, $u$, or a state, $x$, is found. The input arguments to $h(x,u)$ are found by backtracking from the residual equation using the same method [7]. By using this method each function, $\bar{g}(x,u)$ and $h(x,u)$, has information on how the variables in a particular MSO set are connected. This is the reason to why this method is not expected to require as much training data for fault classification as general structure neural networks.

The state-space model is then discretized using Euler forward with the sample time, $T_s$, as following:

$$\begin{aligned} x_{k+1} &= x_k + T_s\, \bar{g}(x_k, u_k) \\ r_k &= y_k - h(x_k, u_k) \end{aligned} \tag{3.6}$$

The following example illustrates how the transformation from structural model to RNN is done:

**Example 6** *By backtracking in the computational graph in Example 5 the arguments to the two non-linear functions being modeled by RNN:s, $g(x,u)$ and $h(x,u)$, can be determined:*

$$\begin{aligned} u_{C_{k+1}} &= u_{C_k} + T_s\, \bar{g}(y_i) \\ r_k &= u - h(u_{C_k}, y_i) \end{aligned} \tag{3.7}$$

*Now when the arguments to the functions are determined, this information is used to generate the RNN with the same model structure as the residual in* (2.5) *and* (2.6).

# Chapter 4

# System description

This chapter describes the exhaust gas treatment system, how the diesel exhaust fluid (DEF) flow system has been modeled in this thesis, and how faults are structurally implemented.

## 4.1 Exhaust gas treatment system

The exhaust gas treatment system on modern Scania trucks shown in Figure 1.1 includes five main parts: a diesel particulate filter (DPF), a diesel oxidation catalytic converter (DOC), two selective catalytic reduction catalysts (SCR), two extra ammonia slip catalytic converters (ASC), and the evaporator and its pressure build up system. This thesis will only take the urea dosing and pressure build up part of the system into consideration.

The evaporator will be referred to as the dosing unit in the rest of this report. The dosing unit and its associated components, the pump, tank and the hoses and filters main task is to transfer DEF from the tank into the exhaust gas treatment system. It accomplishes this by running the pump to build up the pressure in the dosing unit to 10 bar. Then the dosing unit opens a valve that releases DEF into the exhaust gas housing, where it can combine with the exhaust gases.

## 4.2 System model

The delimited system consists of a reductant pump, tank and a dosing unit. The components are connected by electrically heated hoses. Inside the pump, there is a main filter, pre-filter and an overflow valve. The intake of the pump is connected to the tank and the outtake is connected to the dosing unit. There is a filter in the tank for the flow into the pump. Inside the dosing unit, there is an inlet filter and a solenoid valve that controls the dosing amount. Before the flow leaves the dosing unit it passes through a restriction that builds up the pressure in the circuit. The cooling circuit is not considered, only the reductant circuit.

The pressures within the system are modeled using control volumes and are represented as pressure states $p_{tp}$, $p_{bp}$, $p_{ap}$ and $p_{du}$. These states are modeled using the continuity equation

$$\sum q_{in} = \frac{dV}{dt} + \frac{V}{\beta_e}\frac{dp}{dt}, \tag{4.1}$$

where $q_{in}$ is the inflow, $V$ is the control volume, $p$ is the pressure and $\beta_e$ is the effective bulk modulus. The volumes are constant and therefore the volume derivative term $\frac{dV}{dt}$ is zero in all cases for this system.

The flows through each filter are modeled as flows through restrictions using the orifice equation

$$q_{orifice} = C_q A \sqrt{\frac{2}{\rho}(p_1 - p_2)}, \tag{4.2}$$

which is derived from Bernoulli's equation and (4.1). The parameter $C_q$ is the flow coefficient, $A$ is the cross-sectional area of the orifice, $\rho$ is the fluid density and $p_1$, $p_2$ are the pressures before and after the orifice, respectively. The restriction in the dosing unit is also modeled as an orifice. The dosing flow is also modeled using the orifice equation, but with a slight modification. The orifice equation is multiplied by the requested duty cycle (DC) from the control unit. Where the duty cycle is defined as requested dosing flow divided by maximum dosing flow. The exact equations used for the model will be presented in more detail in Section 4.5.

A schematic overview of the system shown in Figure 4.1 illustrates how the system has been modeled without consideration to actual spatial placements of the components. The flows are depicted as arrows in between the components, measured signals are circles, and $p_{bp}$ is the only state that is not measured.

There are two components that are not shown in Figure 4.1 that have essential functions for the system. There is a pressure relief valve in the pump to protect the components. It makes sure that the system pressure does not exceed 13.5 bar. This component is not included since there is no fault data available for this component and it does not affect the nominal system behavior. The other component is a vent in the tank coupling the tank pressure to atmospheric pressure.

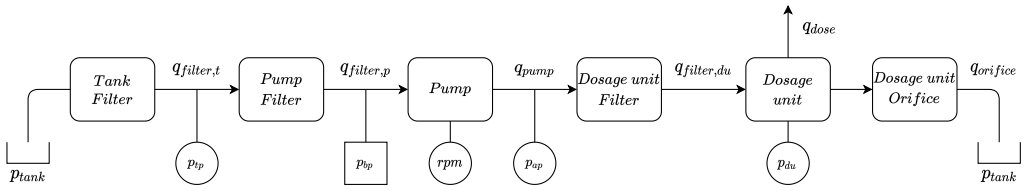



Figure 4.1: Schematic view of the modeled system with the pressure sensors represented as circles.

## 4.3 Assumptions

In order to develop a system model using standard component models, the following assumptions are made when selecting the model structure.

- No losses in hoses
- Tank pressure is constant
- Exhaust pressure is equal to atmospheric pressure
- The pump has a constant displacement
- No leakage
- Cylindrical hoses
- Constant density
- Constant cross-sectional areas and discharge coefficients

- Constant effective bulk modulus throughout pressure ranges
- Sensor dynamics are neglected

## 4.4 Delimitations

Some delimitations of the Scania exhaust gas treatment system have been made for this thesis to be of an appropriate size to fit within 4 months of work and to be large enough to show the characteristics of both of the diagnosis methods in question.

- Only the high-pressure side of the urea dosing system (UDS) is considered, The exhaust part of the exhaust gas treatment system will not be considered.
- Cooling circuit is not considered, only the reductant circuit.
- Pump flow is calculated directly from the measured pump speed, i.e. the pump is not modeled from control input to flow.
- Pressure relief valve is not modeled.

## 4.5 Component models

The equations used to model the systems individual components are described here. The pump flow can be described by

$$q_{pump} = \epsilon_p D_p n_p \eta_{volp}, \tag{4.3}$$

where $\epsilon_p$ is the displacement setting $[0, 1]$, $D_p$ is the pump displacement $[m^3/rev]$, $n_p$ is the pump speed $[rev/s]$ and $\eta_{volp}$ the volumetric efficiency $[0, 1]$ is

$$\eta_{volp} = 1 - C_v \frac{\Delta p}{|\epsilon_p| n_p \eta}, \tag{4.4}$$

where $C_v$ is the laminar leakage losses, $\Delta p$ the pressure difference over the pump and $\eta$ the viscosity.

The flow through the filters can be described by the orifice equation. The first filter, the tank filter is modeled as

$$q_{filt,t} = C_q A_t \sqrt{\frac{2}{\rho}(p_{tank} - p_{bp})}, \tag{4.5}$$

where $A_t$ is the tank filters cross-sectional area and the pressures $p_{tank}$ and $p_{bp}$ are the pressures before and after the filter.

The pump filter flow is modeled as

$$q_{filt,p} = C_q A_p \sqrt{\frac{2}{\rho}(p_{tp} - p_{bp})}, \tag{4.6}$$

where $A_p$ is the pump filters cross-sectional area and the pressures $p_{tp}$ and $p_{bp}$ are the pressures before and after the filter.

The dosage unit inlet filter flow is modeled as

$$q_{filt,du} = C_q A_{du} \sqrt{\frac{2}{\rho}(p_{ap} - p_{du})}, \tag{4.7}$$

where $A_{du}$ is the dosage filters cross-sectional area and the pressures $p_{ap}$ and $p_{du}$ are the pressures before and after the filter.

The outlet orifice flow in the dosage unit is calculated using the same equation as the filters but with different parameters, like

$$q_{orifice} = C_q A_{ori} \sqrt{\frac{2}{\rho}(p_{du} - p_{tank})}, \tag{4.8}$$

where $A_{ori}$ is the orifices cross-sectional area and the pressures $p_{du}$ and $p_{tank}$ are the pressures before and after the orifice.

The duty cycle (DC) that is used to calculate the dosage flow is calculated as

$$DC = \frac{q_{dose,req}}{q_{dose,max}}, \tag{4.9}$$

that takes the requested dose divided by the maximum dosage flow possible at system pressure. The DC (4.9) was converted to a Pulse Width Modulation (PWM) signal as

$$PWM_{dose} = f(DC), \tag{4.10}$$

by implementing a PWM conversion function $f()$ in Simulink.

The average dosing flow can be calculated using the orifice equation multiplied by the duty cycle, as

$$q_{dose} = DC \cdot C_q A_{dose} \sqrt{\frac{2}{\rho}(p_{du} - p_{exh})} \tag{4.11}$$

A more accurate way of modeling the dosing flow would be to include the PWM signal (4.10). The flow can be calculated in a similar way as (4.11) but replacing the DC with the PWM as

$$q_{dose} = PWM_{dose} \cdot C_q A_{dose} \sqrt{\frac{2}{\rho}(p_{du} - p_{exh})}, \tag{4.12}$$

The four pressure states are calculated using the continuity equation. The first state, the pressure between the filters before the pump, is modeled as

$$\frac{dp_{tp}}{dt} = \frac{\beta_{tp}}{V_{tp}}(q_{filt,t} - q_{filt,p}), \tag{4.13}$$

where $\beta_{tp}$ is the bulk modulus and $V_{tp}$ the volume of the control volume between the filters.

The second state, the pressure before the pump, is modeled the same way by

$$\frac{dp_{bp}}{dt} = \frac{\beta_{bp}}{V_{bp}}(q_{filt,p} - q_{pump}), \tag{4.14}$$

where $\beta_{bp}$ is the bulk modulus and $V_{bp}$ the volume of the control volume before the pump.

The pressure after the pump, $p_{ap}$, is modeled as

$$\frac{dp_{ap}}{dt} = \frac{\beta_{ap}}{V_{ap}}(q_{pump} - q_{filt,du}), \tag{4.15}$$

where $\beta_{ap}$ is the bulk modulus and $V_{ap}$ the volume of the control volume after the pump.

The last state $p_{du}$ is modeled as

$$\frac{dp_{du}}{dt} = \frac{\beta_{du}}{V_{du}}(q_{filt,du} - q_{orifice} - q_{dose}), \tag{4.16}$$

where $\beta_{du}$ is the bulk modulus and $V_{du}$ the volume of the dosing unit control volume.

## 4.6 One state model

A simpler model of the system can be constructed only using one state. When doing this, all the other pressures will be modeled without any dynamics. This is possible because the dynamics between the pressures in the system before the dosing unit can be neglected because the dynamics are fast, since the control volumes are small and the effective bulk modulus is large so that the pressure derivative becomes large. The flow can then be seen as the same in and out of the first three pressure states. This can be assumed since in the nominal case they are only separated by clear filters which do not restrict the flow significantly. The first three pressures $p_{tp}$, $p_{bp}$ and $p_{ap}$ can be modeled using algebraic equations i.e. the orifice equation (4.2).

The only state for this simpler model is $p_{du}$ described by

$$\frac{dp_{du}}{dt} = \frac{\beta_{du}}{V_{du}}(q_{pump} - q_{orifice} - q_{dose}), \tag{4.17}$$

where previously the input flow was through a filter it is now $q_{pump}$ since the flow through the pipe and filter is seen as having fast enough dynamics for the pressure to be modeled as algebraic.

The three remaining pressures in the system $p_{tp}$, $p_{bp}$ and $p_{ap}$ can then be calculated using the orifice equation (4.2). The expressions are the following

$$p_{tp} = p_{tank} - \frac{\rho}{2}\left(\frac{q_{pump}}{C_q A_t}\right)^2, \tag{4.18}$$

$$p_{bp} = p_{tp} - \frac{\rho}{2}\left(\frac{q_{pump}}{C_q A_p}\right)^2, \tag{4.19}$$

$$p_{ap} = p_{du} + \frac{\rho}{2}\left(\frac{q_{pump}}{C_q A_{du}}\right)^2, \tag{4.20}$$

where the pressures on one side of a filter is dependent on the other side's pressure and the cross sectional area of the filter.

## 4.7 Fault models

Possible known and unknown faults for the system need to be included in the system equations for the structural analysis. This enables the use of structural methods to facilitate residual generation and analysis of the detectability and isolability of the system. A majority of the faults for this system are blockages in the different components and are modeled as an additive fault on the cross-sectional area of the different components. These faults are faults in the tank filter ($f_{A_t}$), pump inlet filter ($f_{A_p}$), dosing unit inlet filter ($f_{A_{du}}$), dosing unit orifice ($f_{A_{ori}}$) and the dosing unit nozzle ($f_{A_{dose}}$). A general pump flow fault is modeled as an additive fault on the pump flow equation and is denoted as $f_{q_{pump}}$. For each pressure sensor there is a additive sensor fault modeled denoted as $f_{p_{tp}}$, $f_{p_{ap}}$ and $f_{p_{du}}$.

$$q_{pump} = \epsilon_p D_p n_p \eta_{volp} + f_{q_{pump}} \tag{4.21}$$

$$q_{filt,t} = C_q(A_t + f_{A_t})\sqrt{\frac{2}{\rho}(p_{tank} - p_{bp})} \tag{4.22}$$

$$q_{filt,p} = C_q(A_p + f_{A_p})\sqrt{\frac{2}{\rho}(p_{tp} - p_{bp})} \tag{4.23}$$

$$q_{filt,du} = C_q(A_{du} + f_{A_{du}})\sqrt{\frac{2}{\rho}(p_{ap} - p_{du})} \tag{4.24}$$

$$q_{orifice} = C_q(A_{ori} + f_{A_{ori}})\sqrt{\frac{2}{\rho}(p_{du} - p_{tank})} \tag{4.25}$$

$$q_{dose} = PWM\, C_q(A_{dose} + f_{A_{dose}})\sqrt{\frac{2}{\rho}(p_{du} - p_{exh})} \tag{4.26}$$

$$y_{p_{tp}} = p_{tp} + f_{p_{tp}} \tag{4.27}$$

$$y_{p_{ap}} = p_{ap} + f_{p_{ap}} \tag{4.28}$$

$$y_{p_{du}} = p_{du} + f_{p_{du}} \tag{4.29}$$

The faults are modeled as either additive or multiplicative faults for each component. The equations (4.21) to (4.29) show how faults (marked in red) that can occur in the system have been modeled. This is also how it is implemented in the Fault Diagnostics Toolbox [6].

# Chapter 5

# Diagnosis system design process

This chapter presents the method used for the development, implementation and evaluation of the diagnosis systems for the thesis work. It will also include some results from the design process of the diagnosis systems.

## 5.1 General implementation process

The implementation for this thesis has been an iterative process where the first iteration was completed using a simple model of the system. For each iteration, the model was extended. The results presented in the report are the results for the final iteration.

The work can be divided into subcategories as shown in Figure 5.1.

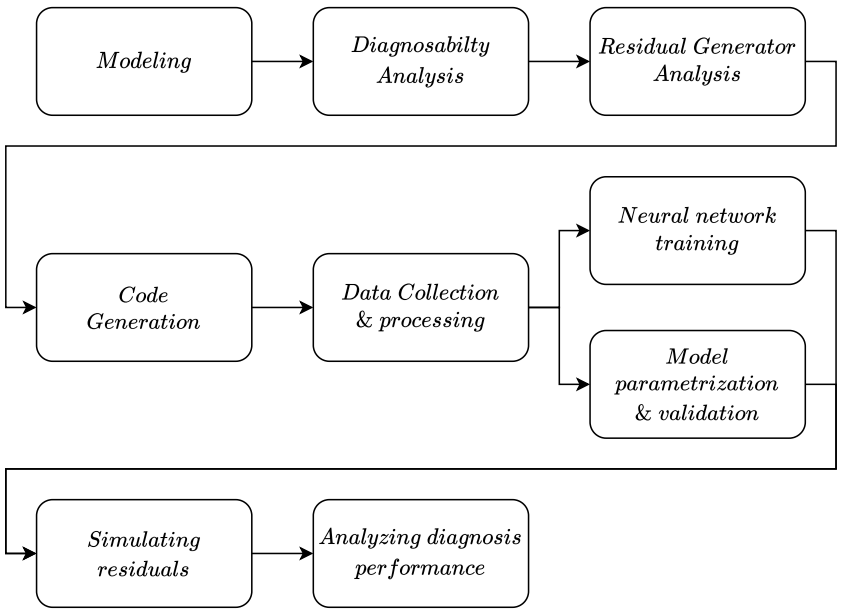



Figure 5.1: Workflow for the design process.

## 5.2 Modeling

All of the physically-based modeling in this thesis was done in Matlab and Simulink. Simulink was initially used since Scania provided some previous models implemented in Simulink. These Scania models were used as a starting point for development of the models for this thesis. Using Simulink was also an easy way to change parameters and analyze specific signals in the model for troubleshooting purposes. Later the models were transferred to Matlab where parameter estimation was done using the System Identification Toolbox [22]. Two physical-based models were developed, one simpler model neglecting some of the pressure dynamics only containing one state and a more complex model containing more dynamics with four states. Both models and their associated equations are presented in Chapter 4. All results and figures presented in the following sections will be for the four state model.

## 5.3 Sensor selection

The current UDS has a limited amount of internal sensors with only the pump speed and dosage unit pressure being measured. With the original set of sensors in the system, the faults are structurally detectable but not isolable. Therefore additional sensors have been mounted to improve isolability performance. The isolability matrix when not using any external sensors is shown in Figure 5.2. It shows that in the ideal case all faults are detectable but none are isolable.

Two high-performance pressure sensors were ultimately chosen due to at that time in the thesis they were the only sensors available since Scania had previously ordered them. Data from the additional sensors will also be used for parameter estimation of the equation-based models.

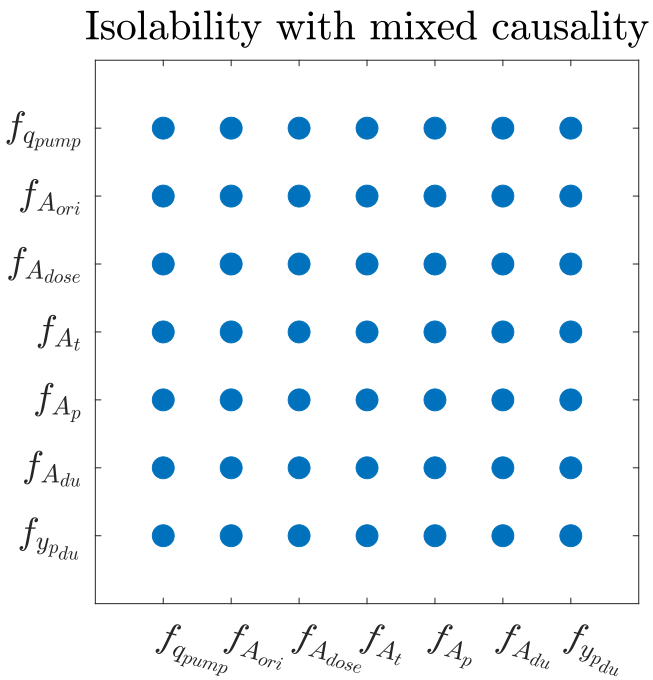



Figure 5.2: Isolability matrix with mixed causality when using no external sensors.

The placement of the extra two pressure sensors was determined using the Fault Diagnosis Toolbox (FDT) and system knowledge of where sensor placement is possible. The first pressure

sensor was positioned right before the pump, between the tank and pump filter, because it was not possible to place it after the pump filter since it is inside the pump. The second pressure sensor was placed right after the pump. The sensor's placement can be seen in the schematic view of the system in Figure 4.1. The isolability matrices when adding the external sensors can be seen in Figure 5.5.

## 5.4 Structural analysis

The FDT was used to perform structural analysis [6]. The first step when using the FDT is to define the model. This was done by defining the known parameters, unknown parameters, fault signals and system equations as symbolic expressions. Then built-in functions were used to generate figures and relevant structural information.

Figure 5.3 shows the structural representation of the model. The variables are divided into three groups and are illustrated using three different colors. The unknown variables are shown in blue, with the $I$ and $D$ indicating integrated or differentiated variable relation. The faults are represented by red dots and the known variables by black dots.

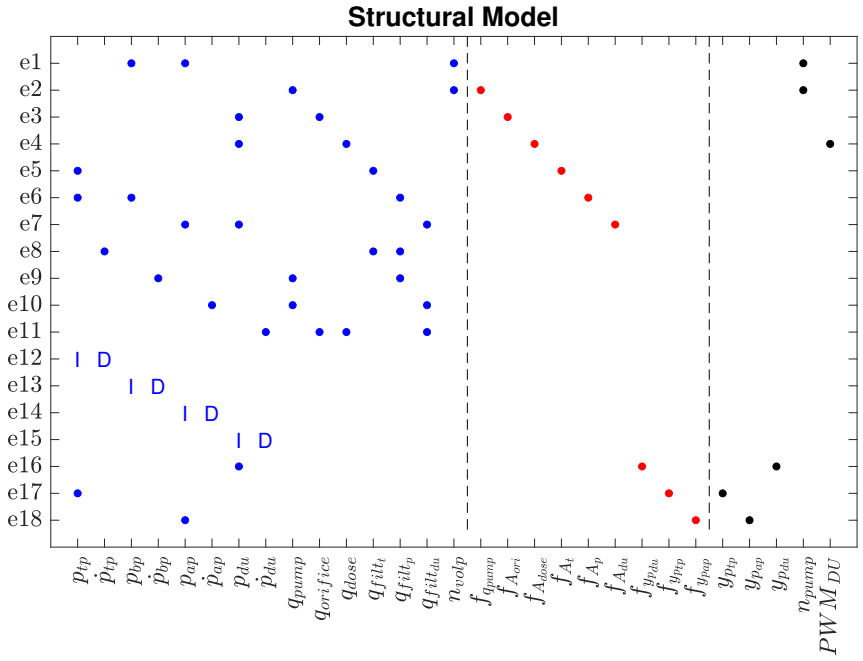



Figure 5.3: Structural model for the final model iteration with the added sensors.

### 5.4.1 Diagnosability analysis

The purpose of diagnosability analysis is to see which faults are detectable and which faults can be isolated or not. This was done in the FDT by plotting isolability matrices. Isolability matrices with integral and mixed causality are shown in Figure 5.5. Note that the order of the faults is different in the two subfigures. The isolability matrix when using residuals with derivative causality is shown in Figure 5.5a. In this case, the derivative causality matrix is the same as the mixed causality matrix which shows the theoretically best isolability possible. Comparing the figures shows that for this system derivative causality has better isolabilty

than integral causality. This matrix shows that a fault in the orifice inside the dosing unit can not be isolated from a fault in the dosage nozzle using derivative or mixed causality. This is expected since both components are connected inside the dosing unit and there is no information available about the outputs of those components to differentiate them.

The Dulmage-Mendelsohn decomposition when using mixed causality is shown in Figure 5.4. The model only consists of an overdetermined part, which is contained in the blue box. The faults and where they enter the model are shown in red. The grey blocks show which equations need to be used together to generate residuals. Faults are only isolable if they enter separate blocks or equations. The Dulmage-Mendelsohn decomposition shows the theoretically best possible isolabilty.

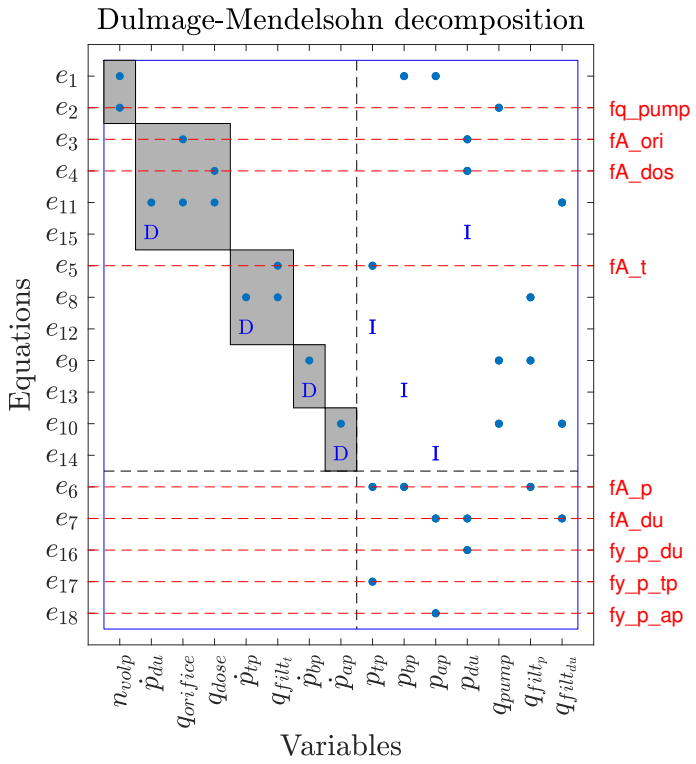
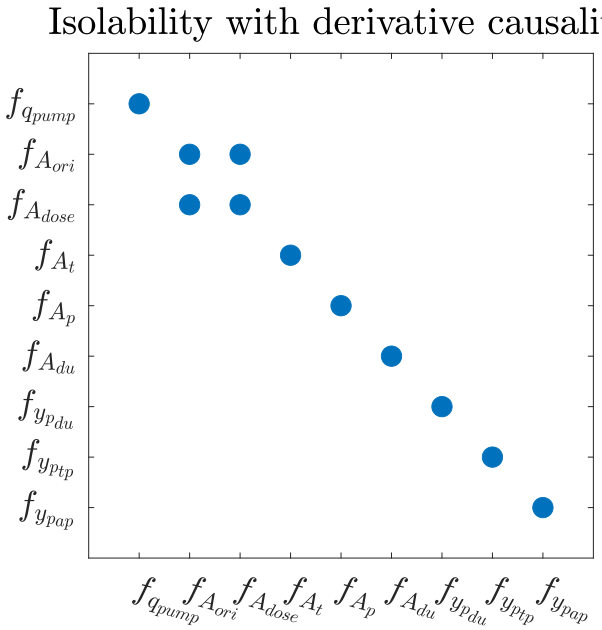
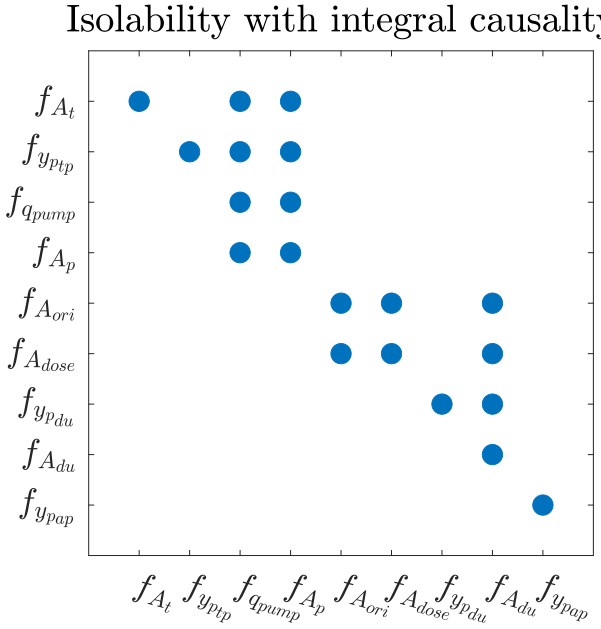



Figure 5.4: Dulmage-Mendelsohn decomposition.

When using residuals with integral causality the isolability matrix looks like Figure 5.5b. Here it can be seen that faults in the tank filter ($f_{A_t}$) and in the sensor measuring the pressure between the tank and the pump ($f_{y_{p_{tp}}}$) can not be isolated from a fault in the pump $f_{q_{pump}}$ or pump filter $f_{A_p}$. A fault in the dosing unit orifice ($f_{A_{ori}}$), dosing unit sensor ($f_{y_{p_{du}}}$) and in the dosing unit nozzle ($f_{A_{dose}}$) can not be isolated from a fault in the dosing unit filter ($f_{A_{du}}$) either.




(a) Derivative causality.




(b) Integral causality.

Figure 5.5: Isolability matrices for the four state model.

## 5.5 Residual generator analysis

The next step was to find appropriate residuals. Since the neural network-based method requires a residual with integral causality, all MSO sets that have at least one residual equation with integral causality were generated. The model-based method can use residuals with derivative causality but for this thesis, only integral causality MSO:s were used. The MSO sets were found using a built-in function in the FDT [6]. The final four-state model has 18 equations and 15 unknown variables which means that the degree of redundancy is 3.

To be able to see which residuals should react to which faults, the FSM was plotted for the chosen MSO sets. The FSM for the model can be seen in Figure 5.6. On the y-axis are the generated MSO sets that were used and on the x-axis are the different faults.

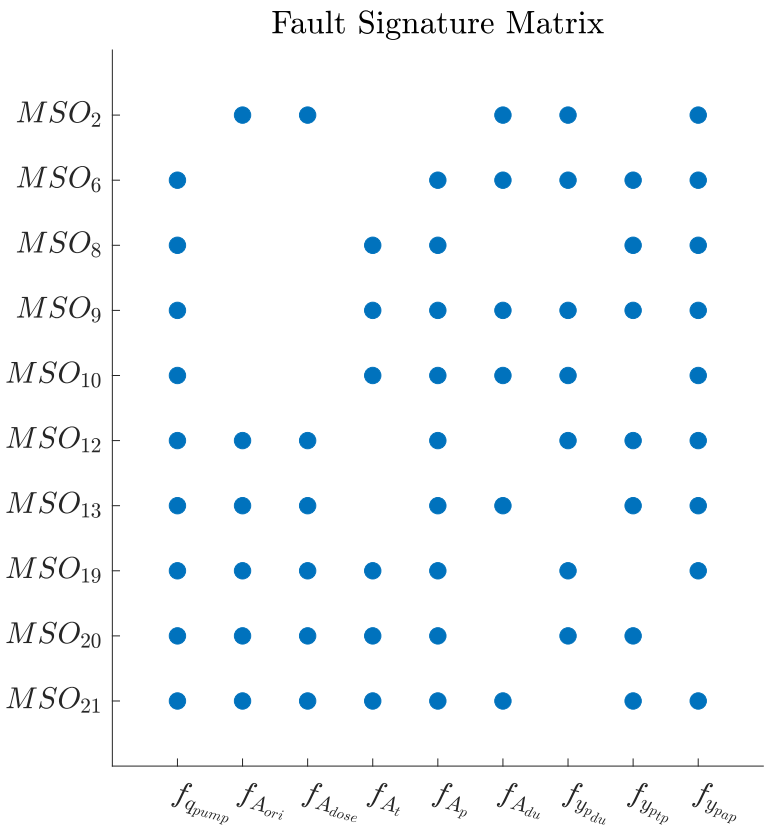



Figure 5.6: Fault signature matrix for the set of MSO's selected for the final model iteration.

## 5.6 Code generation

Matlab code for the model-based residuals was automatically generated for each residual by using the FDT. The residual code could later be simulated given model parameters and measurement data as inputs. The code generated for $MSO_2$ can be seen in Figure 5.7.

```
function [r, state] = Modell7_2_16(z,state,params,Ts)
% MODELL7_2_16 Sequential residual generator for model 'Modell7'
 % Causality: int
 % Structurally sensitive to faults: fA_ori, fA_dos, fA_du, fy_p_du, fy_p_ap

    % Parameters
  Cq = params.Cq;
  A_ori = params.A_ori;
  A_dos = params.A_dos;
  A_du = params.A_du;
  rho = params.rho;
  p_tank = params.p_tank;
  p_exh = params.p_exh;
  V_du = params.V_du;
  B_du = params.B_du;

  % Known variables
  y_p_ap = z(2);
  y_p_du = z(3);
  PWM_dos = z(5);

  % Initialize state variables
  p_du = state.p_du;

  % Residual generator body
  p_ap = y_p_ap; % e18
  q_filt_du = sqrt(2.0)*A_du*Cq*sign(p_ap-p_du)*sqrt(abs(p_ap-p_du)/rho); % e7
  q_dos = (sqrt(2.0)*A_dos*Cq*PWM_dos*sign(p_du-p_exh)*sqrt(abs(p_du-p_exh)/rho))/1.95e+2; % e4
  q_ori = sqrt(2.0)*A_ori*Cq*sign(p_du-p_tank)*sqrt(abs(p_du-p_tank)/rho); % e3
  dp_du = -(B_du*(q_dos-q_filt_du+q_ori))/V_du; % e11

  r=-p_du+y_p_du; % e16

  % Update integrator variables
  p_du = ApproxInt(dp_du,state.p_du,Ts); % e15

  % Update state variables
  state.p_du = p_du;
end

function x1=ApproxInt(dx,x0,Ts)
  x1 = x0 + Ts*dx;
end
```

Figure 5.7: Code for $MSO_2$ automatically generated by the FDT.

The code for the neural network-based residuals was generated for each residual by a script provided by our supervisor Daniel Jung. It is the same script that was used in [7]. The script generates code that setups the structure of the neural network based on the structural model. The generated code is written in Python and utilizes PyTorch [23].

## 5.7 Data collection

The data collection was performed on a Scania truck and a UDS test-bench located at Scania, Södertälje. Data were collected four times in total, each time a little different because it was improved upon from the experience of the previous measurements. The first measurements were collected on a test bench since it is easier to access and perform measurements on and could be done early in the thesis process. It was important to do this early in the scope of the thesis so that the first iterations of the diagnosis method could be created and evaluated. Only data from the last measurement occasion on the truck and test bench will be presented in this report since it is the most accurate and extensive collection.

Communication with the test bench and truck was performed by connecting a laptop to a data acquisition module (DAQ) which was then connected to either the test bench or the truck. Sending control commands and receiving sensor signals was then achieved through the DAQ interface.

A .NET script was written in C# for the communication from the laptop to the ECU controlling the system to control components and read sensor values. In this script the input cycles for the dosing unit and pump and which sensors to read were specified. Also, the creation of a file to save all the data was included in the script. The data collection script sampled the data at as high of a frequency as possible using this setup through the CAN bus which turned out to be ~5 Hz.

The data from the two external sensors were collected using a separate data logging system developed by Scania for this purpose that is still under development. This is a separate data collection system than what was previously described for the internal sensors and controllers. This external data would later have to be synchronized with the internal data. The external signals were sampled at 10 Hz.

All data were collected with the engine turned off and since the system is not connected to a live engine there was no actual dosing and the system pressure was stationary at 10 bar because the pump was closed-loop controlled at that reference pressure. The dosage unit was disconnected from the exhaust system but still connected to the pressure build-up system so that the exhaust would not be flooded with urea. This makes it so that the modeled exhaust pressure is equal to atmospheric pressure. In nominal case connected to the exhaust, it should be roughly the same pressure. The two setups using either a real Scania truck and the test bench both contain the same components and only differ in the length of hoses.

### 5.7.1 Data cycles

To get relevant data for training the neural network, parameterizing the equation-based models and validating both the equation-based models and the neural network models two cycles were created, a validation cycle and a training cycle. These cycles consist of different steps in requested dosage and pump PWM, although not simultaneously. The cycles were in similar ranges but steps were performed at different time points. These two cycles would then be used for collecting training and validation data for both the truck and test bench.

Figure 5.8 illustrates the nominal fault-free data used for training of the neural network and parameter estimation of the equations. The pressure between the filters $p_{tp}$ is around 1 bar because it is directly coupled to the tank, which has a vent out to atmospheric pressure. The pressures in the dosage unit $p_{du}$ and right after the pump $p_{ap}$ have in the fault-free case no dynamics between them, only a constant offset. There are clearly dynamics in the transition from the input signal, pump speed ($n_p$), to the pressure after the pump and in the dosage unit.

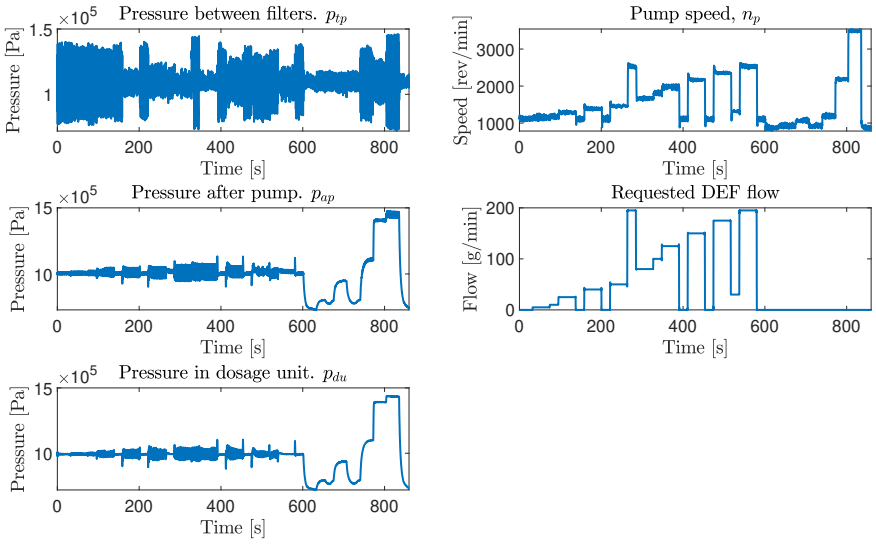



Figure 5.8: Data for the nominal fault free case used as training data.

### 5.7.2 Collecting data from faults

To evaluate the fault detection and isolation performance of both the model-based and the neural network-based residuals, data is collected from a set of fault scenarios. Since the system consists of flows through pipes and restrictions most of the possible faults are restriction faults. There are also other possible faults such as leakage and pump faults. In the end, the following faults were implemented and data was collected using the validation cycle:

- Tank filter, $f_{A_t}$
- Pump filter, $f_{A_p}$
- Dosage unit orifice, $f_{A_{ori}}$
- Dosage unit filter, $f_{A_{du}}$

The faults were implemented the same way on both the test bench and the truck. The three first faults were implemented using valves positioned at the appropriate places in the system for the respective faults. The dosage unit filter fault was implemented by replacing the existing dosage unit with a faulty unit that has a restriction in the filter.

For each fault the valve is turned until a noticeable change in the system behavior is reached, either a change pressure or pump speed. This will be the smallest noticeable fault. After that, another measurement is done for the same fault but with a larger restriction. This is done so that both diagnosis methods can be compared for both small and large faults.

Only one example of fault data will be presented and explained. Figure 5.9 presents data for the fault case $f_{A_{du}}$. The pressure between the filters is not affected by this since it is disconnected from that part of the system by the pump. Since there is a restriction in the filter between the pump and the dosage unit, $p_{du}$ and $p_{ap}$ now have dynamics between them when the flow is large enough for the restriction to take effect. At large dose requests, the flow through the filter is not enough for the pressure to remain constant and the pressure drops

until it is too low and the ECU turns of the dose until the system pressure reaches its reference value again.

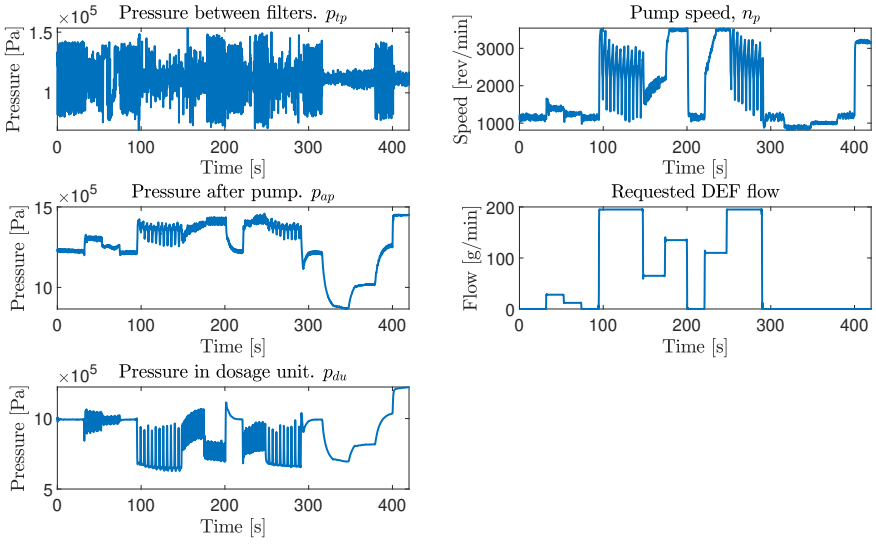



Figure 5.9: Data from the validation cycle with a broken dosage unit implemented. Corresponding to the $f_{A_{du}}$ fault.

## 5.8 Signal processing

The internal sensor data was sampled at approximately 5Hz and the external data was sampled at 10Hz. Since both data sets were not uniformly sampled they were first processed by resampling to a uniform rate of 5Hz. This was done using the `resample()`-function in Matlab which resamples the signal and applies an FIR anti-aliasing low-pass filter to the input signal.

Since the data from the internal and external sensors were collected using two separate logging systems it needed to be synchronized. The synchronization was performed in Matlab using the `xcorr()`-function. This function outputs the cross-correlation of two signals. It measures how similar the two signals are and can be used to synchronize them. For some data sets where `xcorr()` could not find a clear cross-correlation, the synchronization needed to be performed manually. The manual synchronization was done by looking at features, such as a step in pressure, in both signals and lining them up. Since the system dynamics is faster than the sensor's sampling frequency it was deemed fine to synchronize them so that a pressure step in the sensor after the pump and the internal sensor react simultaneously. The data from the external sensors also had to be converted from voltage to pressure. This was done by using a conversion table given in datasheets.

The last step was to calibrate the external sensors. Data was collected from the external sensors measuring the atmospheric pressure in the workshop. Then they were calibrated by adjusting the values to be equal to 1 bar (approximately atmospheric pressure). For this thesis, the important part was not to perfectly calibrate the sensors but to make sure the measurements were consistent.

The dosing unit is controlled by a PWM signal, ideally using this signal would be a more accurate dosing flow model. Using a PWM signal to calculate the flow includes the oscillations in the pressure caused by the valve opening and closing during each PWM pulse. The problem

is that for this thesis acquiring the PWM signal was not a possibility. Therefore a method of estimating the PWM signal from the DC was implemented. The PWM signal is estimated using a Simulink PWM generator from a DC using the right PWM interval.

## 5.9 Model parameter estimation and validation

The parameter values needed for the model equations were identified using grey-box model estimation. The model was made to match measurement data as close as possible. The grey-box modeling was performed with constraints that made sure that the parameters are in a range that is close to the values from data sheets and from previous modeling work that had been done on the system.

The solver used for simulating the model in Matlab, which is a variable-step solver using numerical differentiation formulas [24], was ODE15s. Euler forward and other nonstiff solving methods were either not accurate enough or very slow at simulating. One reason why other solvers were slow partially depends on that the model could easily become unstable due to the four control volumes used for the four states.

Measurement data and predicted output for both the one-state and four state model are plotted and presented in Figure 5.10. The largest errors in the model are at just before 300 seconds and between 500 and 600 seconds. During these times there is a maximal dose which implies that the PWM is constantly on and the dosing unit valve does not oscillate. It is apparent that the model is not accurate during transitions to and from maximal doses.

The farther away from the system pressure of 10 bar the model will also be worse since there is known to be non-linear characteristics due to pump displacement and varying hose stiffness which affects the bulk modulus.

These models, as described in Chapter 4, use the duty cycle as input rather than a PWM since the model does not describe the flow well enough for all dose sizes when using the PWM. The flow is much better modeled using the duty cycle to estimate the average of the flow rather than using the PWM signal.

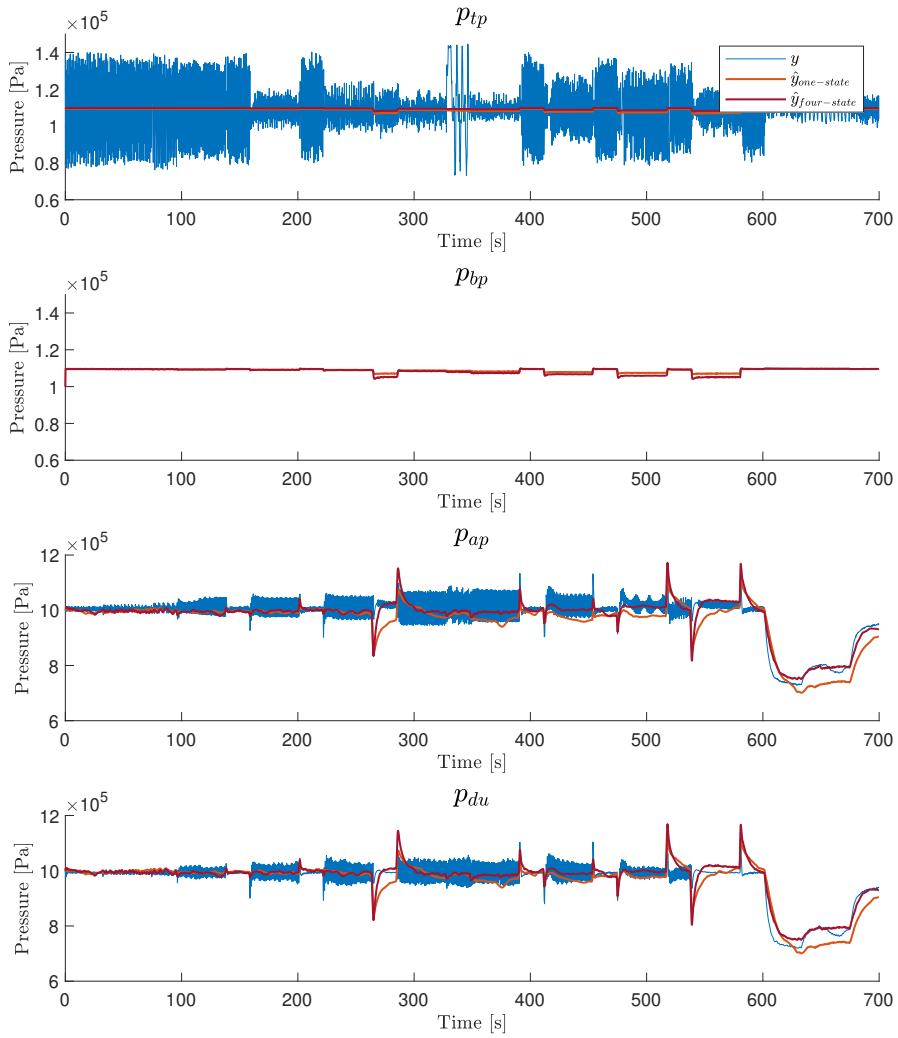



Figure 5.10: Validation of both the four state and one state models. Where blue lines are measured data, orange is the one state model and dark red the four state model. Note that in the second subplot there is no blue line since this pressure is not measured.

## 5.10 Training neural networks

The training of the neural network-based residuals was performed using scripts provided by our supervisor from Linköping University, Daniel Jung. The scripts use the automatically generated code (as mentioned earlier) and trains the NN:s using design parameters that are chosen by the user. The training scripts, just like the generated NN code, are written in Python and utilize PyTorch [23].

First the structure of the NN:s was set up by choosing the number of layers, nodes and what activation function to use. It was chosen based on testing and recommendations in [7]. Each function, $\bar{g}(x, u)$ and $h(x, u)$ (as presented in (3.6)), is modeled using an NN with 3 hidden layers, 128 nodes and the ReLU function as the activation function.

The training was performed by minimizing the mean squared error (MSE) cost function for a certain amount of epochs. The number of epochs was chosen to be around 1500-1800 to achieve a good trade-off between computation time and performance. Adam was used as the optimization algorithm for the training with an adaptive learning rate. The initial learning rate was chosen to be 0.005 and then it was decreased by 2% for every tenth epoch.

Initial values for the states were defined before the training could be started. They were chosen appropriately by looking at the measurement data. The measurement data used as input signals to the neural network were normalized by dividing them with a normalization constant. The normalization constants were chosen so the signals were in the range of 0 to 1. The training script saved the loss functions value for each epoch as well as the final model that can be used for simulation and validation on different data sets.

The value of the MSE loss function for each epoch of the training are presented in Figures 5.11 and 5.12. In Figure 5.11 the MSE for MSO sets modeling $p_{tp}$ are plotted and in Figure 5.12 the MSE for sets modeling $p_{du}$ and $p_{ap}$ are shown. The figures shows how the loss function converges after a certain amount of epochs. This gives an indication on how many epochs the residuals need to be trained for.

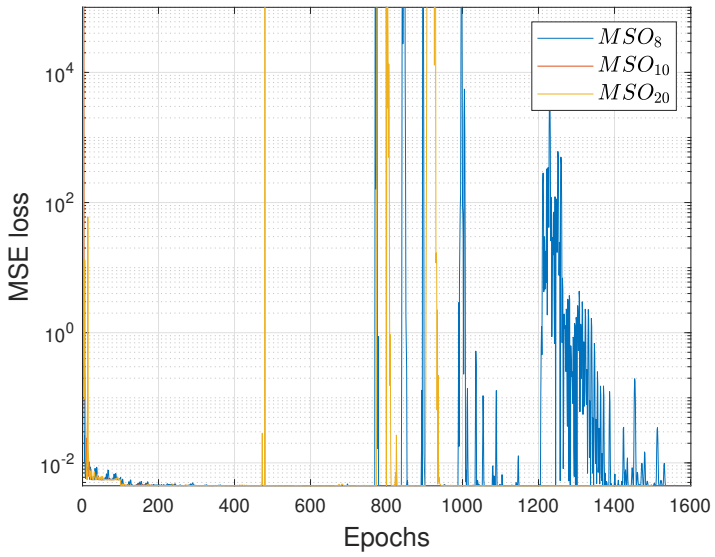



Figure 5.11: MSE for each epoch for MSO sets predicting $p_{tp}$.

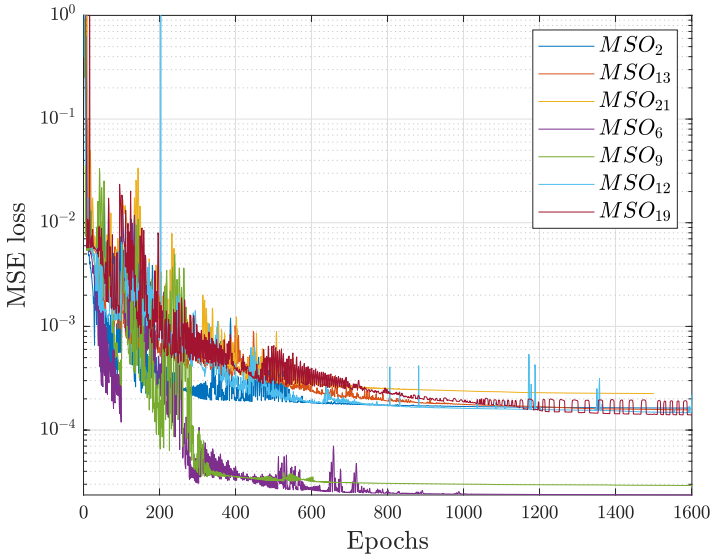



Figure 5.12: MSE for each epoch for MSO sets predicting $p_{du}$ or $p_{ap}$.

Because the initial weights of neurons in the neural networks for each training session are randomized the performance will not be the same every time even if the hyperparameters are the same. The neural networks will converge to local minimums with similar MSE but sometimes the visual difference in a plot is significant. Because of this, it can be beneficial to train the same model multiple times. To show the differences between training iterations $MSO_{13}$ were trained several times with the same hyperparameters and then plotted. As shown in Figure 5.13 the performance differs between each training even though hyperparameters are the same. This will be discussed further in Chapter 7.

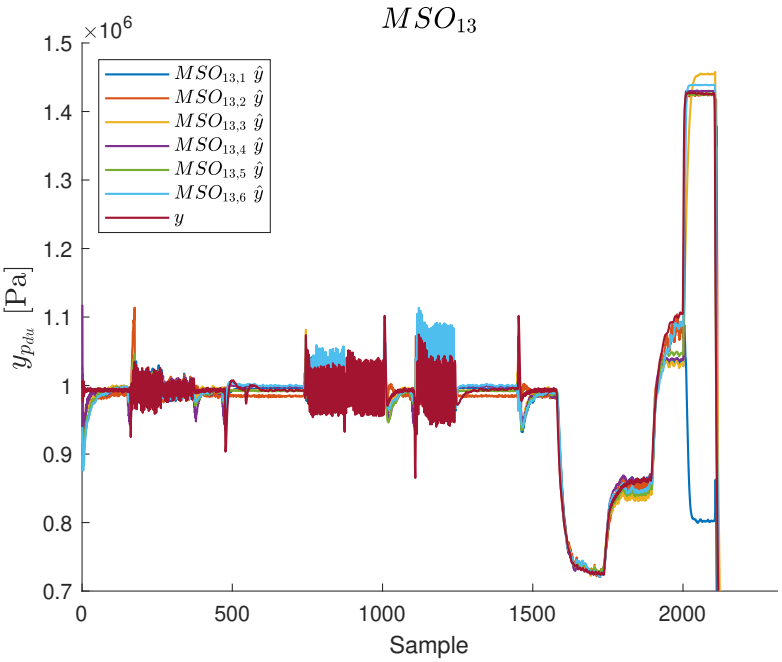



Figure 5.13: $MSO_{13}$ model predictions for different training iterations plotted together with the measured value, $y$.

## 5.11 Residual simulation and analysis

The last step of the workflow was to simulate the residuals and analyze them. The model-based residuals were simulated in Matlab by using the automatically generated code together with measurement data and the parameter values. Each residual was simulated using data from the nominal mode and all fault modes. Then the fault detection performance of each residual were evaluated and compared with the theoretical performance from the structural analysis.

The neural network-based residuals were simulated for all modes using the previously mentioned script that used the trained residual and simulated it for a given data set. Then the residuals were evaluated in the same way as for the model-based residuals, and finally both methods were compared against each other. The results are shown and analyzed in Chapter 6.

# Chapter 6

# Results

This chapter presents results from the implemented residuals, both model-based and data-driven.

## 6.1 Model-based residuals

This section presents the results for the simulated model-based residuals.

### 6.1.1 Validation

All the model-based residuals have been validated using the validation data . Figure 6.1 shows the validation of residuals based on $MSO_2$, $MSO_6$ and $MSO_{10}$ by plotting the measured and modeled outputs in the left column of plots and the corresponding residuals in the right column. It is evident that the residuals have a larger output during larger doses. The modeling for the pressure before the pump is also very flat, and does not follow any oscillations. The variations in the residual noise is due to the oscillations from the PWM.

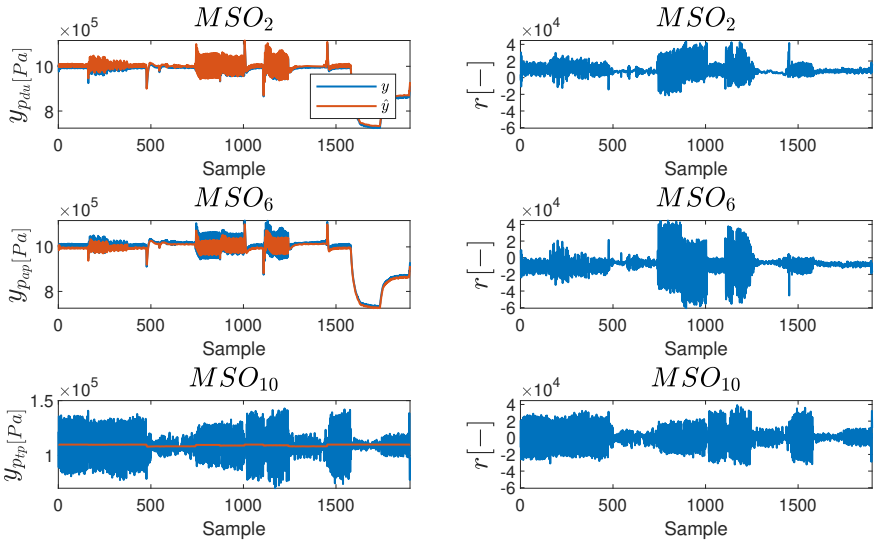



Figure 6.1: Model prediction and measurements in the left column and corresponding residuals in the right column for the model based residuals using validation data for three different MSO sets, $MSO_2$, $MSO_6$ and $MSO_{10}$. Each MSO set in the figure models a different state/pressure of the system.

### 6.1.2 Histograms

To evaluate the performance of all residuals for the implemented faults, histograms have been plotted in Figures 6.2 and 6.3. These figures contain histograms for each fault in separate columns and a residual based on a different MSO set for each row. In the histograms the blue bars are the no-fault case and the orange is for the fault case. The no-fault data is the same for every histogram, but a new prediction is made for every residual. Faults are denoted as L for large or S for small. The plots highlighted in grey are faults that should be detected by the residual on that row. The figures in white should be decoupled from the fault for that case.

A histogram shows the distribution of samples for a series of data and if the blue and orange bars do not line up it implies that the data can be distinguished from each other. For the residuals the goal is to be able to distinguish faults that the residual is sensitive to from the fault free case and faults that it should be insensitive to.

It is more important for a residual to decouple faults that it should not be sensitive to than detecting faults that it should be sensitive to. Not detecting faults can be because of the model deficiencies. Looking at the histograms in Figures 6.2 and 6.3 the distributions highlighted in white are mostly aligned, as wanted.

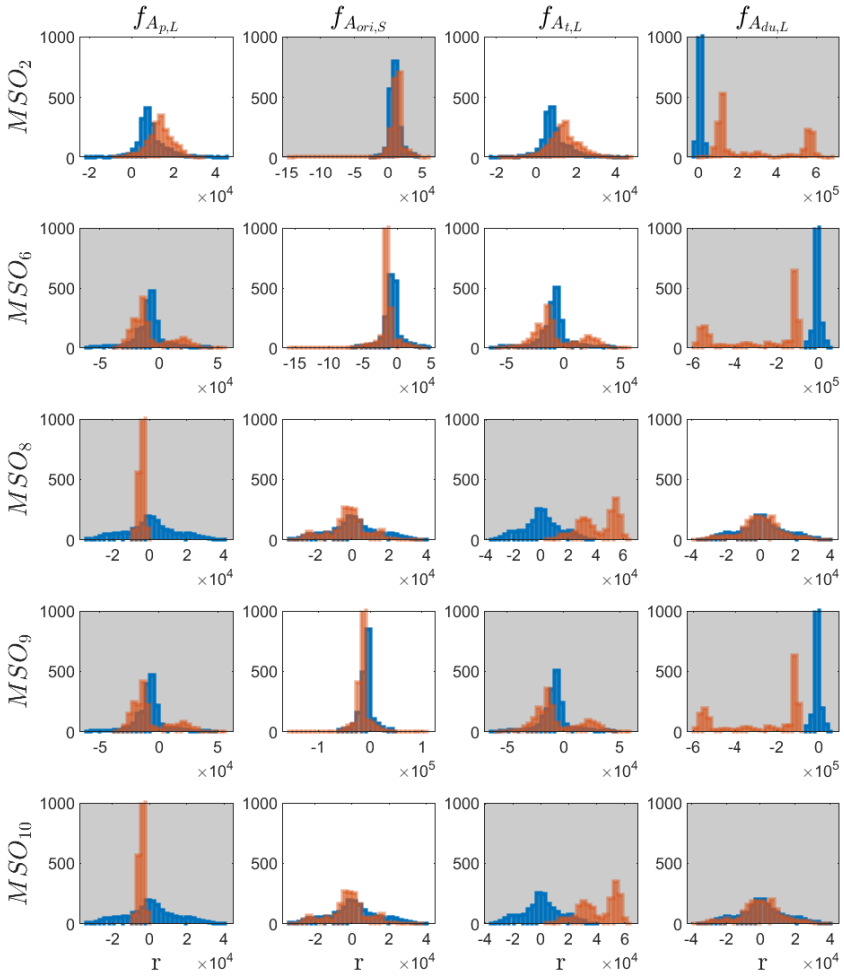



Figure 6.2: Histograms for residuals based on MSO sets 2-10. Where each row represents one MSO set and each column a fault. The blue histograms in each plot is the fault-free validation data and used in all plots for reference. And the orange histogram is the respective fault data for each MSO set. Where the x-axis shows the residual value and y-axis shows the amount of samples at that residual value.

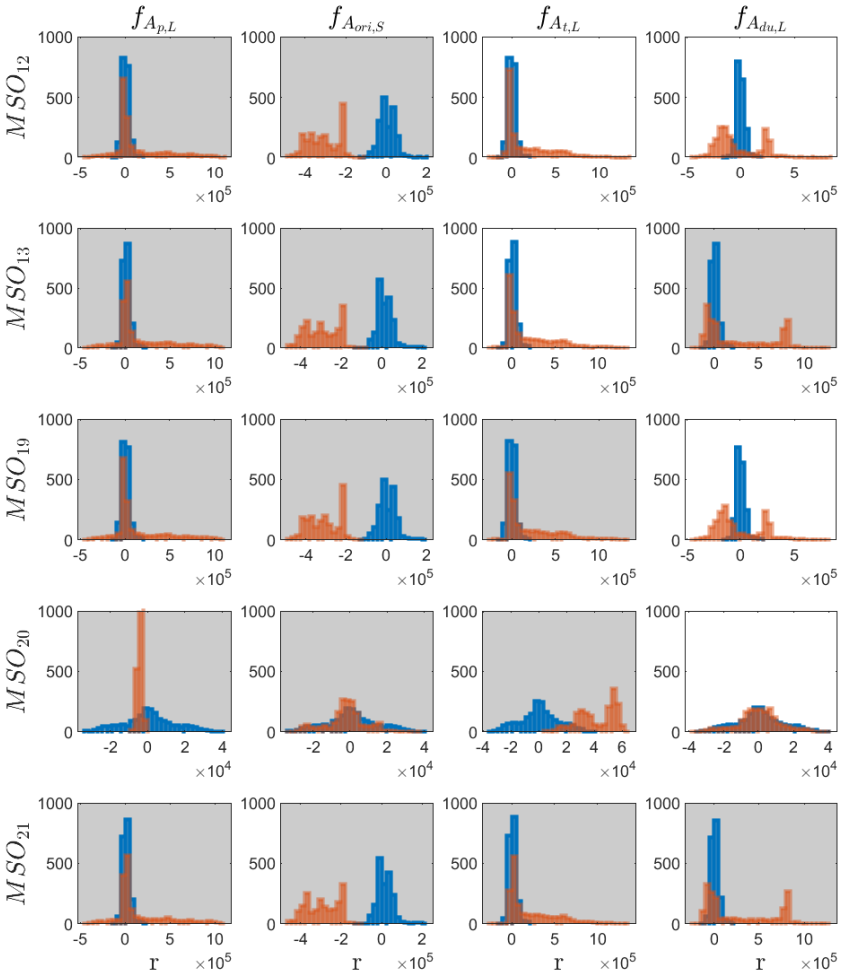



Figure 6.3: Histograms for residuals based on MSO sets 12-21. Where each row represents one MSO set and each column a fault. The blue histograms in each plot is the fault-free validation data and used in all plots for reference. And the orange histogram is the respective fault data for each MSO set. Where the x-axis shows the residual value and y-axis shows the amount of samples at that residual value.

### 6.1.3 Residuals for fault data

In Figure 6.4 the residual of $MSO_{20}$, predicting $p_{tp}$, is plotted for four different faults. Where the residuals highlighted in blue are for the no fault case and orange is for the fault case. The plots in highlighted in grey are faults that the residual should detect and the faults in the white plot should be decoupled.

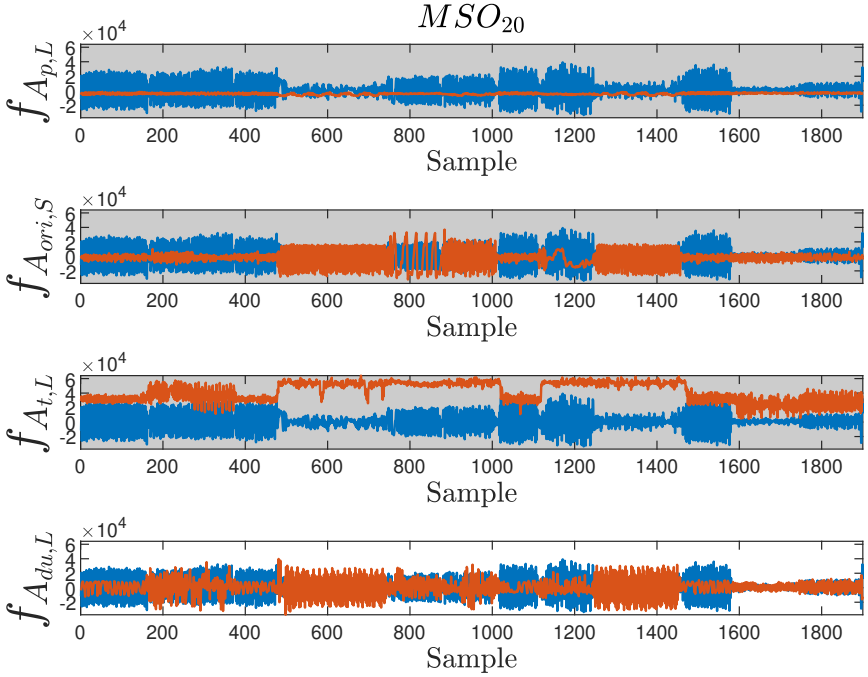
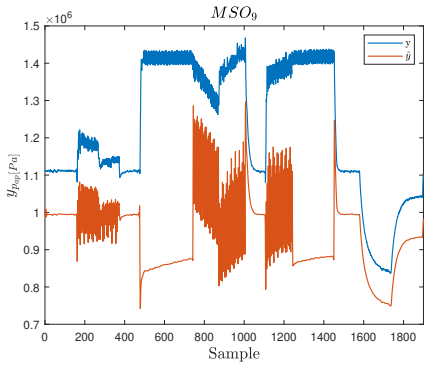
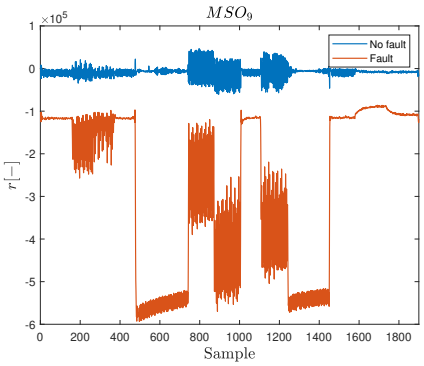



Figure 6.4: Residuals of $MSO_{20}$ where the blue line is the fault free residual and orange is the fault case seen to the left.

An example of an implemented residual is shown in Figure 6.5. $MSO_9$ models the pressure after the pump $p_{ap}$, the measured and predicted values for this MSO for the large dosing unit filter fault is shown in Figure 6.5a and the corresponding residual and no fault residual in Figure 6.5b. This fault is clearly distinguishable in the residual from the no fault case.




(a) Model prediction in orange and measurement data in blue.




(b) Residual plotted for the fault in orange and for the fault free mode in blue.

Figure 6.5: Predicted and measured values for and the corresponding residual for $MSO_9$ for a large fault in the dosing unit filter, $f_{A_{DU},L}$.

### 6.1.4 CUSUM

To further evaluate the performance of the residuals CUSUM tests are used to see if the individual residuals can isolate and detect the faults as they are theoretically supposed to according to the FSM in Figure 5.6. Figure 6.6 shows the residuals for $MSO_9$ in the left column and the corresponding CUSUM tests to the right. $MSO_9$ will not be able to detect or isolate correctly because at sample ~1000 there are large oscillations because of a large dose. This results in a large residual even in the no fault case decreasing the performance of the MSO. Just as for the histograms faults in the CUSUM tests that should be detected are highlighted in grey.

Because the model does not include the oscillations during doses there is an increase in amplitude depending on the size of the requested dose. This information can be used to create an adaptive threshold that is modeled to fit the residual by increasing or decreasing depending on the size of dose. An adaptive threshold would increase the performance of the residuals enabling better diagnosis by improving detection rate without increasing false alarm rate. This is possible by reducing the threshold when the residual is smaller during no fault i.e. smaller doses.

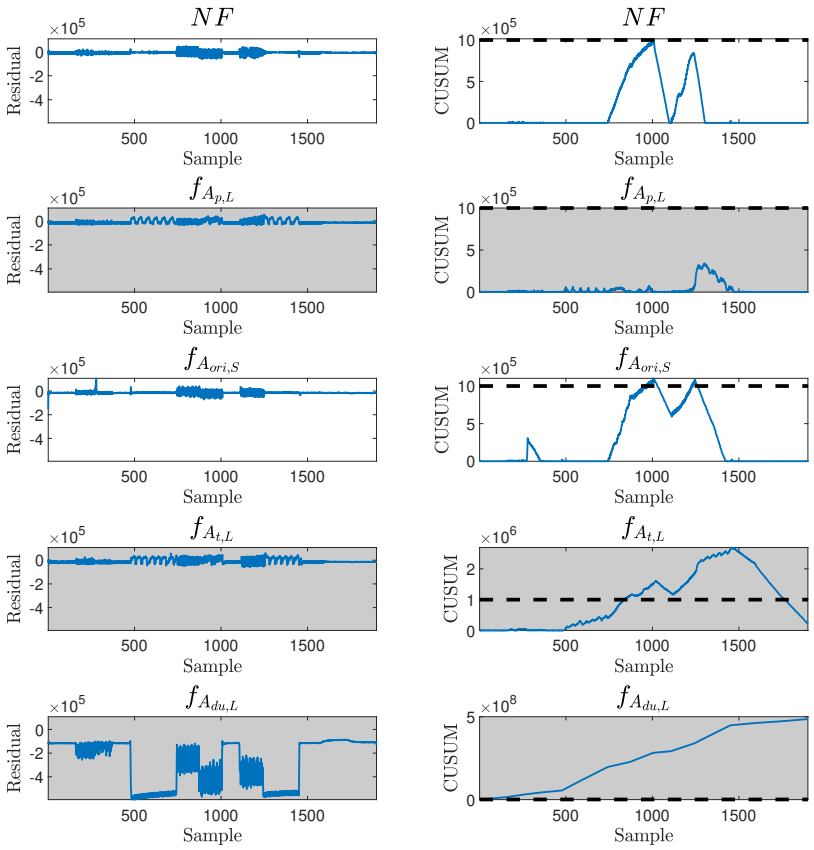



Figure 6.6: Residuals for $MSO_9$ in the plots on the left column and the corresponding CUSUM plots in the right column of plots. Where each row is a different data set.

Since some faults are not detectable for $MSO_9$ because of the high values of the residual during a large dose as seen Figure 6.6 and an adaptive threshold has not been implemented. Taking a part of the data that contains a maximal dose, detecability and isolability should be improved. Because during a maximal dose it is the largest flow the system can achieve and it contains no oscillations. Looking at Figure 6.7, a CUSUM test for a maximal dose, the MSO:s CUSUM test passes the threshold for the faults that it should detect and not for the no fault data and the fault that it should not be sensitive to.

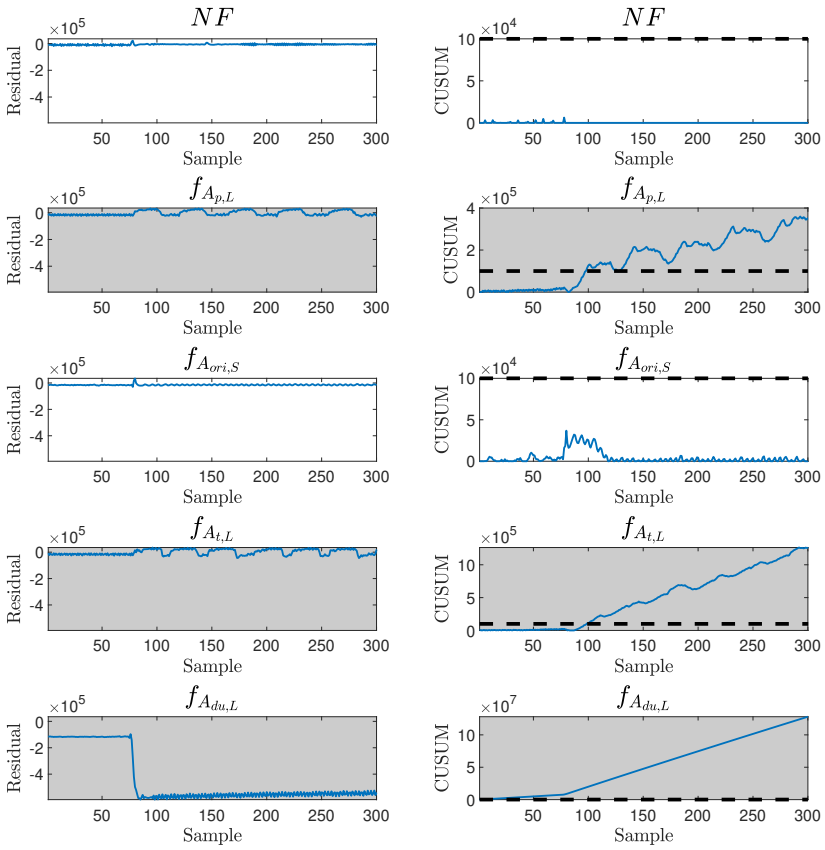



Figure 6.7: Residuals for MSO 9 in the plots on the left column and the corresponding CUSUM plots in the right column of plots, where each row is a different data set. The residuals are evaluated on data from maximal dosing.

## 6.2 Greybox RNN residuals

This section presents the results for the grey-box RNN-based residuals.

### 6.2.1 Validation

To validate the models, the model prediction and measurement data from the validation cycle were plotted in the same plot for each MSO set. In Figure 6.8 $MSO_2$, $MSO_6$ and $MSO_{10}$ are presented. $MSO_2$ models $p_{DU}$, $MSO_6$ models $p_{ap}$ and $MSO_{10}$ models $p_{tp}$. The rest of the MSO sets that are not shown have similar performance as the ones shown here. Overall, the model predictions follows the measured signals quite well. $MSO_{10}$ is only able to capture the mean value of the pressure. The variations in the residual noise is due to the difficulty of capturing the oscillations from the PWM.

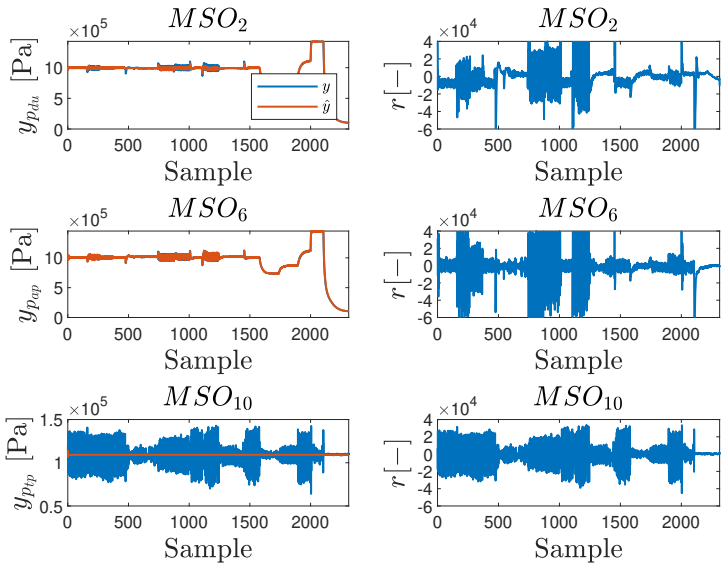



Figure 6.8: Model prediction and measurements using validation data for three different MSO sets. Each MSO set in the figure models a different state from each other.

### 6.2.2 Histograms

Histograms for all faults were generated for the neural network-based residuals in the same way as for the model-based residuals and can be seen in Figure 6.9 and 6.10. The blue bars are the residuals from the no-fault case and the orange is for the fault case. Faults are denoted as L for large or S for small. The plots highlighted in grey are faults that should be detected by the residual on that row. As described in Subsection 6.1.2 the goal is that the distribution of the residual during the fault-free case can be distinguished from the fault that it should be sensitive to.

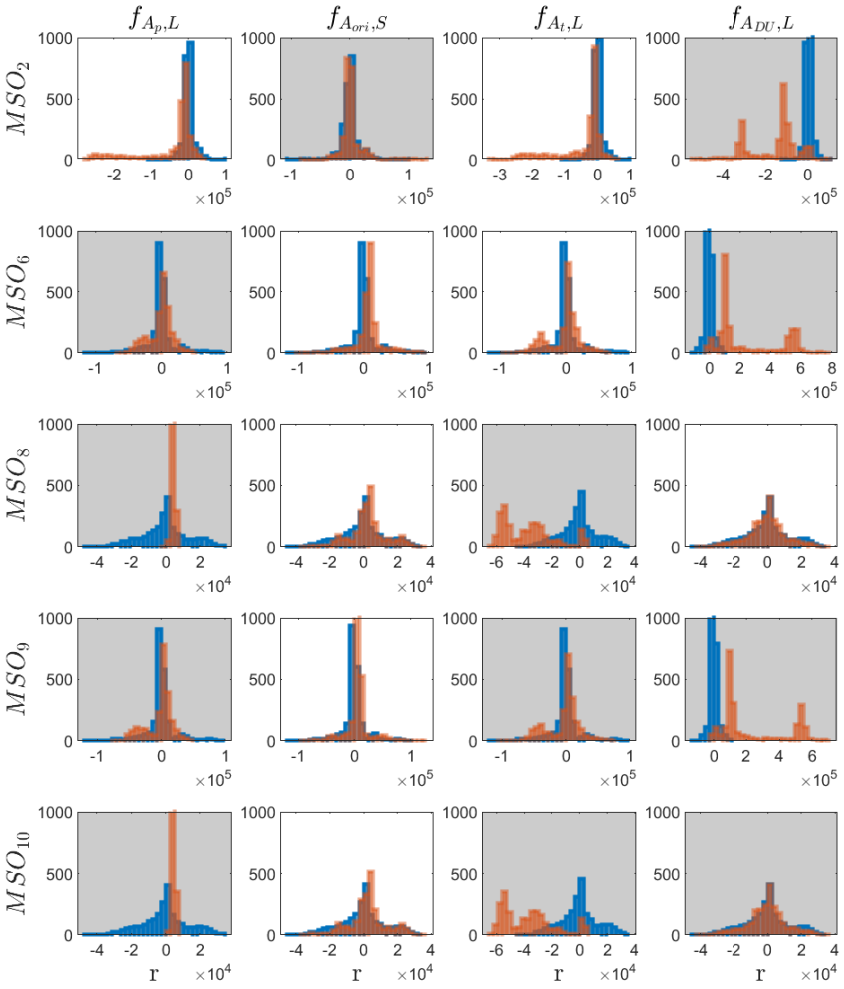



Figure 6.9: Histograms for $MSO_{2-10}$ with residual data from fault free conditions in blue and from faults in orange. Faults that should be detected are highlighted in grey.

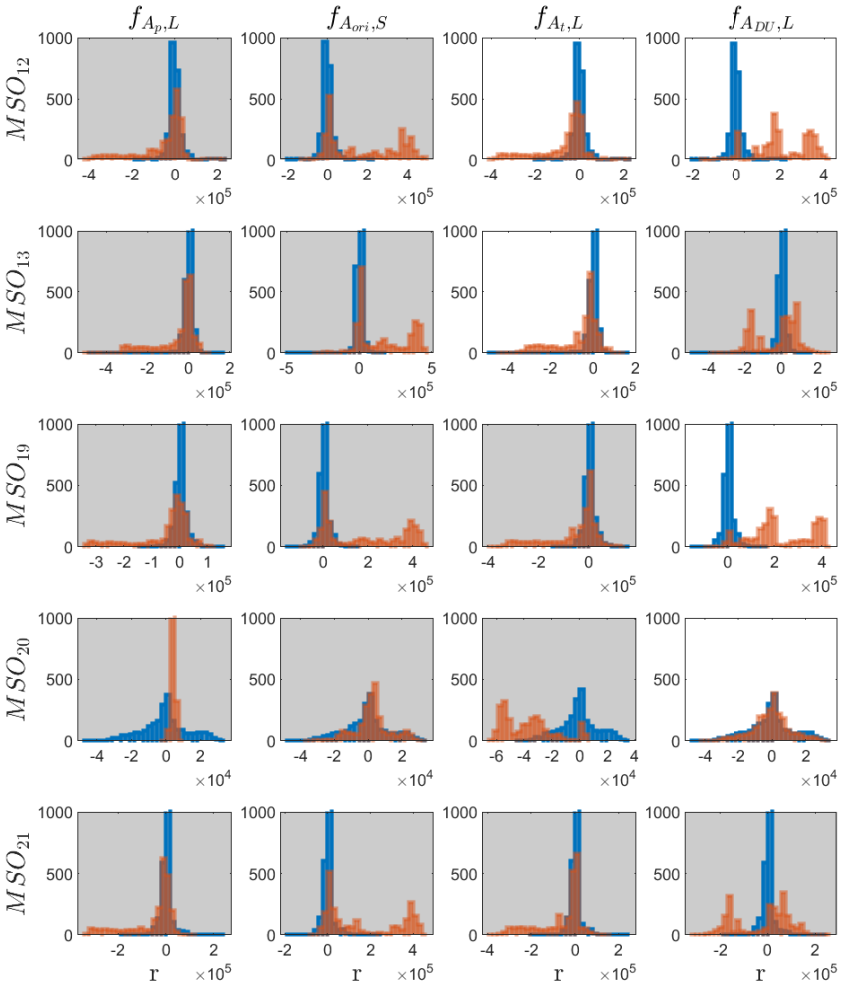



Figure 6.10: Histograms for $MSO_{12-21}$ with residual data from fault free conditions in blue and from faults in orange. Faults that should be detected are highlighted in grey.

### 6.2.3 Residuals for fault data

In Figure 6.11 the residual of $MSO_{20}$ is plotted for four different faults. The plots in grey are faults that the residual should detect and the fault in the white plot should be decoupled.

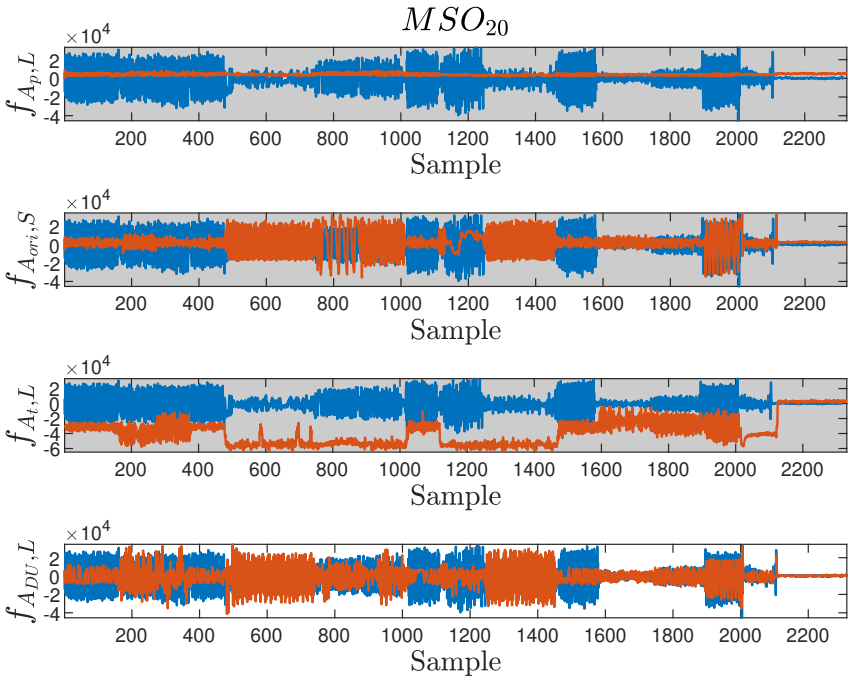
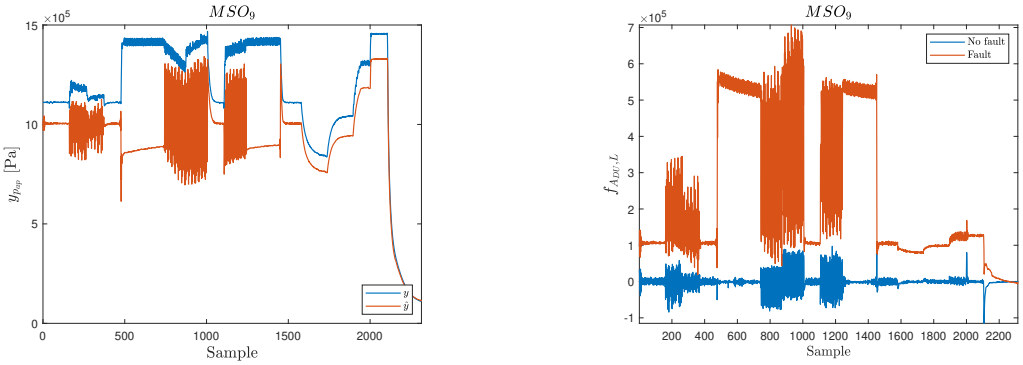



Figure 6.11: Residual values for different fault modes plotted in orange and the nominal mode plotted in blue.

Below in Figure 6.12 the model prediction and residual for $MSO_9$ is plotted for a large fault in the dosing unit filter, $f_{A_{DU},L}$. There is a clear distinction between the residual at nominal and faulty mode.




(a) Model prediction in orange and measurement data in blue.

(b) Residual plotted for the fault in orange and for the fault free mode in blue.

Figure 6.12: $MSO_9$ for a large fault in the dosing unit filter, $f_{A_{DU},L}$.

### 6.2.4 CUSUM

A CUSUM test was implemented for $MSO_9$. In Figure 6.13, the residual signal and the CUSUM test are shown for the full data sets. The faults that should be detected are highlighted in grey. The threshold is plotted as a dashed black line. The CUSUM test shows that it is not possible to achieve the wanted fault isolability for the entire data set. Because of the high residual amplitude during large doses, it is not possible to decouple the orifice fault while simultaneously detect the pump and tank filter faults.

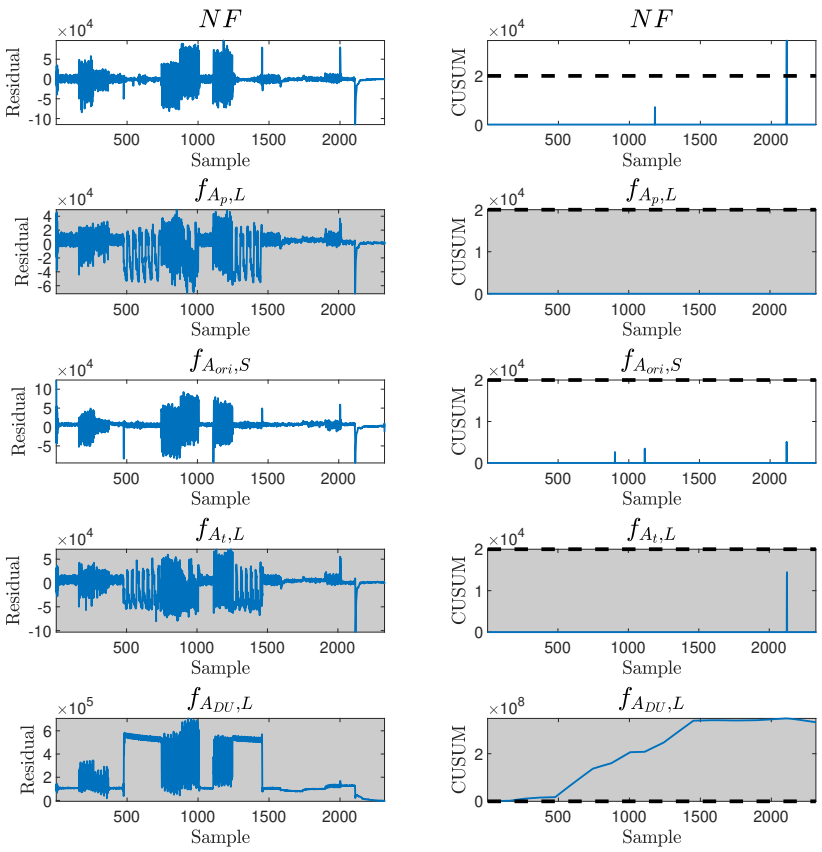



Figure 6.13: Residuals for $MSO_9$ in the plots on the left column and the corresponding CUSUM plots in the right column of plots, where each row is a different data set.

However, if the CUSUM test is used during maximum dosing the faults can be detected as wanted. In Figure 6.14 the same plot as in Figure 6.13 is shown but only for the samples between 500 and 700 where there is maximum dose flow.

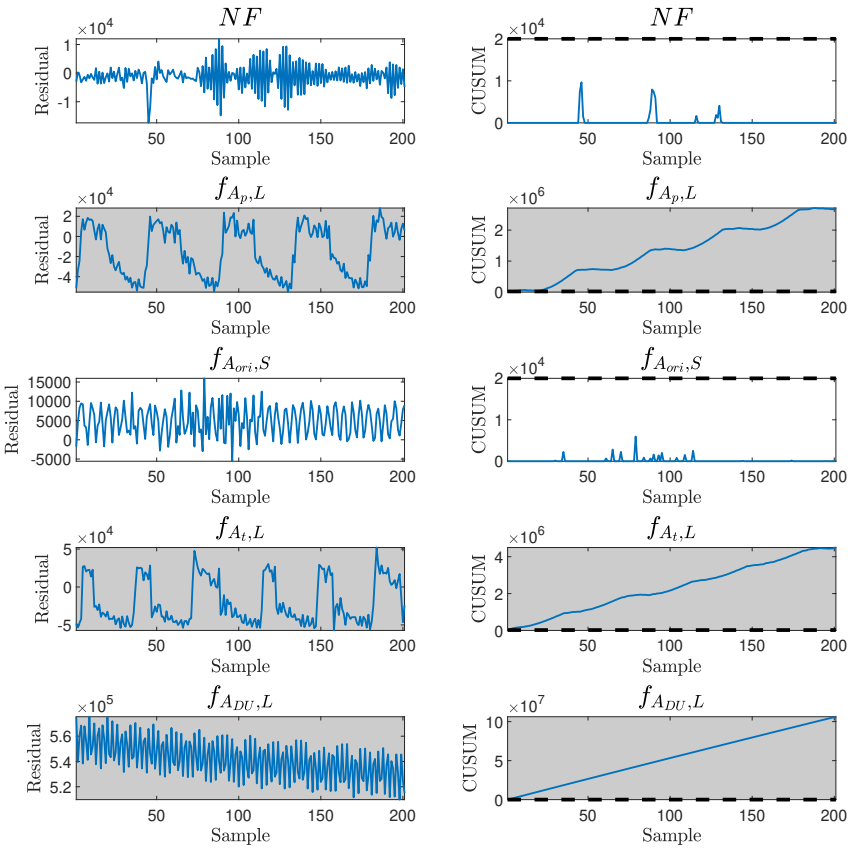



Figure 6.14: Residuals for $MSO_9$ in the plots on the left column and the corresponding CUSUM plots in the right column of plots, where each row is a different data set. The residuals are evaluated on data from maximal dosing.

### 6.2.5 Training on fault data

Figure 6.16a shows the model prediction (in orange) and the measured dosing unit pressure (in blue), $y_{p_{DU}}$, for $MSO_2$ when trained on the nominal training data. The measured data and the validation data used for simulating the model are data collected for a fault in the pump filter $f_{A_p}$.

According to the FSM in Figure 5.6 it can be seen that the $MSO_2$ is not supposed to be sensitive to $f_{A_p}$. From Figure 6.16a it looks like the model is sensitive to the fault when the pressure drops at approximately sample 500 and 1400. This is due to this operating point not being included in the training data as can be seen in Figure 6.15. In the figure the input signals $y_{p_{ap}}$ and the requested dose are plotted for the training data set and for the samples between 500 and 750 in the $f_{A_p,S}$ data set.

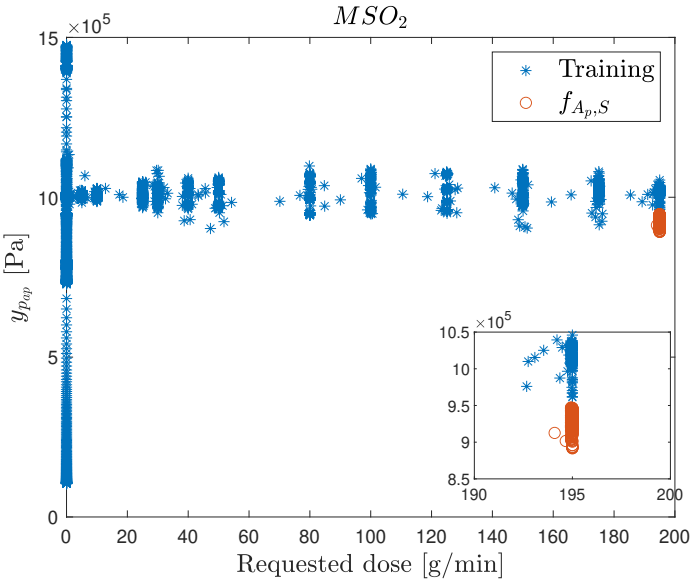
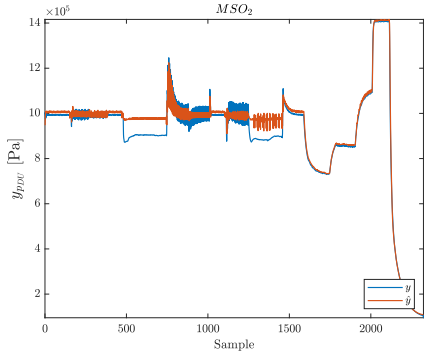
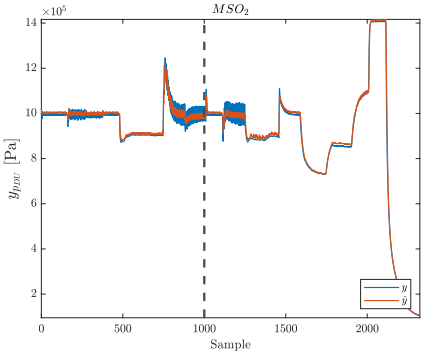



Figure 6.15: Operating points for the training data in blue and for the fault data in orange.

To show that the residual will decouple the fault as expected given representative training data the first 1000 samples of the fault data were included in a new set of training data. This is shown in Figure 6.16b where $MSO_2$ has been trained using the new training data set. The data from sample zero up to the dashed line were included in the training data. The data after the dashed line was used as validation data.




(a) $MSO_2$ trained on the original training data set.




(b) $MSO_2$ trained on training data which includes part of the fault data set.

Figure 6.16: $MSO_2$ simulated for a fault in the pump filter, $f_{A_p}$.

## 6.3 Comparison

To show the differences in model accuracy between the NN-based and the model-based residuals, $MSO_{21}$ is plotted for both methods. $MSO_{21}$ models a large part of the system, which the model-based method has issues modeling, due to the difficulty of finding parameters.

In Figure 6.17 the same model prediction is plotted for the model-based method in orange and the NN-based method in yellow. The measured pressure is shown in blue. It can be seen that the NN-based model prediction follows the measured value better than the model-based one.

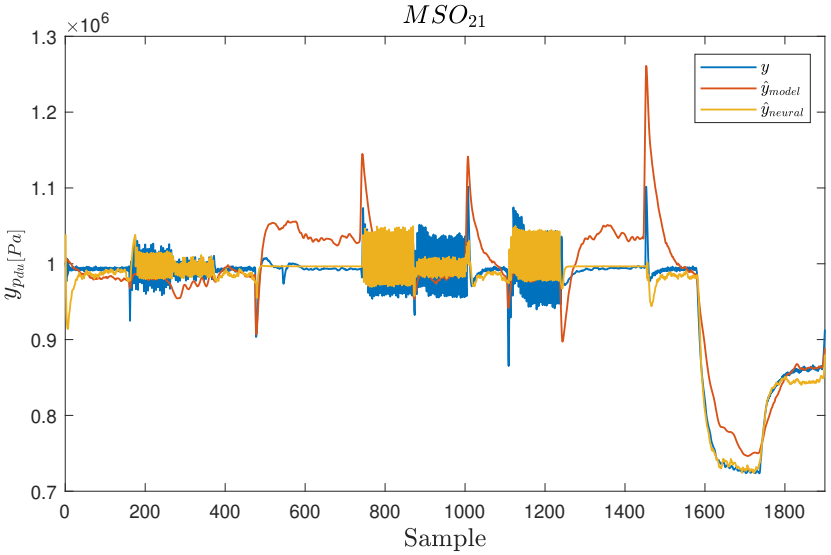



Figure 6.17: Predicted dosing unit pressure estimated by MSO 21 for both model based and neural network based MSO compared to the measured value.

# Chapter 7

# Discussion

This chapter is a discussion of the results and method of this thesis. The results, methodology and the future potential of both methods will be analyzed and discussed.

## 7.1 Data

It is important that the training data is representative of the nominal system behavior. The training data cycle was chosen to somewhat mimic how the system would behave when the truck is on the road. The cycle starts with dosing sequences of different sizes and duration for a constant system pressure and ends with steps in the pressure without dosing. The pressure steps were included to capture the system dynamics at different pressure levels.

The validation cycle was created in a similar way to the training data cycle, but with different dosing amounts and duration, as well as different pressure levels for the pressure steps. It is essential to validate the trained model on a different data set than the training data.

It is apparent that in some cases such as seen in Figure 6.16 that there are working points in the system not captured by the training data. The problem was that dosing for pressures other than the 10 bar the system was set at was not possible to do for this thesis because of restrictions in method used for control input. Therefore there are some system dynamics not captured by the training data or the neural networks trained on the data.

## 7.2 Data collection

Since this thesis work has been performed as an iterative process the data collection is also done similarly. The goal was to collect as many relevant signals and data as possible during each measurement and sort it out later.

When using external sensors, a different data collection script was used which required the data to be synchronized with the built-in sensors. This was expected since the data collection system used for the external sensors is currently under development and a demo application was used. If the data collection could have been done in a single script, it would have removed the need for synchronization between data. This would have both saved time and removed the risk of not synchronizing correctly.

There was a limit to which faults that were possible to simulate or recreate due to what equipment was available to use and due to accessibility of the components on the truck. Faults that were implemented were chosen based on what faults would be most prevalent in the system and faults that was easily implementable in both a test bench and on a truck. This

resulted in only restriction faults. Sensor faults could also have been simulated but there were enough real faults for testing of the diagnosis systems.

The benefit of using a valve as a simulation of a clogged filter is that it is variable in the size of the restriction. This allowed for easily tuning the restriction during implementation. The downside is that it is another component and affects the size of the pressurized volume and length of the hose.

The sampling frequency of the internal sensors was limited to the speed of the CAN bus and the external signals were matched to be able to be used simultaneously. Optimally the sampling frequency would have been higher than the system dynamics allowing for more accurate modeling and diagnosis. The low sampling frequency used for the system measurements resulted in the dynamics between all the pressures in the system not being captured, therefore the pressure can be estimated with static connections.

The available sensors were pressure sensors. It would have been interesting to use flow sensors since the structural model gave indications that the isolability properties could have been improved by measuring the flow in at least one position.

## 7.3 Model validation

The physical-based models were able to model the overall dynamics of the system but there are working points that the model that output a large amplitude in the residuals. The system has non-linearities coupled to the pressure, therefore the system was parameterized around the nominal working point of the system. Because of that, the model works best around nominal values and it is the reason why the dynamics of the system are not correct for large steps in pressure away from the nominal working point. Another problem in the model is the modeled dosage flow. There is a noticeable non-linearity in the dosage flow when the dosage valve goes from closed or partly open to fully open and from fully open back to closed or partly closed. This is not caught by the implemented model and causes the model to have large spikes and dips in pressure during maximal doses. This can be seen in Figure 5.10 where there is a maximal dose from 540-590 seconds.

There are two dosage flow models tested, one using the requested dose flow from the ECU and the orifice equation and the other model where the requested dose is first converted to a PWM, still using the orifice equation. Using the requested dose model, the system models the average pressure well but does not include any oscillations in the pressure induced by the valve being controlled by a PWM. The dosage model including the PWM models the oscillations much better but is worse in modeling different sizes in doses. Therefore ultimately the model without PWM was chosen even though it does not model the oscillations in pressure.

For diagnostic purposes, it is not necessary to estimate the PWM signal for faults where the change in the mean of the signal is large enough. For smaller faults where the mean might not shift as much, and where the variance changes it is necessary with a more accurate PWM model. It is also possible to create an adaptive threshold based on the requested dose using a modeled variance for a given dosing request and use that in combination with the residual estimating the mean in order to decrease or increase the threshold of the residual depending on the size of the variance.

Two models of the system were developed, a one-state model and a four-state model. The reason for creating the four-state model was that there were problems auto-generating the residuals for the one-state model. The one-state model has variables in a square root which leads to multiple solutions, in the four-state model these variables are states so it does not have multiple solutions. Looking at Figure 5.10 the performance of the one-state and four

state models is quite similar. There is a small difference in gain of the models but this could also be due to parameter estimation not being the same for both models. Using models with fewer states requires fewer parameters to be identified which decreases the simulation time. A model with fewer states is also easier to troubleshoot.

## 7.4 Model-based residuals

Overall, the results show the potential benefits of using model-based approaches when designing a diagnosis system. The CUSUM tests for the model-based residuals in Figures 6.6 and 6.7 show that there are residuals that work as expected and can isolate and detect the implemented faults. When only using data from maximal dosing the wanted isolation for this particular residual can be achieved. Using an adaptive threshold would be ideal since there is a known connection between residual size and dose, it would increase the diagnosis performance by avoiding false alarms and missed detections during certain working points.

Looking at the model-based residuals as shown in Figure 6.1, the residuals modeling the pressure between the filters, $p_{tp}$, does not capture any dynamics. This is because there is not a clear relation between $p_{tp}$ and the other available signals in the system. Optimizing the parameters resulted in a close to a straight line as a predicted value. Looking at Figure 6.4, there are still cases where the fault causes the residual to increase significantly which are detectable with this model. A problem is that faults such as a blockage in the pump filter cause the variance in the pressure to decrease causing a decrease in the residual during a fault. However, there are methods to detect faults that have this appearance, such as in the report [25] where the authors propose a way of detecting variations in the residual distribution. So even though the model does not capture the dynamics of $p_{tp}$, it is still possible to differentiate the residual during no-fault from fault data.

Residuals modeling smaller parts of the system such as $MSO_2$ and $MSO_6$ as seen in Figure 6.1 captures the oscillations in the pressure and have overall increased performance. This is because the smaller MSO:s have measured pressure signals as inputs, while the larger MSO:s model a larger part of the system without that information. As seen for the model of the whole system in Figure 5.10 the oscillations are not captured because it models the average flow.

Looking at the histograms for the model-based residuals in Figure 6.2 and 6.3, residuals modeling a larger part of the system, $MSO_{12}$, $MSO_{13}$, $MSO_{19}$ and $MSO_{21}$ appear to have a similar distribution in most cases. This is most likely because of the information available in the input signals is not different enough or does not affect the modeled output enough between the residuals. For instance, two of the residuals both modeling $p_{du}$ but one has $p_{bp}$ as modeled state and the other has the sensor input. The residual with the state should be sensitive to faults affecting $p_{bp}$, but the problem is that the pressure before the pump does not affect the flow enough for it to be noticeable. Therefore the distributions are similar.

## 7.5 Greybox RNN residuals

The validation of the RNN-based residuals showed that the neural networks are good at capturing the overall dynamics of the model. However, for the residuals predicting the pressure before the pump, it was not capable of capturing the oscillations in the pressure, as can be seen for the residual based on $MSO_{10}$ in Figure 6.8. For the residuals that predicted $p_{DU}$ or $p_{ap}$ the oscillations could be captured to a varying degree. It was harder for the residual generators that did not have the PWM signal as input to model the oscillations.

The RNN-based residuals that included a larger part of the model captured the system behavior better than their model-based counterparts. This can be seen in Figure 6.17. The reasons for the model-based residuals performing worse are mainly the difficulty of modeling the non-linearities as well as finding appropriate parameters that worked for the entire model.

The diagnosis performance varies between the residuals. The residual value and the corresponding CUSUM tests for different faults are shown in Figure 6.13 for $MSO_9$. In this figure, the residuals are shown for the entire data set. The orifice fault is decoupled as wanted for this particular residual. The residual for the dosing unit filter fault, $f_{A_{DU},L}$, is detected as expected. In Figure 6.12 it is shown that the fault is distinguishable for the nominal mode. The faults $f_{A_p,L}$ and $f_{A_t,L}$ can not be detected for the full data set. These faults are filter faults in the pump and tank filter and are not as easily detected as they do not affect the system behavior as much as the dosing unit filter fault.

During maximal dose, the faults are more easily detected, which is expected. In Figure 6.14 the same residual is evaluated on the same faults but only for operating points during maximal dosing. With the CUSUM test the faults are detected or decoupled as wanted according to the FSM in Figure 5.6.

For some MSO sets the faults can not be decoupled as wanted. The reasons for this are mainly the lack of representative training data. The neural network-based residuals can not decouple faults if there are new operating points that they have not trained on. Some careful collection of additional training data could address this as illustrated when training on faulty data in Section 6.2.5.

Like the model-based residuals the RNN-based residuals modeling $p_{tp}$ do not capture any dynamics. The reasons for this are the same as for the model-based residuals. In Figure 6.11 it can be seen that the variance of the pressure during the faults are lower than for the nominal mode. The fault ($f_{A_{DU},L}$) that should be decoupled have higher variance than the non-faulty mode during maximum dose. Therefore using only operating points from maximal dose will not help with the diagnosis performance for this particular residual. There are other approaches to address this as mentioned earlier for the model based residuals also seen in [25].

### 7.5.1 Training of Neural Networks

The performance of the neural network-based residuals can vary when training different instances of the same NN model because the initial weights for each training session are randomized. In Figure 5.13 $MSO_{13}$ was trained multiple times. It can be seen that for some training instances the performance varies. Especially for the samples from 1500 and onward. Some instances perform quite badly during these samples. It can be beneficial to train the same model several times. One technique is to use ensemble averaging where multiple instances of a model are fitted to the training data and then the final output is chosen as the average of the predictions. By doing this the variance can be reduced at the expense of higher computational cost.

The choice of the learning rate can make a big difference. If it is too big the gradient descent might not be able to converge and if it is too small it will take too long to converge. Therefore it is important to find a good trade-off.

The number of epochs that the neural networks were trained for also made a difference. Even though the loss did not change much between 1000 and 1600 epochs the way that the neural networks captured the oscillations in the pressure during dosing varied.

How well the neural network-based residuals captured the oscillations also depended on if the requested dose or PWM was used as input. When using the requested dose, the neural networks could not capture the oscillations as well as when using the PWM signal. This was the same situation as for the model-based residuals.

## 7.6 Model-based diagnosis method

The model-based method can be very useful when designing diagnosis systems for a system where there already exists parameterized models. It is a systematic way of designing diagnosis systems and can have good diagnosis performance given an accurate model. Compared to ad hoc methods the model-based method can detect previously unknown faults and specify which equation (representing a component) is faulty.

Model-based approaches such as structural analysis and CUSUM tests can be very useful when designing and implementing diagnosis systems. Structural analysis can be done early in the design process and can give indications on what diagnosis performance that can be expected.

A benefit to the model-based method is that it can fully take advantage of possible residuals if measured signals can be differentiated. Though this is often not the case for measured signals since they are often noisy.

The first step in model-based diagnosis is the modeling of the system. As expected this process has taken a large amount of time to implement and there are still model improvements that can be done. It is hard to find parameters that describe the system adequately. The use of grey-box modeling simplifies the parameter estimation process so that not each parameter must be validated one by one through tests. But even with grey-box modeling, it took a large amount of time to complete.

## 7.7 Data-driven diagnosis method

The neural network-based methods modeling process was not time-consuming, and still, the predicted output of the NN captured the dynamics of the system better than the physical model-based version for the operating points included in the training data. This indicates that the neural network-based residuals can have similar performance as the model-based residuals but with a much simpler structural model. The important part is to capture which variables are connected. This does not necessarily need to be done by deriving physical-based equations from theory, using standard components from modeling tools like Modelica or Simscape could be used to find a structural model.

The diagnostic performance of the neural network-based residuals is heavily dependent on the training data. The training data needs to capture the nominal behavior for each component for a wide range of operating points that may occur when a fault is implemented.

It can be difficult to know beforehand which operating points that need to be included in the training data. New operating points may appear in fault data and a residual that should decouple that particular fault may not do so because of the new operating points not being covered by the training data. Therefore it can be relevant to use data from faults, if available, to train residual generators. This can be done by extending the training data set with the fault data and retraining the residual.

# Chapter 8

# Conclusion and Future work

## 8.1 Conclusion

This thesis work has shown that when designing a model-based diagnosis system there is a need for an accurate model. Even though there were similar models available to the model developed for this work it took time to adjust the models and find appropriate parameters. However, even though the model is not perfect, it is still possible to detect and isolate faults in the system. There are methods such as CUSUM tests and adaptive thresholds that can deal with model inaccuracies. Model-based diagnosis can be very effective, it gives the engineer a systematic and efficient method of developing a diagnosis system. Model-based approaches such as structural analysis can be an efficient way of analyzing large-scale systems early in the design process.

The grey-box RNN residuals are shown to have similar performance as the model-based residuals given that the working points are included in the training data. It is also shown how important it is to have representative training data for the NN models. Given a more sophisticated data collection method, the performance of the grey-box RNN residuals could have been improved.

It can be difficult to know which operating points that need to be included in the training data. Some operating points may only appear during faults. Therefore it can be beneficial to use data from faults to improve fault classification.

If given representative training data the grey-box RNN method can be a more efficient way of designing diagnosis systems compared to the model-based method. Parameterizing the residuals is automatically done by training the neural networks and since the neural network only requires a structural model, a simpler model can be used.

Both methods can be applied to different systems and SCR systems using different component setups. Since the diagnosis systems developed with these methods are divided into submodels it is possible to parameterize the models that use different components and reuse similar ones. This is important when applying the methods on different types of trucks that might use slightly different pumps or dosage units for instance. Compared to ad hoc methods where the diagnosis might be performed by looking at a certain signal to determine if a component is faulty or not, the model-based and the hybrid method can be transferred and be valid for different component setups. It becomes more modular by generating residuals from models, whether method they are derived by, the diagnosis can be performed regardless of the input signal.

The model-based method of developing a diagnosis system can be a efficient way to utilize existing mathematical models that have been developed for simulation and control applications. The grey-box RNN method can be an even more efficient technique to develop a diagnosis system for a system where no accurate model is available.

## 8.2 Future work

In this section, some ideas for future work are presented.

### 8.2.1 Adaptive thresholds

To improve the diagnosis performance of the NN-based residuals an idea is to, in addition to the current models, train NN:s that model the variance of the residuals during nominal conditions. Using these models an adaptive threshold can be implemented increasing diagnosis performance.

### 8.2.2 Implementation

To further investigate if the methods presented in this report are modular, an idea of future work is to apply them to a different system. It would also be interesting to develop a workshop diagnosis test or an on-board system based on the work in this thesis to see how Scania can utilize these methods in their products.

### 8.2.3 Isolation of multiple faults

In this thesis only single faults have been considered, an idea for future development is to analyze these methods for multiple faults that occur simultaneously.

### 8.2.4 Model improvements

There are model improvements that can be done. The model for the dosage flow can be improved. One idea is to use the same equations but to have a more accurate PWM signal. If data from the actual PWM signal could be collected it would improve the performance.

As mentioned earlier in the thesis, the accuracy of the model depends on the size of the dose. Developing an adaptive threshold based on residual variance can be an alternative to improving the model for increased diagnosis performance. This can be accomplished using simple linear regression or more advanced models.

### 8.2.5 Training data

One improvement that can be done for this particular diagnosis system of the UDS is to collect better training data. By using a more sophisticated data collection method more representative training data can be used when training the RNN-based residuals. It would also be interesting to evaluate both methods on data from real fault scenarios to evaluate the diagnosis performance.

A topic that would be interesting to investigate in future work is how robust the RNN-models are. The question is if the same trained model can be used for a system with slightly different components, or if they need to be retrained for each new component combination.

# Bibliography


[1] M. Nyberg and E. Frisk. *Model Based Diagnosis of Technical Processes*. Linköping, Sweden: Linköping University, 2020.

[2] L. Dong, S. Liu, and H. Zhang. "A method of anomaly detection and fault diagnosis with online adaptive learning under small training samples." In: *Pattern Recognition* 64 (2017), pp. 374–385. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2016.11.026.

[3] D. Jung, K. Y. Ng, E. Frisk, and M. Krysander. "Combining model-based diagnosis and data-driven anomaly classifiers for fault isolation." In: *Control Engineering Practice* 80 (2018), pp. 146–156. ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2018.08.013.

[4] V. Johnson Timothy. "Review of Diesel Emissions and Control." In: *SAE International Journal of Fuels and Lubricants* (2010). ISSN: 19463952. URL: https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edsjsr&AN=edsjsr.26272638&lang=sv&site=eds-live&scope=site.

[5] E. Frisk, A. Bregon, J. Aslund, M. Krysander, B. Pulido, and G. Biswas. "Diagnosability Analysis Considering Causal Interpretations for Differential Constraints." In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 42.5 (2012), pp. 1216–1229. DOI: 10.1109/TSMCA.2012.2189877.

[6] E. Frisk, M. Krysander, and D. Jung. "A Toolbox for Analysis and Design of Model Based Diagnosis Systems for Large Scale Models." In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 3287–3293. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2017.08.504.

[7] D. Jung. *Residual Generation Using Physically-Based Grey-Box Recurrent Neural Networks For Engine Fault Diagnosis*. 2020. arXiv: 2008.04644 [eess.SP].

[8] D. Jung. *Isolation and Localization of Unknown Faults Using Neural Network-Based Residuals*. 2019. arXiv: 1910.05626 [eess.SP].

[9] B. Pulido, J. M. Zamarreño, A. Merino, and A. Bregon. "State space neural networks and model-decomposition methods for fault diagnosis of complex industrial systems." In: *Engineering Applications of Artificial Intelligence* 79 (2019), pp. 67–86. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2018.12.007.

[10] R. Rahimilarki and Z. Gao. "Grey-box Model Identification and Fault Detection of Wind Turbines Using Artificial Neural Networks." In: *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*. 2018, pp. 647–652. DOI: 10.1109/INDIN.2018.8471943.

[11] R. Hofmann, V. Halmschlager, M. Koller, G. Scharinger-Urschitz, F. Birkelbach, and H. Walter. "Comparison of a physical and a data-driven model of a Packed Bed Regenerator for industrial applications." In: *Journal of Energy Storage* 23 (2019), pp. 558–578. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2019.04.015.

[12] P. Polverino, E. Frisk, D. Jung, M. Krysander, and C. Pianese. "Model-based diagnosis through Structural Analysis and Causal Computation for automotive Polymer Electrolyte Membrane Fuel Cell systems." In: *Journal of Power Sources* 357 (2017), pp. 26–40. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2017.04.089.

[13] M. Krysander and E. Frisk. "Sensor Placement for Fault Diagnosis." In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 38.6 (2008), pp. 1398–1410. DOI: 10.1109/TSMCA.2008.2003968.

[14] A. Rosich, E. Frisk, J. Åslund, R. Sarrate, and F. Nejjari. "Sensor Placement for Fault Diagnosis Based On Causal Computations." In: *IFAC Proceedings Volumes* 42.8 (2009). 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, pp. 402–407. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20090630-4-ES-2003.00067.

[15] Y. Cheng, M. D'Arpino, and G. Rizzoni. *Structural Analysis for Fault Diagnosis and Sensor Placement in Battery Packs*. 2020. arXiv: 2008.10533 [eess.SY].

[16] M. Holmgren. *Modelling Pressure in the Exhaust After Treatment Dosing System : Design and Tolerance Analysis of a State Space Model*. 2018. URL: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1268473&dswid=2180.

[17] C. Svärd, M. Nyberg, and E. Frisk. "Realizability Constrained Selection of Residual Generators for Fault Diagnosis With an Automotive Engine Application." In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 43.6 (2013), pp. 1354–1369. DOI: 10.1109/TSMC.2013.2258906.

[18] A. L. Dulmage and N. S. Mendelsohn. "Coverings of Bipartite Graphs." In: *Canadian Journal of Mathematics* 10 (1958), pp. 517–534. DOI: 10.4153/CJM-1958-052-0.

[19] M. Krysander, J. Åslund, and M. Nyberg. "An Efficient Algorithm for Finding Minimal Overconstrained Subsystems for Model-Based Diagnosis." In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 38.1 (2008), pp. 197–206. DOI: 10.1109/TSMCA.2007.909555.

[20] C. C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Cham: Springer International Publishing, 2018. ISBN: 978-3-319-94463-0. DOI: 10.1007/978-3-319-94463-0.

[21] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[22] T. M. Inc. *System Identification Toolbox*. Natick, Massachusetts, United States, Release 2020b. URL: https://se.mathworks.com/help/ident/.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison A .and Kopf, E. Yang, M. DeVito Z., A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[24] L. F. Shampine, M. W. Reichelt, and J. A. Kierzenka. "Solving Index-I DAEs in MATLAB and Simulink." In: *SIAM Review* 41.3 (1999), pp. 538–552. ISSN: 00361445. URL: http://www.jstor.org/stable/2653267.

[25] D. Jung, E. Frisk, and M. Krysander. "Residual change detection using low-complexity sequential quantile estimation**The research has been funded by Volvo Car Corporation in Gothenburg, Sweden." In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 14064–14069. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2017.08.1842.