Flow Anomaly Detection and Reporting

Development and implementation of an automatic dosing unit flow anomaly detection and reporting tool

Robin Wiklund

Examensarbete för ingenjörsexamen (YH)

El- och automationsteknik

Vasa 2023

BACHELOR'S THESIS

Author:	Robin Wiklund
Degree Programme:	Electrical Engineering and Automation, Vaasa
Specialisation:	Automation
Supervisor:	Joachim Böling

Title: Flow Anomaly Detection and Reporting

Date: 10.12.2023 Number of pages: 28

Abstract

The assignment was provided on behalf of Wärtsilä Catalyst Systems. This thesis covers the development and implementation of an automatic dosing unit flow anomaly detection and reporting tool. There was a request from Factory Acceptance Testing to improve the dosing unit testing procedure. The improvement involves a transition from validating the dosing through the visual inspection of urea flow to an automatic stability assessment method.

To automatically assess the urea flow stability, anomaly detection is used. The foundation, pros, and cons of different flow anomaly detection methods for industrial control loops are explained. Also, how to implement these detection methods in practice using mathematical formulas and Python code is presented. The result is a robust tool for automatic dosing unit performance assessment through the analysis of flow data.

The anomaly detection methods that are used include analysing the desired flow compared with the real measured flow using mean absolute error, oscillation detection using real-valued fast Fourier transform, and outlier detection using interquartile range.

This contributes to enhancing the reliability and efficiency of dosing unit testing procedures, providing a valuable resource for experts involved in Factory Acceptance Testing. The architecture of the implemented system is visualized using a component diagram in Unified Modeling Language.

Language: English
Key words: SCR, IQR, MAE, RFFT, PID control loop

EXAMENSARBETE

Författare:	Robin Wiklund
Utbildning och ort:	El- och automationsteknik, Vasa
Inriktning:	Automation
Handledare:	Joachim Böling

Titel: Detektering och rapportering av flödesanomali

Datum: 10.12.2023 Sidantal: 28

Abstrakt

Detta examensarbete utfördes på uppdrag av Wärtsilä Catalyst Systems. Examensarbetet behandlar utveckling och implementering av en automatisk anomalidetektering för doseringsenheter samt ett rapporteringsverktyg. Det fanns en begäran från Factory Acceptance Testing om att förbättra testproceduren för doseringsenheterna. Förbättringen består av en övergång från validering av doseringen genom visuell inspektion av ureaflödet till en automatisk stabilitetsbedömningsmetod.

För att automatiskt bedöma flödesstabiliteten för urea används anomalidetektering. Grunden samt fördelar och nackdelar med olika metoder av anomalidetektering för industriella styrslingor förklaras. Dessutom presenteras implementering av dessa detekteringsmetoder i praktiken med hjälp av matematiska formler och Python-kod. Resultatet är ett robust verktyg för automatisk bedömning av doseringsenhetens prestanda genom analys av flödesdata.

De metoder för anomalidetektering som används inkluderar analys av önskat flöde jämfört det verkliga uppmätta flödet genom genomsnittlig absolutavvikelse, oscillationsdetektering med hjälp av snabb Fouriertransform med verkligt värde och avvikelsedetektering med interkvartilområde.

Detta bidrar till en förbättrad tillförlitlighet och effektivitet i testprocedurer för doseringsenheter, vilket ger en värdefull resurs för experter som är involverade i Factory Acceptance Testing. Arkitekturen för det implementerade systemet visualiseras med hjälp av ett komponentdiagram i Unified Modeling Language.

Språk: engelska

Nyckelord: SCR, IQR, MAE, RFFT, PID-kontrollslinga

OPINNÄYTETYÖ

Tekijä:	Robin Wiklund
Koulutus ja paikkakunta:	Sähkö- ja automaatiotekniikka, Vaasa
Suuntautumisvaihtoehto:	Automaatio
Ohjaaja:	Joachim Böling

Nimike: Virtausanomalian havaitseminen ja raportointi

Päivämäärä: 10.12.2023 Sivumäärä: 28

Tiivistelmä

Tehtävä annettiin Wärtsilä Catalyst Systemsin toimeksiannosta. Opinnäytetyö käsittelee annosyksiköiden automaattisen poikkeamahavainnon kehittämistä ja toteuttamista sekä raportointityökalua. Factory Acceptance Testing pyysi parantamaan annostusyksiköiden testausmenettelyä. Parantaminen tarkoittaa siirtymistä annoksen validoinnista ureavirran silmämääräisen tarkastuksen avulla automaattiseen stabiilisuuden arviointimenetelmään.

Urean virtauksen stabiliteetin automaattista arviointia varten käytetään anomalianhavaintoa. Selitetään teollisuuden ohjaussilmukoiden eri poikkeamien havaitsemismenetelmien perusteet sekä edut ja haitat. Lisäksi esitellään näiden havaitsemismenetelmien käytännön toteutus matemaattisten kaavojen ja Python-koodin avulla. Tuloksena on vankka työkalu annosteluyksikön suorituskyvyn automaattiseen arviointiin virtausdatan analyysin avulla.

Käytetyt anomalianhavaitsemismenetelmät sisältävät halutun virtauksen analysoinnin verrattuna todelliseen mitattuun virtaukseen käyttäen keskipoikkeamaa, heilahtelun havaitsemisen käyttäen reaaliarvoista nopea Fourier-muunnosta ja poikkeaman havaitsemisen käyttäen kvartiiliväliä.

Tämä edistää annosteluyksiköiden testausmenetelmien luotettavuutta ja tehokkuutta, tarjoten arvokkaan resurssin tehtaan hyväksymistestaukseen osallistuville asiantuntijoille. Toteutetun järjestelmän arkkitehtuuri visualisoidaan komponenttikaaviolla Unified Modeling Language -kieltä käyttäen.

Kieli: Englanti
Avainsanat: SCR, IQR, MAE, RFFT, PID-tarkastussilmukka

Table of Contents

Terms and abbreviations

SCR	Selective catalytic reduction
NOx	Nitrogen oxides
NOR	NOx reducer
UNIC	Wärtsilä Unified Controls
PDU	Power distribution unit
FAT	Factory Acceptance Testing
HIL	Hardware-in-the-loop
IAE	Integral of absolute error
MAE	Mean absolute error
RFFT	Real-valued fast Fourier transform
IQR	Interquartile range
UML	Unified Modeling Language
GUI	Graphical user interface

1 Introduction

This thesis covers the development and implementation of an automatic dosing unit flow anomaly detection and reporting tool. The assignment was provided on behalf of Wärtsilä Catalyst Systems. One of the main products Catalyst Systems is developing is the NOx reducer (NOR) system.

The NOR system is an emission after-treatment system based on the selective catalytic reduction (SCR) technology for NOx reduction [1]. In the Catalyst Systems team are Factory Acceptance Testing (FAT) experts who will mainly benefit from this study. The focus will be testing the dosing units in FAT.

1.1 Background

The purpose of FAT is to make sure that the dosing unit, pump unit and power distribution unit (PDU) are properly inspected and tested. It is required to save the test results of the dosing unit before they are shipped to the customer. The dosing unit is currently validated by visually examining the urea flow from the flow sensor.

From the flow trend, the deviation and stability of the flow can be examined as it tries to converge with the flow setpoint. It is hard to determine whether a dosing unit with unstable flow is good enough to maintain NOx targets in the field.

1.2 Reducing agents

SCR with ammonia is the most efficient way to reduce NOx emissions from exhaust gases. NOx emissions from combustion processes pose a major threat to human health and are difficult to avoid. To achieve high NOx reduction efficiencies, it is important to use the proper dosage of reducing agent, such as ammonia, in addition to selecting the appropriate type of SCR catalyst. [2].

Due to its toxic properties and safety concerns, ammonia is supplied in an aqueous urea solution. In most SCR applications, a 32.5wt.-% urea solution is used, which is commercially named AdBlue® in Europe and Diesel Exhaust Fluid in the USA. The precursor liquid is sprayed into the tailpipe in front of the SCR catalyst and decomposes into ammonia and carbon dioxide. [2].

1.3 NOR System

The main components of the NOR installation can be seen in Figure 1. The function of the pump unit is to provide constant pressure of urea to the dosing unit and the function of the dosing unit is to regulate the flow of urea. The distribution unit distributes power and communication.

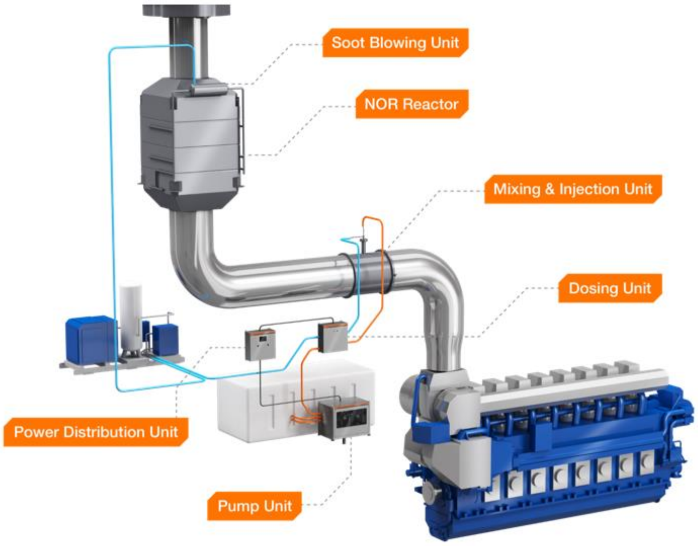

Figure 1: NOR technical diagram [1].

The dosing unit uses the Wärtsilä Unified Controls (UNIC) system which in this case consists of a communication module. The communication module in the dosing unit communicates with the pump unit, PDU and the engine.

The description of the communication module by Wärtsilä is: "The communication module is the main gateway to the UNIC system from vessel systems, supporting multiple interfaces such as Modbus, OPC, hardwired I/O, etc. COM is a key module for UNIC system communication and responsible for several control functions, software and configuration update management." [3, p. 3].

The module has a control loop implemented with PID controller for regulating the urea dosing. The PID controller manages the control valve which regulates the flow of urea. The flow feedback is measured with a flow sensor.

1.4 Dosing issues

Oscillations in process data are a common problem in many industries. Oscillations significantly affect how control loops perform in industrial processes. Oscillations in the urea dosing process would result in poor NOx values and may even cause excessive ammonia slip which is not desirable [2]. That is why it is critical to ensure proper dosage of urea. [4].

The primary objective is to detect oscillations and noise within the system and assess its stability. Specifically, the study observes how the control loop, particularly the PID controller, responds to these disturbances in its efforts to stabilize urea flow. Rather than searching for specific faults, this thesis aims to comprehend how the system responds to these challenges. This is done to determine if there are any faults in the system. In a good dosing unit, the flow in the control loop is rapidly stabilized and kept stable as shown in Figure 2 where the black line represents the setpoint and the red line represents the urea flow.

Figure 2: Stable control loop.

Overshooting will be ignored since the test will start after a configured time. Also, overshooting in this application is normal. A transient time is used to wait for the system to stabilize. The test will start after a configured transient time whether the system is stable or not. More about the testing methods in Chapter 2.1.

1.5 Goal

The goal is to create and implement a Python tool dedicated to evaluating the stability of urea flow in the dosing unit through the analysis of flow data. The primary focus is on formulating a robust mathematical assessment. Additionally, the tool should feature an interactive graphical user interface (GUI) with the capability to manage communication with the dosing unit. The tools should have the ability to both write setpoints and read urea flow values independently while also providing the result of the flow stability.

The evaluation of the flow data using the tool will be conducted against predefined acceptance criteria. In this case, the acceptance criteria consist of value bounds that the urea flow must not exceed. For more information about the implementation and the acceptance criteria, read the implementation in Chapter 3. Concrete acceptance criteria are important since FAT experts currently lack clear guidelines. The tool will help to standardize the testing.

The computer used by the FAT experts is connected to the communication module placed in the dosing unit. The tool will set urea setpoints in the communication module and automatically validate the urea flow based on the established acceptance criteria. The Python tool operates by looping through each urea flow setpoint in a configurable list. These step changes can be seen in Figure 2.

After a new setpoint is set, the system waits for the urea flow to stabilize. This was briefly explained in the end of Chapter 1.4. The test starts after the transient time has passed and the Python tool then requests the flow values from the communication module. Flow values are measured by a flow sensor and then transmitted to the communication module. These flow values are used to determine whether the urea flow stability is accepted or not. The result will then be automatically exported to a test document by the Python tool.

1.6 Delimitations

The study is limited to dosing units pre-commissioning. This means only the dosing units that are tested in factory acceptance testing. Dosing units shall not be connected to any engine undergoing a test. Thus, resulting in less external noise for better test performance. The script will not automatically start and shut down the test system. The FAT expert needs to manually operate the test system. This is for safety reasons. This thesis will only contain the implementation of the mathematical assessment. This means that the GUI and the communication will be implemented but will not be explained as it is not of interest.

2 Anomaly detection

Anomaly detection is a technique to find rare events in data streams, also called anomalous events. Anomaly detection is fundamental in many applications, most notably in real-time applications. Spotting anomalies is vital in health, critical infrastructures, and security applications, to name a few. Anomaly detection allows for the detection of unexpected occurrences, making it an essential tool in data analysis and processes. [5]. In this study, anomaly detection will be used to validate dosing units. Anomalies in the system can be control valve stictions, oscillations, high frequencies, noise, or outlying values. The impact of oscillations and other anomalous events was mentioned in Chapter 1.4.

2.1 Simulation

To find anomalous events, a hardware-in-the-loop-simulation (HIL-simulation) can be used. This kind of simulation is a widely used testing method in system development. The simulation communicates with real physical system components like pumps, valves, or sensors. This approach allows engineers to test how the control algorithm behaves in real-time with real hardware, providing valuable insights into the stability and performance of the system. [6].

A HIL-simulation is especially useful when dealing with complex systems, ensuring that the control system responds accurately and efficiently to different scenarios, such as flow oscillations, in a controlled virtual environment. [6].

In this study, the HIL-simulation is the Python tool that requests hardware variable values such as the flow and writes variable values such as the setpoint. Physical system components are also simulated, such as the engine. One of the reasons the engine is simulated is described in Chapter 1.5. However, the engine simulator for this scenario already exists in the company.

2.2 Control valve stiction

It is common to find control valves in the industry. They are essential in various process control systems such as regulating fluid flow, pressure, and temperature. They ensure precise control and maintain product quality. However, a significant challenge in control valve operation is stiction. A problem where the valve stem gets stuck due to static friction. When friction is overcome, an abrupt jump known as the slip-jump may cause issues like oscillations in the control loop. [7].

This disrupts the valve's performance and product quality. This may also accelerate the wear and ageing of the valve. There are countless methods of detecting valve stictions since this phenomenon is common in the industry. Detecting valve stiction has become a focal point in academic and engineering research. This focus is driven by its practical importance in enhancing system reliability, control performance, and product quality. [7].

A dedicated stiction detection method will not be implemented, instead problems with the control valve can be found using oscillation detection and other techniques coming in the following chapters.

2.3 Oscillation detection

Oscillations can be described with Horch's definition as a periodic variation visible to the human eye and not entirely concealed by noise. Another definition, put forth by Choudhury and colleagues designate oscillation as a time series with clearly defined amplitude and frequency. [4].

Causes of oscillations can be mechanical faults such as valve stiction, fluctuating air regulator or air pockets in the urea feed line. This was also discussed with Westman in an oral discussion in autumn 2023. Identifying these causes is undoubtedly valuable. Even if the presence of oscillations is confirmed, it is not always a cause for immediate concern, as the impact of oscillations may vary. [4].

To differentiate between disapproved oscillatory loops and those that do not affect performance, the strength of the oscillation needs assessment. [4]. This must be mathematically implemented since the Python tool will be used. The approach to oscillation detection can be done in many ways. Integral of absolute error (IAE) evaluation is the base of many oscillation detection techniques [4]. Similar solutions can be found in PID controller tuning, such as IAE performance criteria [8].

$$IAE = \int_0^\infty |e(t)|\, dt \tag{1}$$

The control error e in formula 1 is the error between the setpoint and the urea flow. The result of IAE is the summed area of the error to the setpoint. [8]. This approach requires a continuous function $e(t)$. Since the flow data is discrete rather than continuous, other similar solutions are preferred.

2.3.1 MAE

Mean absolute error (MAE) is often used as a metric for its practicality in matching the units of error values with the target values being predicted. The predicted value is in this case the setpoint. MAE maintains a consistent linear scale meaning errors are treated equally without any bias towards specific errors as scores increase proportionally with error increments. [5].

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_1 - \hat{y}_i| \tag{2}$$

The MAE score is derived by averaging the absolute error values making MAE suitable for assessing errors regardless of their direction. This characteristic makes MAE a straightforward and effective tool instead of a similar solution to IAE. [5].

2.3.2 Frequency analysis

One of the most straightforward methods for identifying oscillations is through frequency domain analysis, where oscillations manifest as distinct peaks in the spectrum. The frequency domain analysis distinguishes between different types of oscillations. If there are many peaks or a broad peak with a wide range, it indicates various or irregular oscillations in the system. [4].

Frequency analysis can be used to detect oscillations with high frequencies. Large amounts of noise and disturbances will generate peaks in the spectrum. Frequency analysis is a powerful tool for finding phenomena that may occur in the event of failure even if the score of MAE is approved. There are also other frequency-based solutions. however, most of the simple solutions are based on the period of oscillation or time interval between two zero-crossings and thus not suitable for this study. [4].

In the stable control loop as shown in Figure 3, frequency analysis with real-valued fast Fourier transform (RFFT) within the selected interval shows absent frequencies, as seen in the amplitude spectrum in Figure 4.

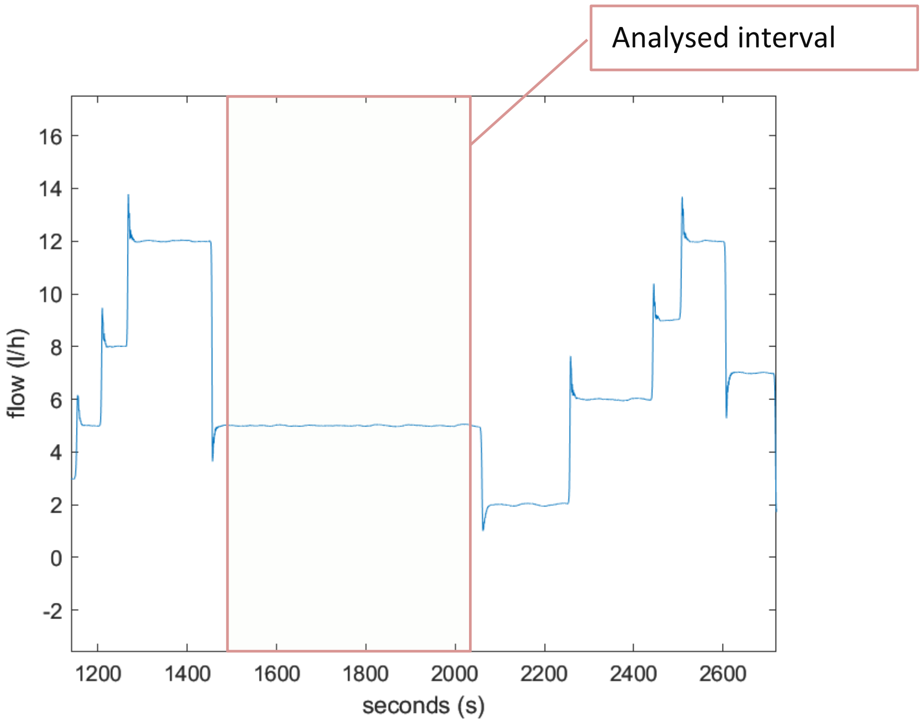

Figure 3: RFFT interval analysis on a stable control loop.

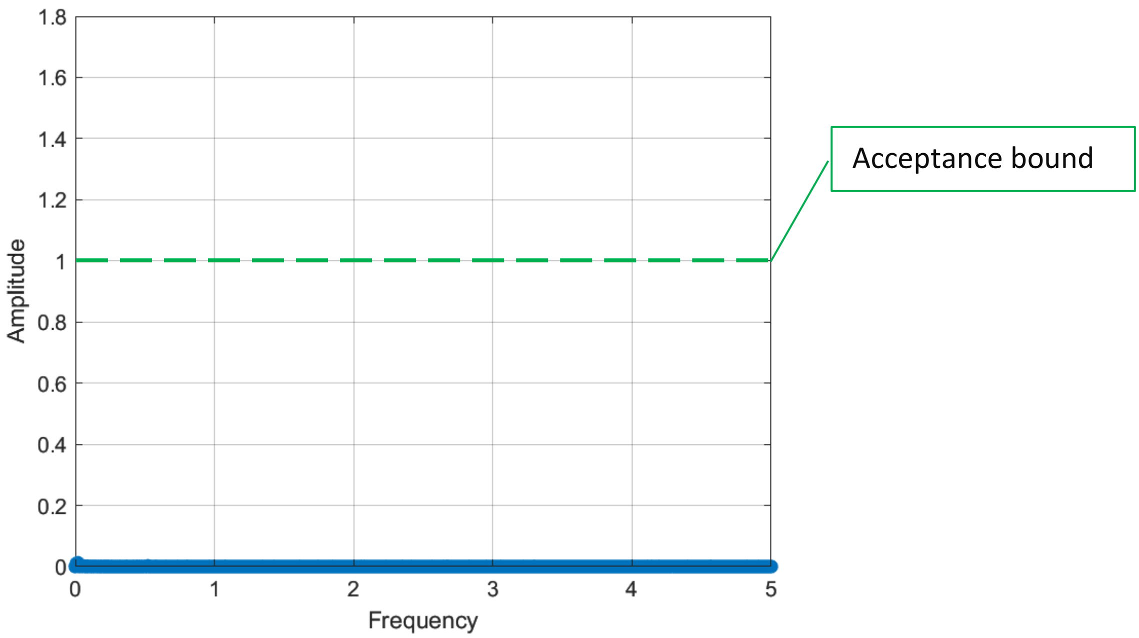

Figure 4: Amplitude spectrum on a stable control loop.

In contrast, the volatile control loop in Figure 5 displays a considerably larger number of frequencies within the selected interval. This can also be seen in the amplitude spectrum shown in Figure 6.

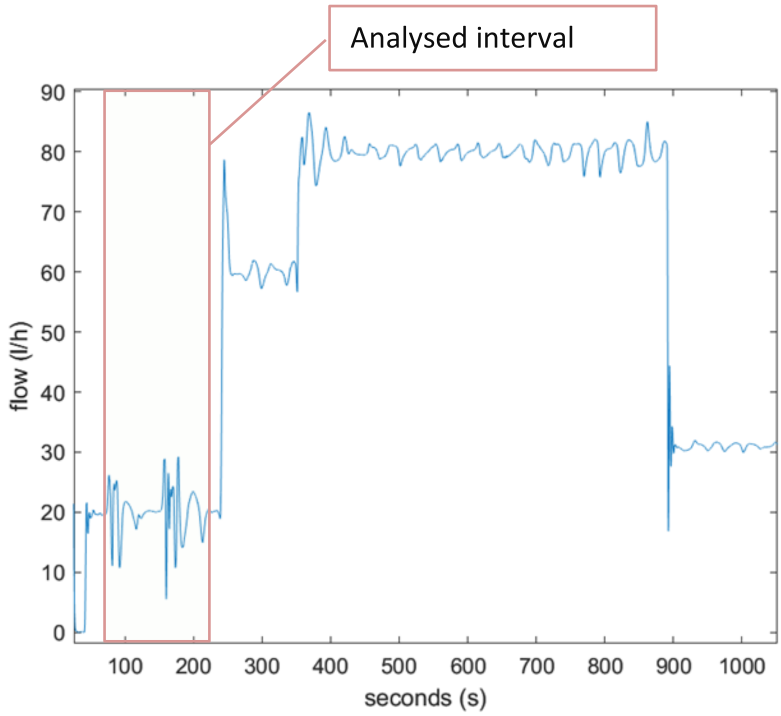

Figure 5: RFFT interval analysis on a volatile control loop.

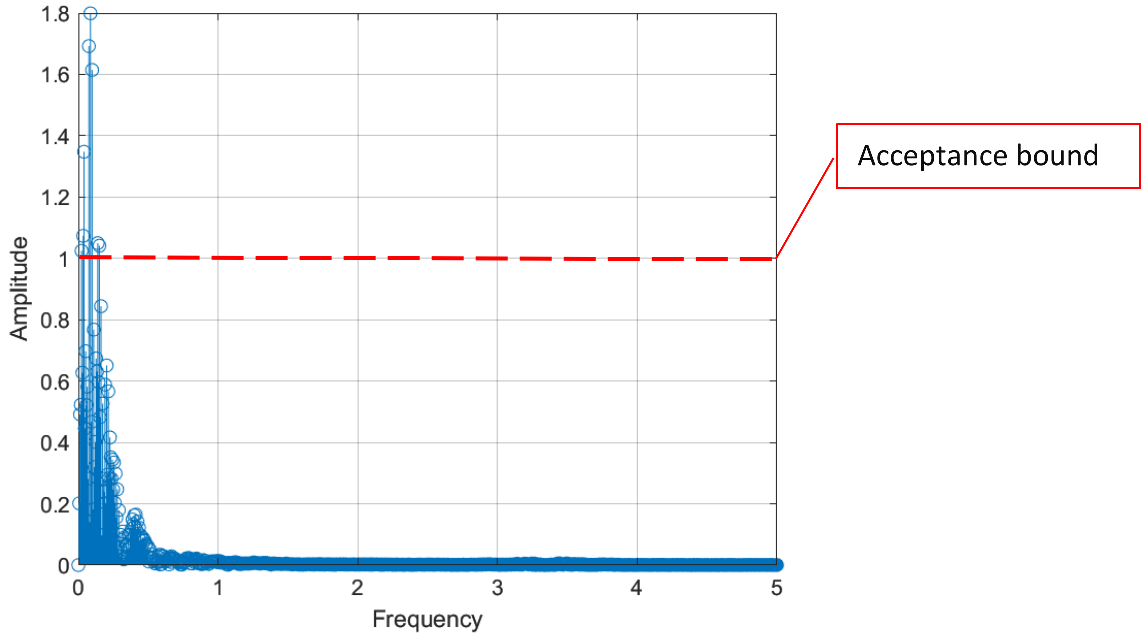

Figure 6: Amplitude spectrum on a volatile control loop.

2.4 Outlier detection

Noisy data and outliers may seem alike. However, outliers are different from noisy data in data analysis. While noise is usually removed as irrelevant, outliers provide important insights. Outliers can be defined as observations that significantly deviate from others, suggesting they come from a different source. [9].

In simple terms, outliers are data points that strongly differ from the usual patterns in a dataset or expected behaviours. Deciding whether to keep or remove outliers depends on the situation. Sometimes, removing them is necessary to avoid misleading analyses, but in other cases, keeping outliers can be beneficial. [9].

An example of keeping outliers is instances where neither MAE nor frequency analysis will detect outliers such as a high peak in flow value for a brief moment as seen in Figure 7, thus not resulting in any unaccepted oscillatory error scores. Note that real data with outliers will differ from this demonstration since the peaks in Figure 7 and Figure 8 are only a simulation to illustrate the problem clearly.

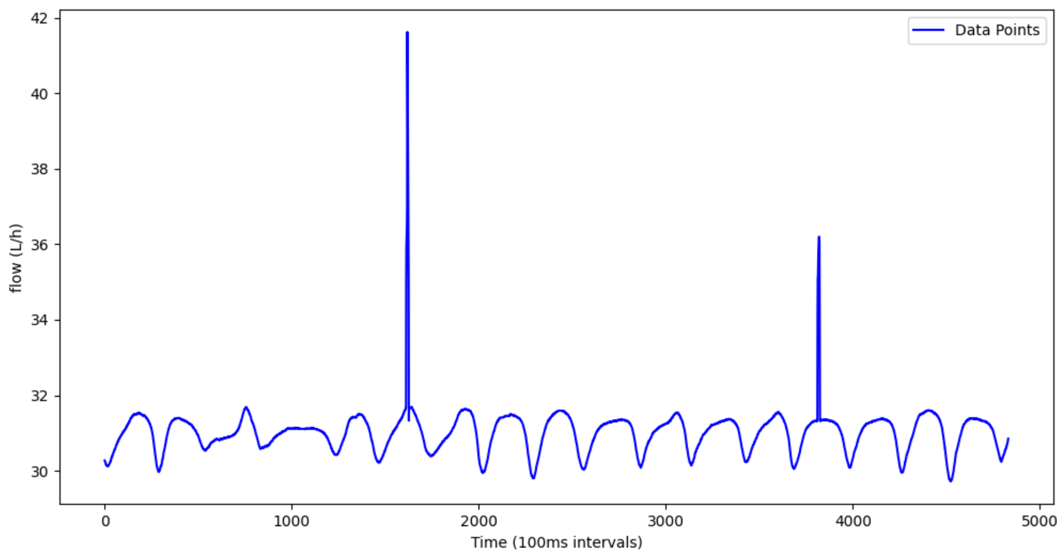

Figure 7: Outlying data points in the data set.

To detect these outliers, an interquartile range (IQR) method, specifically the Gaussian-based method can be used. Outlying values can be detected using an upper and lower bound [9]. The boundaries and outlying anomalous values can be seen in Figure 8.

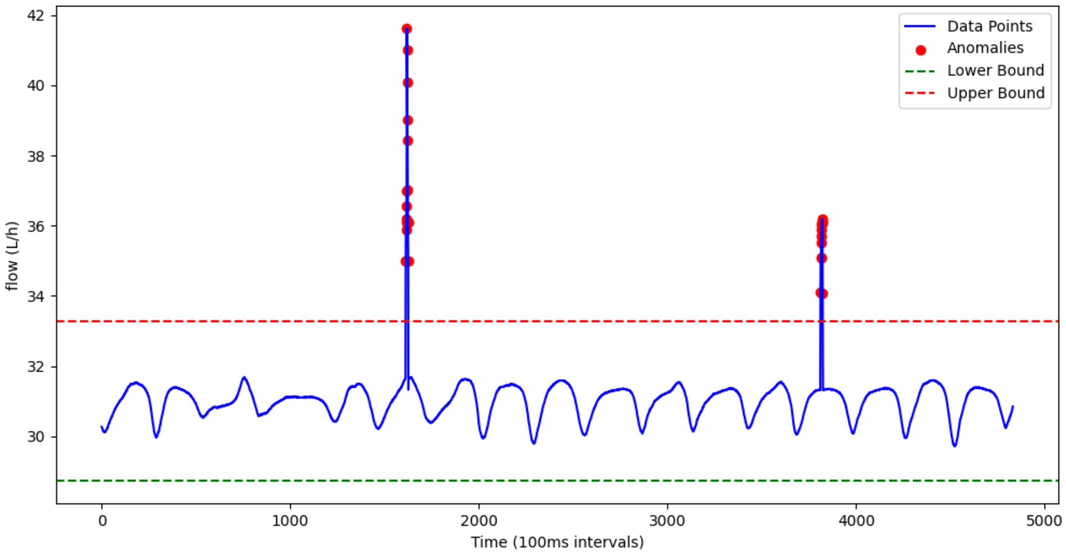

Figure 8: Implemented IQR outlier detection method.

To establish the bounds, the IQR needs to be calculated. This involves determining the median of the whole data set, the median of the lower range and the median of the higher range. Let's consider the dataset 3, 4, 5, 6, 7, 8, 6, 3, 2, 1, 2 as an example. To simplify visualization, sort the dataset in ascending order, as demonstrated in Figure 9. [9].

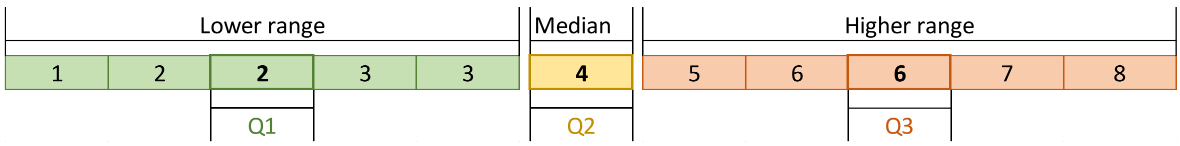

Figure 9: Quartiles of the data set.

The two ranges are divided by the median. The median of the whole dataset Q_2 is equal to 4. All values lower than the median are in the lower range. Values bigger than the median are in the higher range. By taking the median in both ranges, the Q_1 and Q_3 can be set. [9].

In this case the lower range median Q_1 equals 2. The median of the higher range Q_3 equals 6. These values are also visualized in Figure 9. IQR, lower bound (B_L), and higher bound (B_H) can then be calculated using the formulas 3, 4, and 5. [9].

$$IQR = Q_3 - Q_1 \tag{3}$$

$$B_L = Q_1 - 1{,}5 \cdot IQR \tag{4}$$

$$B_H = Q_3 + 1{,}5 \cdot IQR \tag{5}$$

This results in IQR equal to 4, B_L to be equal to -4 and B_H equal to 12. Values below B_L and values above B_H are classed as outlier values. The constant value 1,5 acts as a multiplier to adjust the outlier bounds. [9].

It is important to note that the IQR outlier detection method is suitable only for univariate data. Univariate data consists only of a single variable. This limitation is not problematic in this context since the assessment focuses solely on flow values, without considering the combination of flow and time. [9].

3 Implementation

Methods for validating the urea flow stability and detecting anomalies are:

- MAE, previously explained in Chapter 2.3.1.
- Frequency analysis using RFFT, previously explained in Chapter 2.3.2.
- Outlier detection using the IQR method, previously explained in Chapter 2.4.

By combining these methods, the final anomaly detection tool is formed. The provided data in this work is based on simulated data used for developing this tool. The solution will be tuned for the product in question.

3.1 Error calculation using MAE

The implementation of MAE can be seen in code example 1. This is based on formula 2 taken from Chapter 2.3.1. The accepted error or the bound is calculated from the setpoint. This is done by taking the configurable acceptance bound percentage variable multiplied by the setpoint and then adding a static configurable acceptance bound value.

Code example 1. Establishing MAE boundary using Python.

```
MAE = np.mean(np.abs(flow_samples - setpoint))
Accepted_value = setpoint * acceptance_bound_percentage / 100 + acceptanceBound_static
```

This can be for example 2% + 0,5 L/h where 0,5 L/h provides a consistent increment. This is a useful technique for smaller flows that otherwise would have too strict acceptance bound when only using the percentage of the setpoint. This can be visualized in Figure 10, where the red line is the consistent added increment. The x-axis is the flow setpoint and the y-axis is the accepted error. The accepted error is 2% for the blue line and 2% + 0,5 L/h for the red line.

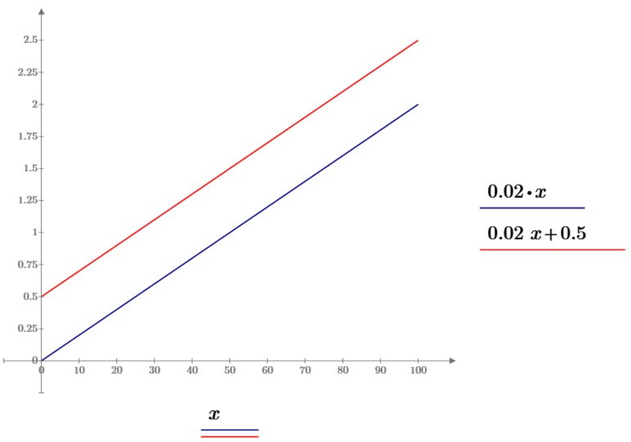

Figure 10: Comparing methods to find optimal accepted error.

3.2 Frequency analysis using RFFT

How to detect oscillating urea flows was previously explained in Chapter 2.3.2. To implement this detection method, RFFT in Numpy is used as seen in code example 2.

Code example 2. RFFT frequency analysis using Python.

```
# Signal around zero
flow_samples = flow_samples - np.mean(flow_samples)

# Amount of samples
n = len(flow_samples)

# Normalize with / n, compensate negative frequencies by doubling the amplitude with * 2
rfft_values = np.fft.rfft(flow_samples) / n * 2

# 100ms interval --> frequency content of up to 5 Hz
frequency_values = np.fft.rfftfreq(n, d=1/10)

amplitudes = np.abs(rfft_values)
```

RFFT is different from discrete Fourier transform (DFT) and fast Fourier transform (FFT). DFT and FFT will compute negative frequencies that are mirrored versions of the positive ones. [10]. This can be seen in the amplitude spectrum in Figure 11.

The upper part of Figure 11 shows the analysed signal. The lower part shows the amplitude spectrum of the analysed signal. In this case, the amplitude spectrum shows two frequencies that were combined into a signal.

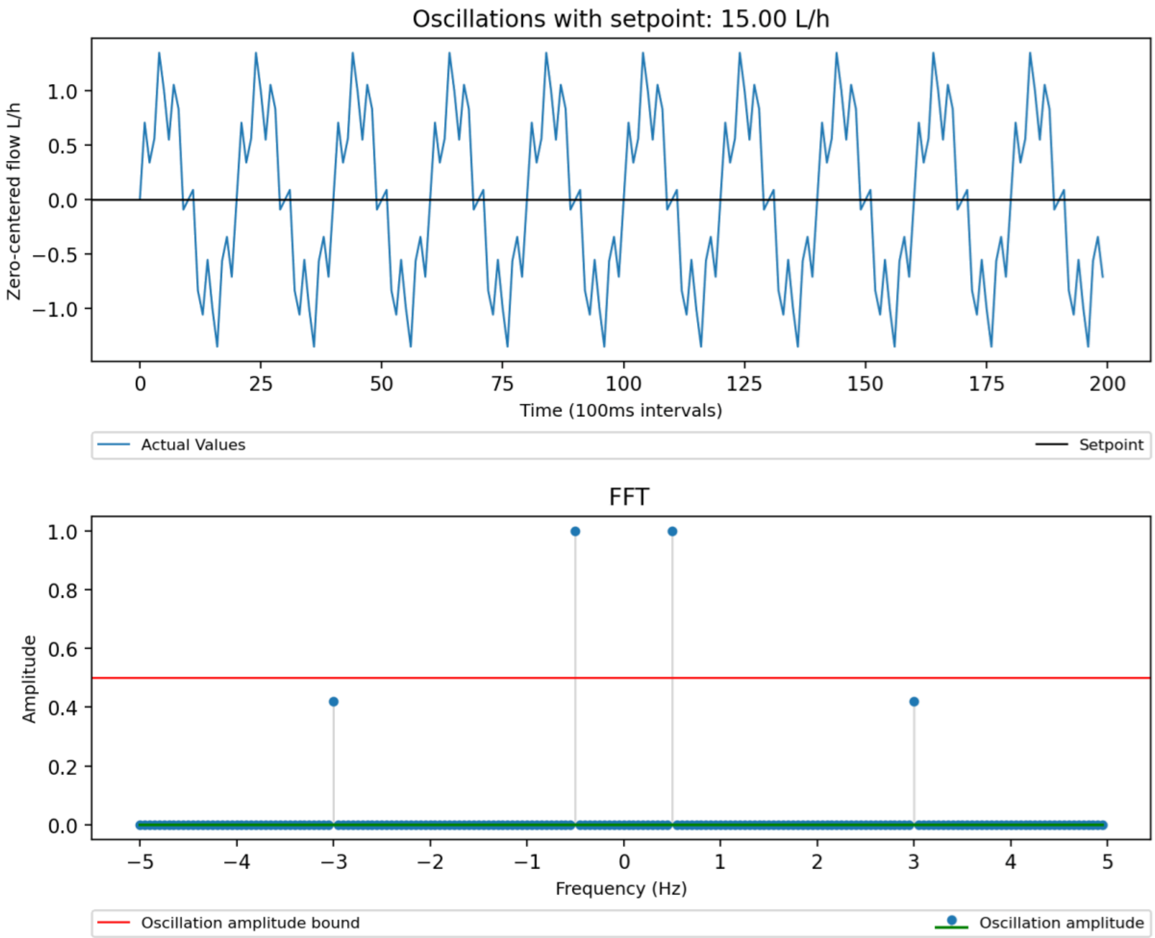

Figure 11: Negative frequencies using fast Fourier transform.

RFFT does not compute the negative frequency terms [10]. This is useful since it is not necessary to have this kind of mirror effect as seen in Figure 11 when using real values. The result of using RFFT with the same data can be seen in Figure 12.

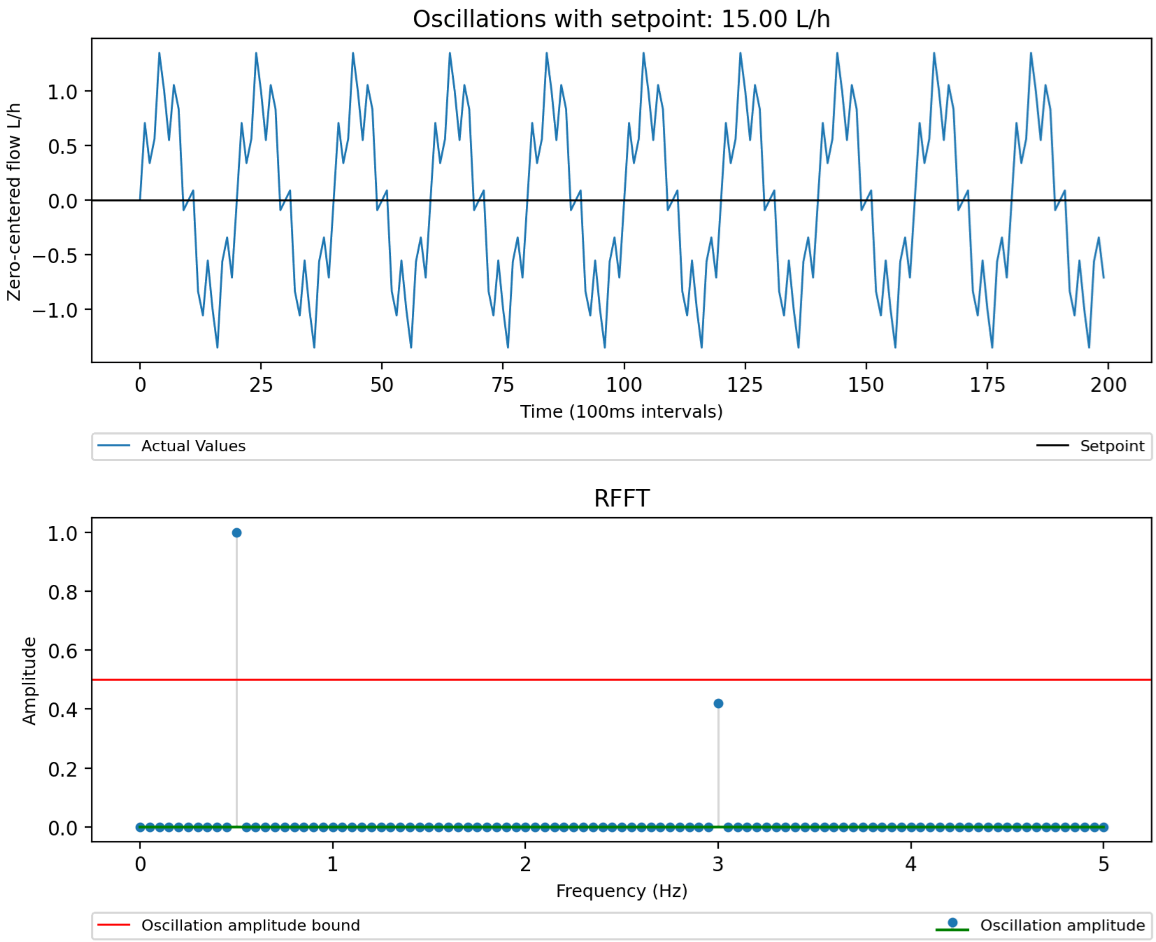

Figure 12: Only positive frequencies using RFFT.

Since RFFT does not compute the negative frequency terms, the length of the transformed axis of the output is therefore n//2 + 1. A compensation for the discarded negative frequencies is necessary. The compensation is implemented by doubling the amplitude of the positive frequencies. [10]. This implementation can be seen in code example 2.

From the amplitude spectrum in both Figure 11 and Figure 12, the dominant frequency can be seen. The dominant or main frequency is 0,5 Hz at the amplitude of 1. The other signal is an overtone at 3 Hz with a lower amplitude. By looking at both Figure 11 and 12, you can see that the maximum frequency that can be analysed is 5 Hz. This is related to the Nyquist frequency.

The Nyquist frequency is half of the sampling frequency, and it represents the maximum frequency that can be accurately represented in a sampled signal without aliasing. The sampling interval is 100 ms, which corresponds to a sampling frequency of 10 Hz. The Nyquist frequency is therefore 5 Hz, as seen in the figures. [11].

To analyse frequency components beyond 5 Hz, you will need to increase the sampling frequency by reducing the sampling interval [11]. Figure 11 and Figure 12 also show that the oscillations are around the zero line instead of the setpoint, even when the setpoint is at 15 L/h.

The reason for this implementation can be visualized by observing the difference between Figure 13 and Figure 14, where the signal only consists of a single frequency. Both figures use the frequency 1 Hz with the amplitude of 1. The signal is set around the setpoint 15 L/h. From the amplitude spectrum in Figure 13 can only the 1 Hz frequency be seen.

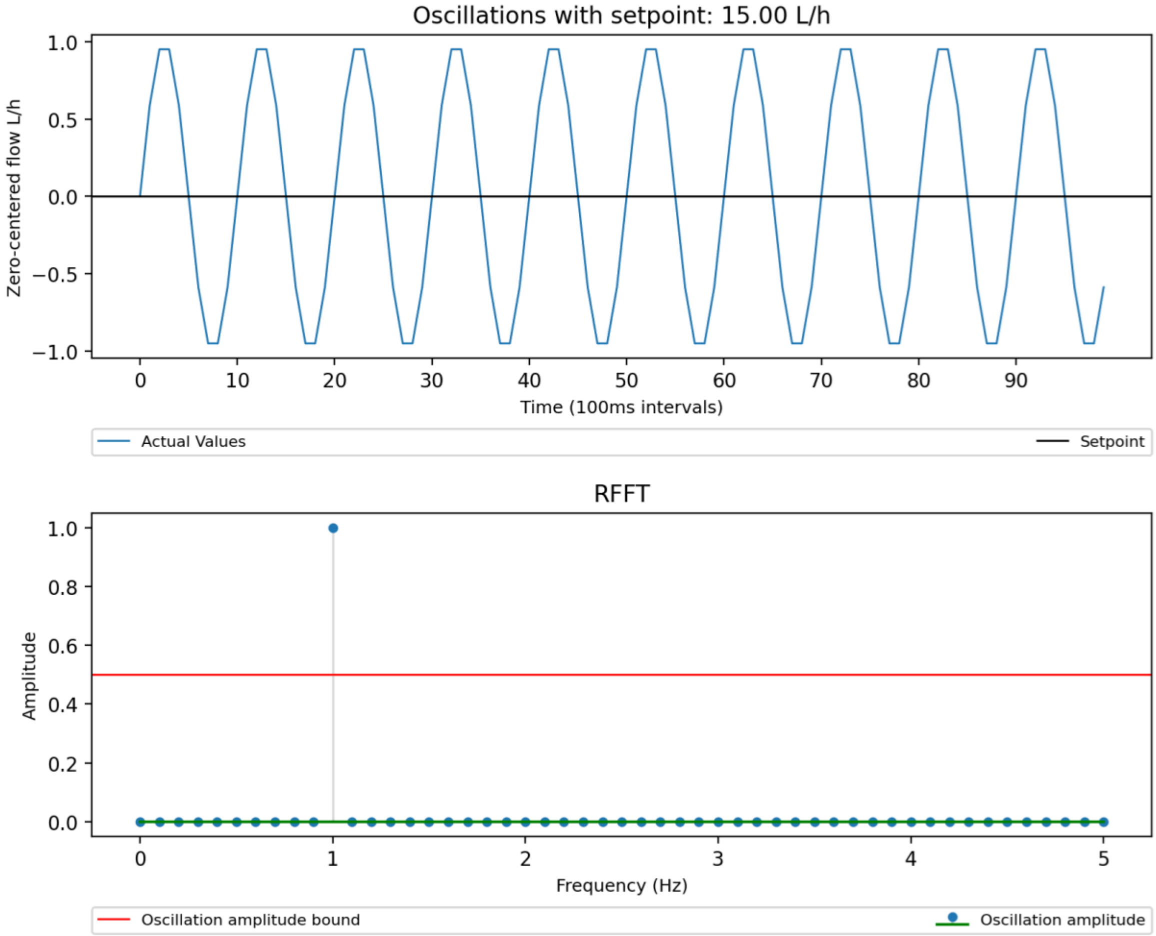

Figure 13: Oscillations around the zero line.

However, Figure 14 shows an additional peak in the amplitude spectrum, which is a consequence of the signal not being centred around zero.

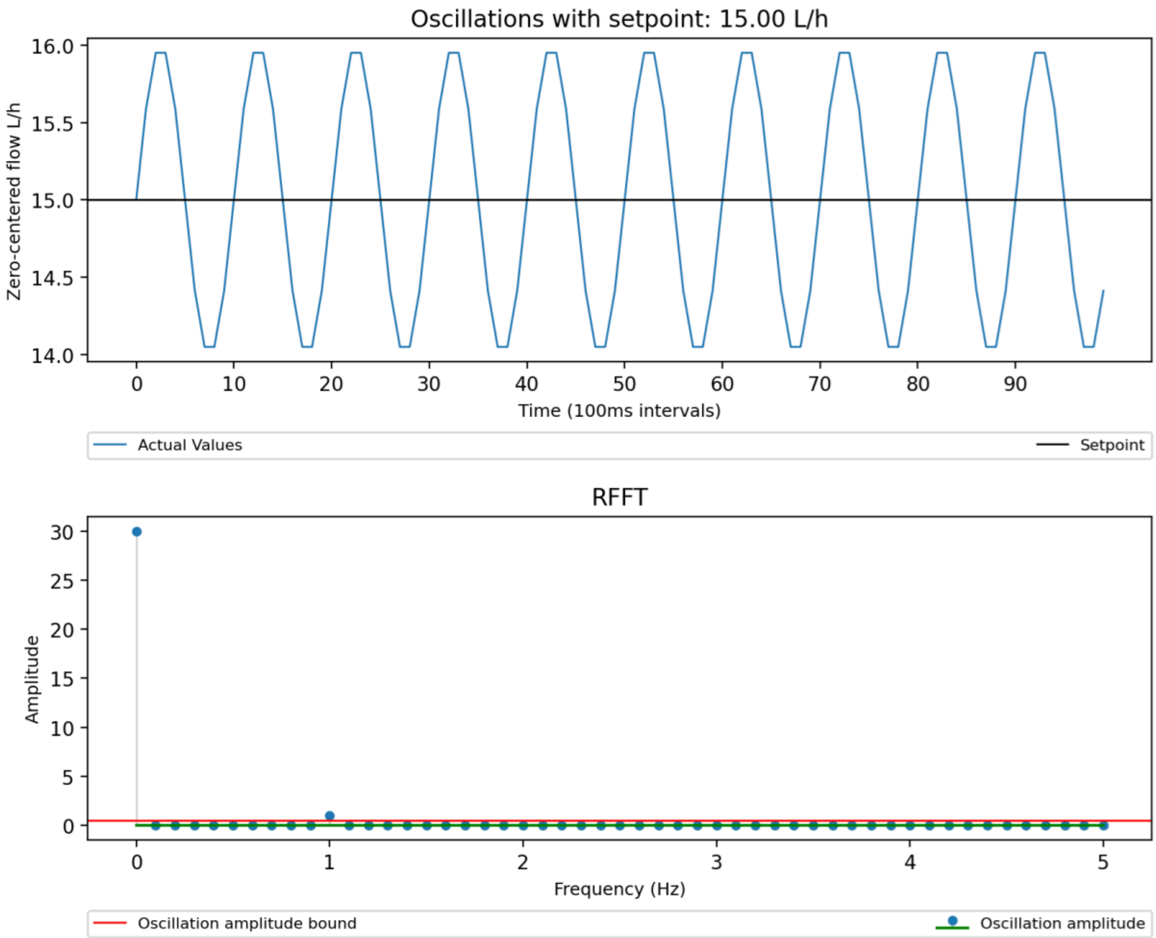

Figure 14: Oscillations around the setpoint.

This extra peak, with the amplitude of 30 at approximately 0 Hz, indicates the presence of an undesired offset in the signal. This was an important example of why it is important to zero-center the signal before analysing. The reason why the amplitude of the offset in Figure 14 is double the setpoint is simply because of the compensation of the negative frequency terms as previously said in this chapter.

3.3 Outlier detection using the IQR method

The utilization of IQR for outlier detection revealed a limitation. The selection of IQR over other methods was its ability to adapt the acceptance criteria according to the whole data set. This becomes a drawback when the majority of the data is flat or stable as the acceptance boundaries become too strict. This results in detected outliers. Figure 15 demonstrates a good control loop upon visual inspection.

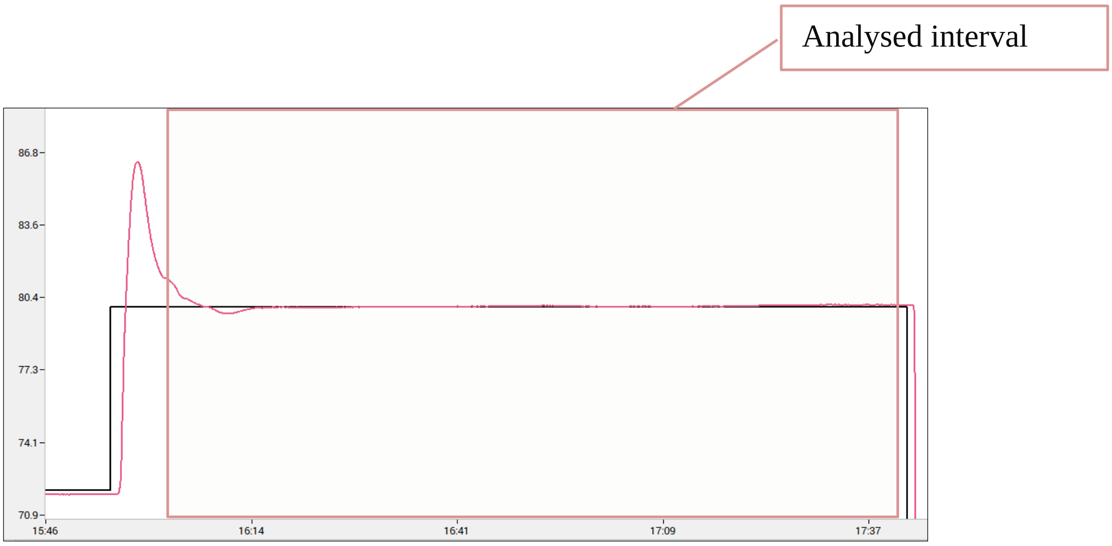

Figure 15: Analysed control loop.

These strict acceptance boundaries can be represented visually. This visualization helps pinpoint values identified as anomalies within the data series. Figure 16 illustrates that the flow values at the start exceed the upper bound and later when trying to stabilize, also fall below the lower bound.

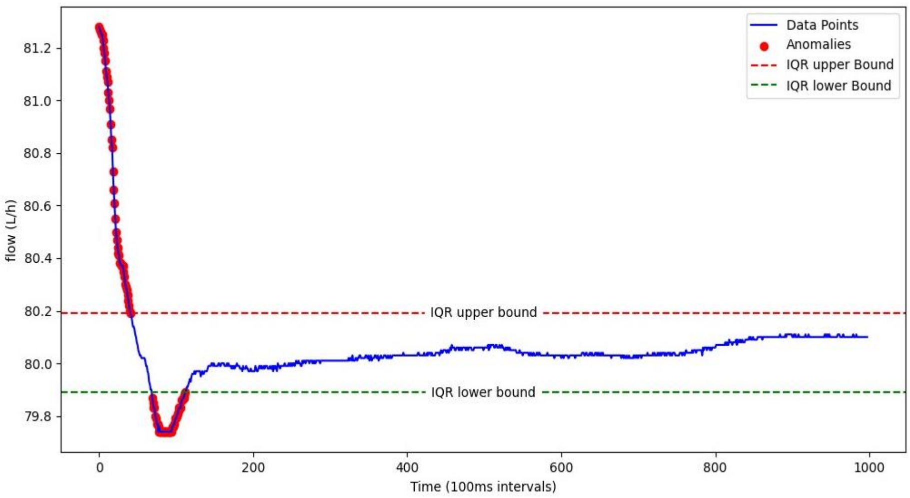

Figure 16: Anomalous values on flat data series using IQR with multiplier value 2.

The first anomalous values detected in Figure 16 happen when the values exceed the upper bound. This is because the flow did not stabilize within the transient time. The reason for this could be a true positive anomalous event where the flow takes too long to stabilize or that the flow stabilization time is too short and should be increased.

The second anomalous event when values fall below the lower bond is a clear false positive event since this type of correction by the control loop is seen as normal. A conceivable solution could be to increase the IQR multiplier to increase the acceptance bounds.

However, the multiplier needs to be considerably increased to include values below the lower bound for this kind of flat data, thus making the IQR method inapplicable for other types of data if increased. To visualize this issue, the multiplier value is increased to 5 from the previous 2. The new boundaries can be seen in Figure 17 with the same data series as used in Figure 16.

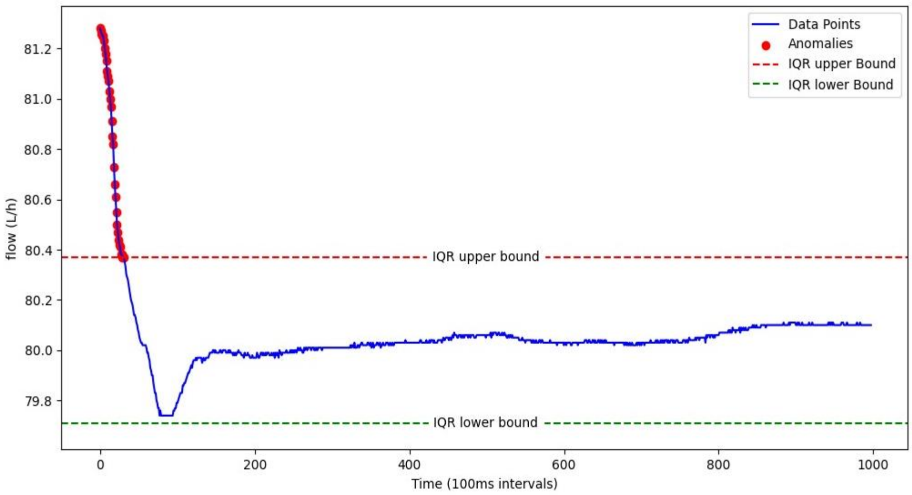

Figure 17: Anomalous values on flat data series using IQR with multiplier value 5.

Increasing the IQR multiplier results in a clear improvement, but the boundaries are still too strict. The flow is still only allowed to deviate roughly $\pm 0{,}3\ L/h$ or $\pm\ 0{,}4\ \%$ from the mean. The multiplier can’t be increasing further since it will affect fluctuating data series. To visually illustrate this issue, fluctuating data can be seen in Figure 18. The result of IQR on fluctuating data series shows completely different acceptance boundaries. Note the y-axis scale on both figures.

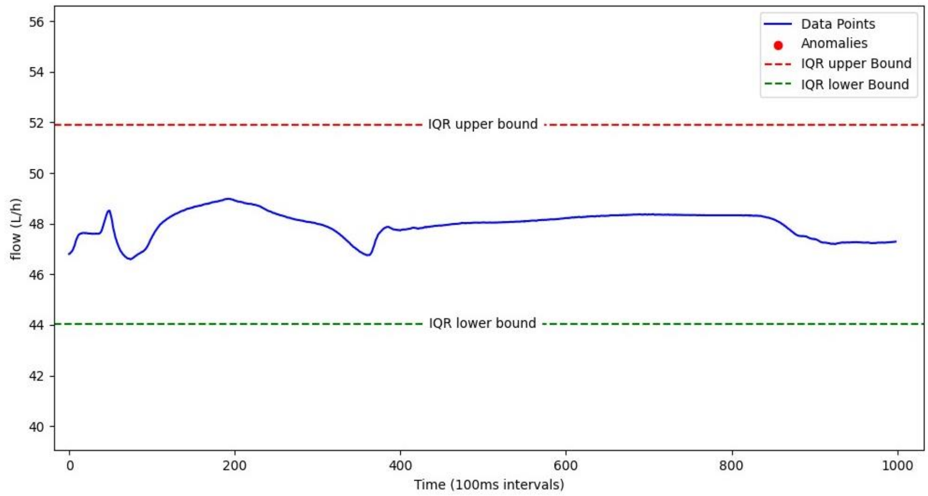

Figure 18: No anomalous values on fluctuating data series using IQR with multiplier value 5.

In Figure 18 is the flow allowed to deviate approximately $\pm 3{,}9\ L/h$ or $\pm\ 8{,}2\ \%$ from the mean, representing a significant disparity compared with Figure 17. This results in lenient boundaries on fluctuating data when increasing the IQR multiplier. The conclusion is that the current IQR method is inapplicable when working with flat data. Thus, should only be used on fluctuating data.

However, the IQR method can be improved by implementing configurable minimum IQR boundaries. This implies establishing boundaries which values must deviate by a specified percentage from the mean to be categorized as outliers. This means that outlying values must deviate beyond both IQR and a minimum percentage from the mean. Thus, enabling the IQR method to be used on flat data.

By implementing minimum percentage, and initially setting the minimum percentage from the mean to 2%, it is possible to set the IQR multiplier to 2,5. The result of the combined methods can be seen in Figure 19 and Figure 20 where no false positive anomalies were found. The reason for no anomalous values in Figure 19 is due to all values are within 2% from the mean.

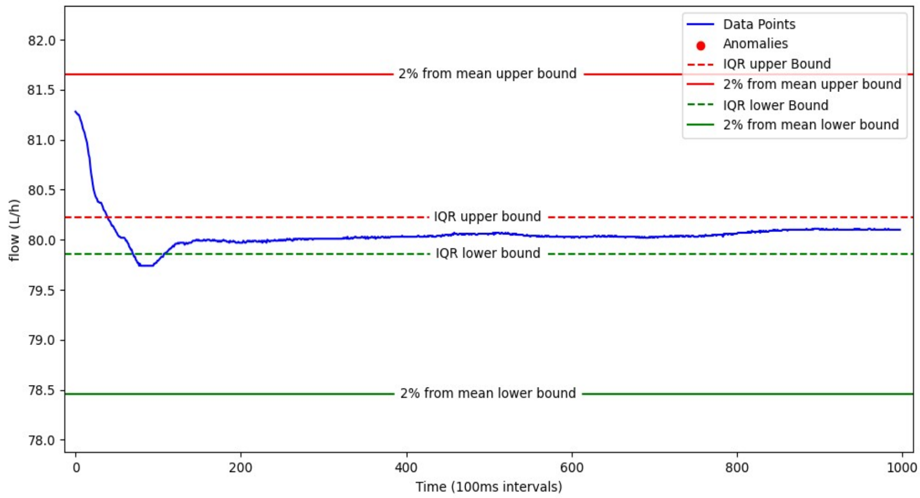

Figure 19: Improved IQR method on flat data series.

The reason for no anomalous values in Figure 20 even when deviating over 2% from the mean is due to all values are within the IQR boundaries.

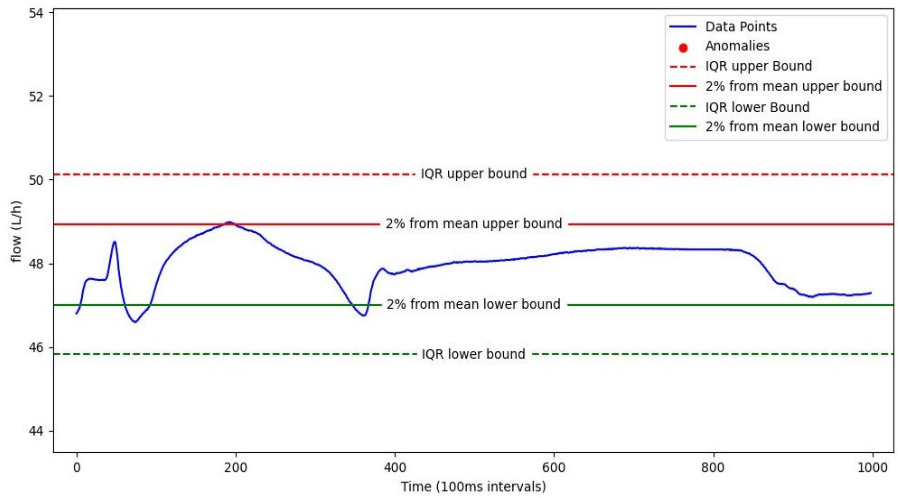

Figure 20: Improved IQR method on fluctuating data series.

The conclusion is that values classed as outliers must deviate beyond both IQR and a minimum percentage from the mean. This method using the boundaries furthest from the mean can be applied to a univariate data series represented by the variable Y using code example 3.

Code example 3. Establishing improved IQR boundaries using Python.

```
multiplier = 2.5
Outlier_min_acceptance_bound_percent = 2

# Calculate the first and third quartiles
Q1 = np.percentile(Y, 25)
Q3 = np.percentile(Y, 75)

# Calculate the IQR (Interquartile Range)
IQR = Q3 - Q1

# Set lower and upper bounds
IQR_upper_bound = Q3 + multiplier * IQR
IQR_lower_bound = Q1 - multiplier * IQR

Percent_upper_bound = mean + mean * Outlier_min_acceptance_bound_percent / 100
Percent_lower_bound = mean - mean * Outlier_min_acceptance_bound_percent / 100

# Set lower and upper boundary furthest from the mean
upper_bound = max(IQR_upper_bound, Percent_upper_bound)
lower_bound = min(IQR_lower_bound, Percent_lower_bound)
```

4 System architecture

The system architecture for the implemented flow anomaly detection application can be visualized using a component diagram in Unified Modeling Language (UML). This type of diagram is widely used and useful when working with software projects. A component diagram in UML is a visual representation that illustrates interconnections among various components. [12].

Components within this context include entities such as humans, software, and hardware. Envision components as modular building blocks, each encapsulating its contents. The behaviour of a component is shaped by the interfaces it provides and those it depends on. A component is like a blueprint, where the components are interconnected through interfaces. [12]. Comments on UML diagrams are inserted as seen in Figure 21.

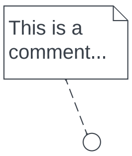

Figure 21: Comment in UML.

There are two types of interfaces as previously mentioned. Interfaces that are provided and interfaces that are required. Provided interfaces offer services or provide others. Required interfaces expect or depend on others. This means that the behaviour of the bank account example component in Figure 22 requires a person and a card to be able to provide bank and balance. [12].

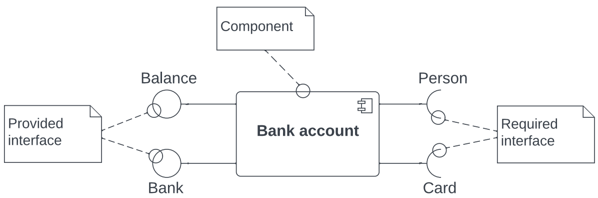

Figure 22: Internal structure of a component [12].

These two types of interfaces can be combined or used separately.

Figure 23 illustrates the component diagram designed for the flow anomaly detection software application.

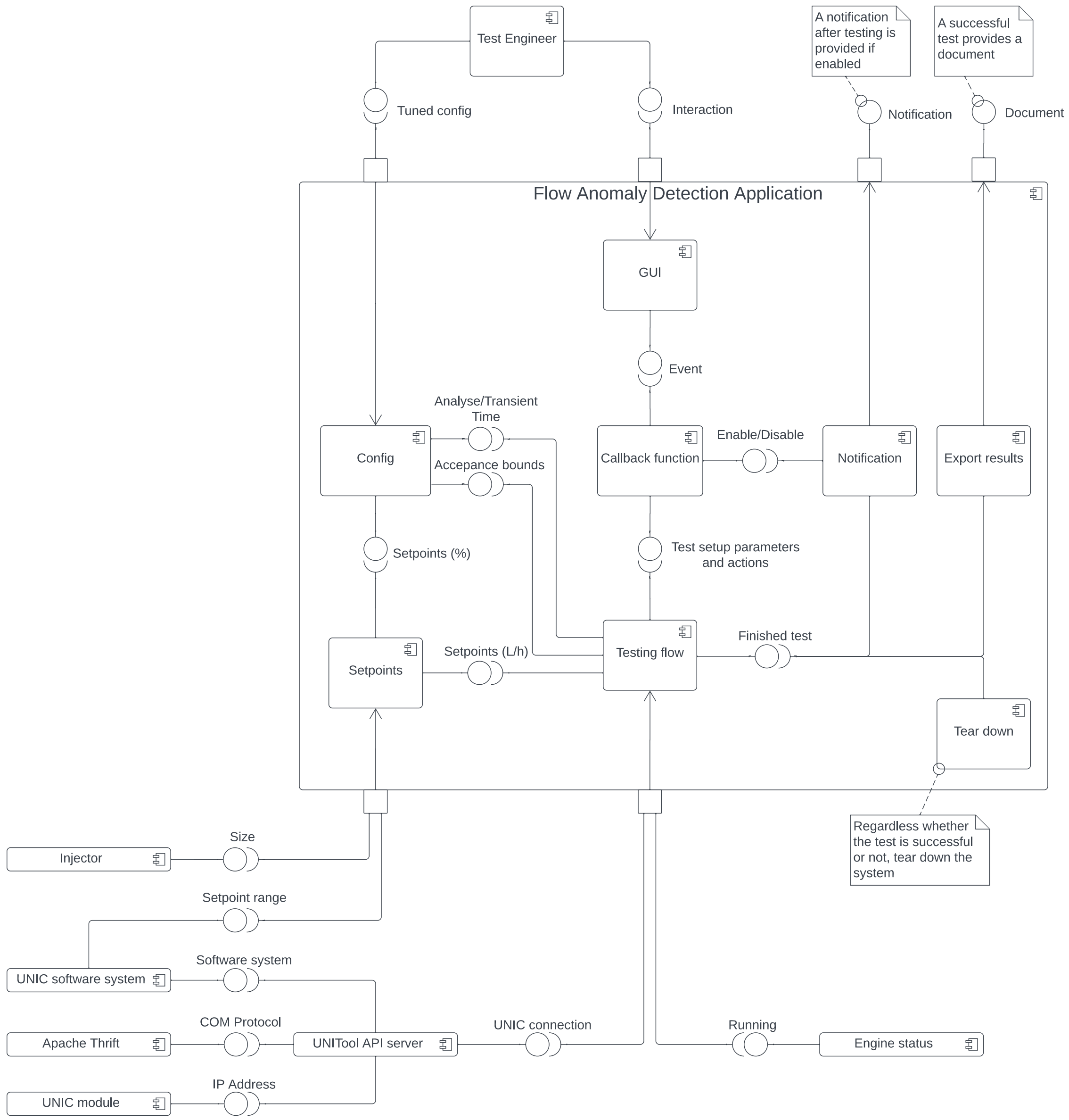

Figure 23: Component diagram of the flow anomaly detection application.

The heart of the component diagram is the flow testing. The required interfaces for testing the flow are the analysis time, transient time, acceptance bounds, and setpoints. The analyse time is the time to analyse each setpoint and transient time is the time for the system to stabilize after setpoint change. Transient time was also explained in Chapter 1.5.

The acceptance bound values are provided and later calculated by the flow testing. Setpoints are inserted as L/h. Also, test setup parameters and actions are a required interface for the flow testing component. This interface includes actions from the user such as settings selected by the user in the GUI and the user pressing the start button. The flow testing component itself then provides a notification, a finished flow test, and tears down the testing system. These components listed previously are also dependent on other components. Setpoints are for example dependent on the nozzle size. Keep in mind that the nozzle size itself does not provide this information directly, instead the size is selected in the GUI.

However, the setpoints are dependent on the size of the hardware. This is true for most hardware components where the user selects what hardware is used, like what dosing unit is connected, and what IP Address to the respective UNIC communication module should be used.

5 Discussion

Many different methods of anomaly detection, oscillation detection, stiction detection, and control loop performance have been studied in the development of the flow anomaly detection application in this bachelor’s thesis. The chosen methods align with the demand that was set.

This is a great tool for dosing unit performance validation that will be of use to FAT experts. There are always improvements that could be researched. This could be some kind of stiction modelling in addition to another research to decide if step-based setpoints are the preferred choice of test sequences.

There are lots of features that are irrelevant and left unmentioned even when implemented in this software application. These are graphical user interface development, exported test protocol development, the end-user experience, exception handling, instruction manual for the end-users, input processing, and image compression, to name a few. The hardest part was to implement the UNIC module communication with the UNITool API server.

There was also a request to use the same software to analyse pre-existing flow data. The software application can read data frames. Meaning it also can be used to analyse any CSV files. This was a simple implementation in the end.

The SCR industry is constantly striving to improve the reduction of NOx emissions, meaning the SCR products change. The software will need to be complemented someday. However, the software application itself is supported for all commissioned NOR dosing units by just changing parameters in the software application. Also, the configuration can be tuned for upcoming dosing units.

6 References

[1] ‘Wärtsilä NOx Reducer (NOR)’, Wartsila.com. Accessed: Sep. 30, 2023. [Online]. Available: https://www.wartsila.com/marine/products/engines-and-generating-sets/nox-abatement/wartsila-nox-reducer-nor

[2] O. Kröcher (Ed.), *Selective Catalytic Reduction of NOx*. MDPI - Multidisciplinary Digital Publishing Institute, 2018. doi: 10.3390/books978-3-03897-365-2.

[3] ‘Wärtsilä UNIC - Engine Control System Overview’. Accessed: Sep. 30, 2023. [Online]. Available: https://cdn.wartsila.com/docs/default-source/product-files/engines/brochure-o-e-unic-ms.pdf?sfvrsn=7ab39545_4

[4] J. W. V. Dambros, J. O. Trierweiler, and M. Farenzena, ‘Oscillation detection in process industries – Part I: Review of the detection methods’, *J. Process Control*, vol. 78, pp. 108–123, Jun. 2019, doi: 10.1016/j.jprocont.2019.04.002.

[5] P. Schneider and F. Xhafa, ‘Chapter 3 - Anomaly detection: Concepts and methods’, in *Anomaly Detection and Complex Event Processing over IoT Data Streams*, P. Schneider and F. Xhafa, Eds., Academic Press, 2022, pp. 49–66. doi: 10.1016/B978-0-12-823818-9.00013-4.

[6] M. Bacic, ‘On hardware-in-the-loop simulation’, in *Proceedings of the 44th IEEE Conference on Decision and Control*, Dec. 2005, pp. 3194–3198. doi: 10.1109/CDC.2005.1582653.

[7] L. Shang, ‘Valve Stiction Detection Method Based on Dynamic Slow Feature Analysis and Hurst Exponent’, *Processes*, vol. 11, no. 7, pp. 1913-, 2023, doi: 10.3390/pr11071913.

[8] P. Airikka, ‘PI Control for Load Disturbance Rejection’, in *Automaatiopäivät24, 13-14 April 2021, Virtual Event.*, Suomen Automaatioseura ry, 2021. Accessed: Aug. 04, 2023. [Online]. Available: http://www.theseus.fi/handle/10024/728609

[9] A. Smiti, ‘A critical overview of outlier detection methods’, *Comput. Sci. Rev.*, vol. 38, p. 100306, Nov. 2020, doi: 10.1016/j.cosrev.2020.100306.

[10] ‘numpy.fft.rfft — NumPy v1.26 Manual’. Accessed: Nov. 11, 2023. [Online]. Available: https://numpy.org/doc/stable/reference/generated/numpy.fft.rfft.html

[11] ‘Frekvensanalys kompendium’. Yrkeshögskolan Novia. Accessed: Nov. 15, 2023. [Online]. Available: https://moodlecourses.novia.fi/pluginfile.php/247997/mod_resource/content/0/Frekvensanalys_kompendium.pdf

[12] J. Osis and U. Donins, ‘Chapter 1 - Unified Modeling Language: A Standard for Designing a Software’, in *Topological UML Modeling*, J. Osis and U. Donins, Eds., in Computer Science Reviews and Trends. , Boston: Elsevier, 2017, pp. 3–51. doi: 10.1016/B978-0-12-805476-5.00001-0.