**Hamburg University of Applied Sciences**

**Faculty of Life Sciences**

# Development and Implementation of a Constraint Detection Tool for Industrial Process Data

Master's Thesis

M. Sc. Process Engineering

submitted by

**Lea Tiedemann**

[redacted]

Hamburg

December 1st, 2022

**Reviewer**: Prof. Dr. Margret Bauer (HAW Hamburg)

**Reviewer:** Prof. Nina Thornhill (Seeq Corporation)

The thesis was supervised and prepared in cooperation with the company Seeq Corporation.

## Affidavit

I, Lea Tiedemann [redacted], certify that I have written this thesis independently without outside help and have used only the indicated aids. Verbatim or in the sense of other works taken passages are marked with the source.

______________________________________________

## Abstract

Saturation occurs in the process industry for a variety of reasons. On the one hand, saturation can be caused by poor actuator dimensioning, external disturbances, and poor controller tuning. On the other hand, saturation can also be a deliberate way of plant operation. It becomes visible in the data captured from the process, in particular in the process variable and controller output signals. This thesis introduces a clear definition and distinction between saturation in the controller output and constraints in the actuator, sensor and setpoint. Two previously published detection methods for constraints and saturation in control loop data are discussed and a novel detection method is introduced. All three detection methods were tested on industrial process data where the newly developed method showed the highest robustness, a low computation time and high comprehensibility. With the newly developed method, a constraint and saturation detection tool was developed using Python as a programming language. The tool has a user-friendly interface, can process several hundred signals at once, and produces easy-to-interpret analysis results which makes it suitable for the use by engineers for control loop performance monitoring. The tool is publicly available on GitHub and PyPI. On PyPI, the tool can be found as a Python package with the name "seeq-constraintdetection" and can be installed as an add-on in the data analytics software Seeq.

# Table of Contents

## List of Figures

# List of Tables

## List of Abbreviations and Symbols

| Abbreviation | Meaning |
|---|---|
| ACF | Autocorrelation Function |
| CPM | Control Loop Performance Monitoring |
| DCS | Distributed Control System |
| ERP | Enterprise Resource Planning |
| HMI | Human Machine Interface |
| IAE | Integral Absolute Error |
| ICS | Industrial Control System |
| MES | Manufacturing Execution System |
| MIS | Management Information System |
| MPC | Model Predictive Control |
| MV | Manipulated Variable |
| $MV_{max}$ | Maximum of the Manipulated Variable |
| $MV_{min}$ | Minimum of the Manipulated Variable |
| OP | Controller Output |
| $OP_{max}$ | Maximum of the Controller Output |
| $OP_{min}$ | Minimum of the Controller Output |
| PID | Proportional-Integral-Derivative |
| PLC | Programmable Logic Controller |
| PV | Process Variable |
| $PV_{max}$ | Maximum of the Process Variable |
| $PV_{min}$ | Minimum of the Process Variable |
| SCADA | Supervisory Control And Data Acquisition |
| SI | Saturation Index |
| SP | Setpoint |
| $SP_{max}$ | Maximum of the Setpoint |
| $SP_{min}$ | Minimum of the Setpoint |
| UI | User Interface |

| Symbol | Meaning |
|---|---|
| $\alpha$ | Significance level of Kolmogorov-Smirnov Two-Sample Test |
| $c(\alpha)$ | Inverse Kolmogorov distribution at significance level $\alpha$ |
| d | Step size between two samples |
| D | D-statistic of the Kolmogorov-Smirnov Two-Sample Test |
| $D_{crit}$ | Critical value for the D-statistic of the Kolmogorov-Smirnov Two-Sample Test |
| $\Delta_{sat}$ | Saturation band |
| e(t) | Error over time |
| $\epsilon$ | Saturation band factor |
| $F(y_i \leq y)$ | Cumulative distribution function |
| h | Sampling period |
| $I_i$ | Idle Index |
| $K_C$ | Controller gain |
| l | Length of extended period of saturation |
| L | Sample length of time series data |
| m | Length of $S_1$ |
| $\mu_{\Delta x}$ | Mean value of $\Delta x$ |
| n | Length of $S_2$ |
| p | Valve position |
| $q_{\Delta x}$ | Regularity index for quantisation detection |
| r(t) | Setpoint over time |
| $sat_n$ | Extended period of saturation |
| $S_1$ | Random sample 1 |
| $S_2$ | Random sample 2 |
| $\sigma_{\Delta x}$ | Standard deviation of $\Delta x$ |
| t | time |
| $t_{pos}$ | Time periods during which the correlation of OP and PV is positive |
| $t_{neg}$ | Time periods during which the correlation of OP and PV is negative |
| $t_{vc}$ | Time a valve is more than 90 % or less than 10 % opened |
| $\tau$ | Time to carry out a setpoint change |
| $\tau_D$ | Derivative time |
| $\tau_I$ | Integral time |
| u(t) | Controller output over time |
| $u_b$ | Controller output bias |
| $x_i$ | Individual values in time series data |
| $\Delta x$ | Difference between two peaks in the process variable histogram |
| $X_{max}$ | Maximum of time series data |
| $X_{min}$ | Minimum of time series data |
| y(t) | Process variable over time |

# 1. Introduction

## 1.1 Background

Industrial process plants gather large amounts of data from sensors, controllers, and alarms. This data is valuable as it contains detailed process information. It is of high importance to make this information understandable for engineers by using data analysis tools to identify process faults and improve efficiency, resource use and product quality. Control loops are regularly evaluated by control engineers for their performance. This evaluation is called control loop performance monitoring (CPM) or sometimes control loop performance assessment. CPM uses statistical methods and fault detection methods to automatically detect poorly performing control loops with the objective to identify and correct the cause for the poor performance. (Jelali 2006; Bauer et al. 2016; Harris 1989). In the area of fault detection, the detection of oscillations and valve stiction are particularly important, but also the detection of sluggish control loops and quantisation (Bacci D. Capaci and Scali 2018; Bauer et al. 2019). These topics have also played an important role in research in recent decades. A less addressed topic in CPM is the detection of saturation and constraints in control loop signals although they are a common phenomenon in the process industry. For this reason, the state of research as well as existing detection methods for saturation and constraints will be investigated in this thesis.

The data analytics software Seeq was developed by Seeq Corporation for the use by engineers to analyse time-series data in the process industry. In this context, the software also provides tools that are suitable for CPM. These tools include the calculation of statistics such as the "number of setpoint changes per controller and day" for example, but also the creation of plant-wide asset trees in which signals can be assigned to specific assets e.g. a controller. Seeq is extensible and provides the user with the option to install additional tools, so-called add-ons, for special analyses and use cases. Seeq's add-ons comprise tools for valve stiction detection, causality analysis, correlation analysis and many more. No Seeq add-on is currently available for the detection of saturation and constraints in control loop data.

## 1.2 Aim of this Work

The aim of the thesis is the development of a Seeq add-on for the detection of constraints and saturation in industrial process data and the implementation of the add-on in the data analytics software Seeq. The add-on should import a Seeq asset tree via an intuitive user interface and provide the user with easy-to-interpret results. Therefore, an asset tree with control loop signals must be created from an industrial data set to develop and test the add-on. The analysis results should help the user to find constrained or saturated signals and quantify the extent of constraints and saturation so that the user can make decisions based on the results. In addition, a comprehensive add-on documentation should be created to assist Seeq users in working with the tool. To ensure the functionality of the add-on, a robust detection method needs to be found in the literature or must be newly developed. In order to find the most suitable detection method, a literature research is conducted. Furthermore, in the context of this work, a clear distinction between saturation and constraints in control loops will be established.

## 1.3 Structure of the Thesis

In Chapter 2, the architecture of industrial control systems (ICS), the fundamental principles of control loops and proportional-integral-derivative (PID) control are discussed. Furthermore, definitions of controller saturation and constraints in control loops are introduced. Subsequently, CPM is explained and fault detection methods for oscillations, valve stiction, sluggish control loops and quantisation are summarised. Based on this, methods for saturation detection are described in Chapter 3 and compared with each other. At the end of Chapter 3, the comparison of the methods will be used to identify the detection method which is most suitable for the implementation in the add-on. In Chapter 4, the embedding of Seeq in the ICS and the software use for CPM is described. Chapter 5 presents the add-on and its functionality and evaluates whether it meets all predefined requirements and whether further improvements could be made. Finally, Chapter 6 summarises the results of the thesis.

# 2. Background

Chapter 2 explains the architecture of industrial control systems (Section 2.1) and the fundamentals of PID control (Section 2.2) in order to introduce a definition and distinction between constraints and saturation in control loops in Section 2.3.1 and 2.3.2. In Section 2.3.3, causes for controller saturation are given and illustrated with examples from the process industry. Subsequently, CPM is discussed in Section 2.4 and an overview of detection methods for oscillations, valve stiction, sluggish control loops and quantisation is given in Sections 2.4.1 to 2.4.4.

## 2.1 Industrial Control Systems

To understand how process data is acquired in industrial plants it is important to understand the architecture of industrial control systems (ICS) and the different types of information which are gathered in industrial processes. The architecture of an ICS is illustrated in Figure 1. An ICS is composed of interconnected equipment used to monitor and control physical equipment in industrial environments. In contrast to commercial networks for corporate environments an ICS is strongly connected to real-world actions and conditions (Galloway and Hancke 2013; Hu et al. 2018). The architecture of an ICS is separated into multiple layers in a hierarchical way where each layer has its own functionality, communication protocols and physical standards (Galloway and Hancke 2013). In general, an ICS can be split into three layers: The enterprise management layer, the supervisory layer, and the field layer.

The enterprise management layer contains management information systems (MIS), enterprise resource planning (ERP) systems, manufacturing execution systems (MES) and other application systems. It connects to the internet and is used for real-time monitoring, process management and enterprise-level decision making. The supervisory layer contains process monitoring systems, historical and real-time databases, and operator and engineer stations. It is responsible for data transmission between the field layer and the enterprise management layer and for controlling field devices via human machine interfaces (HMI) (Hu et al. 2018; Galloway and Hancke 2013). Controllers, sensors, actuators, and other I/O devices are located in the field layer. In this layer, transmission of field information, manipulation of field devices and data exchange between devices through field bus systems take place. On the lowest level of the control system two-wiring signalling techniques are used such as 4-20 mA and 0-10 V (e.g. between the controllers and actuators) (Galloway and Hancke 2013).

Important components of an ICS are programmable logic controllers (PLC), distributed control systems (DCS) and supervisory control and data acquisition (SCADA) systems. PLCs are specialised, computer-based, solid-state electronic devices which can perform binary and analogue input and output and implement proportional, integral and derivative control loops (Galloway and Hancke 2013). A detailed explanation of proportional, integral, and derivative control is given in Section 2.2. PLCs can process several thousand I/O points (Ackerman 2017, 13). The terms SCADA and DCS are often used interchangeably (Ackerman 2017, 13ff.; Galloway and Hancke 2013). They refer to the control hardware in the field layer, the supervisory layer and the communication protocols in both layers (Galloway and Hancke 2013; Ackerman 2017, 13ff.). A main component of SCADA systems and DCS is the HMI in the supervisory layer. The HMI visualises the process and its control and process values as well as alarm events. Operators and engineers can change control values using the HMI (Ackerman 2017, 13).

In the field layer, different information types are acquired which are control information, diagnostic information, and safety information. Control information includes input and output information of control loops such as controller outputs (OP), manipulated variables (MV), process variables (PV), setpoints (SP) and auto manual information. Diagnostic information comprises sensory data which is not integrated into a control loop and is usually used to monitor plant health. Safety information contains alarm events and other information used to implement critical functions such as the shutdown of components. Time consistency of these information types must be guaranteed so that the chronological order of events can be interpreted correctly. For this reason, timestamps are used. All mentioned information types are stored on the data historian as time-discrete signals. In addition, the data historian can also contain calculated signals. (Galloway and Hancke 2013)

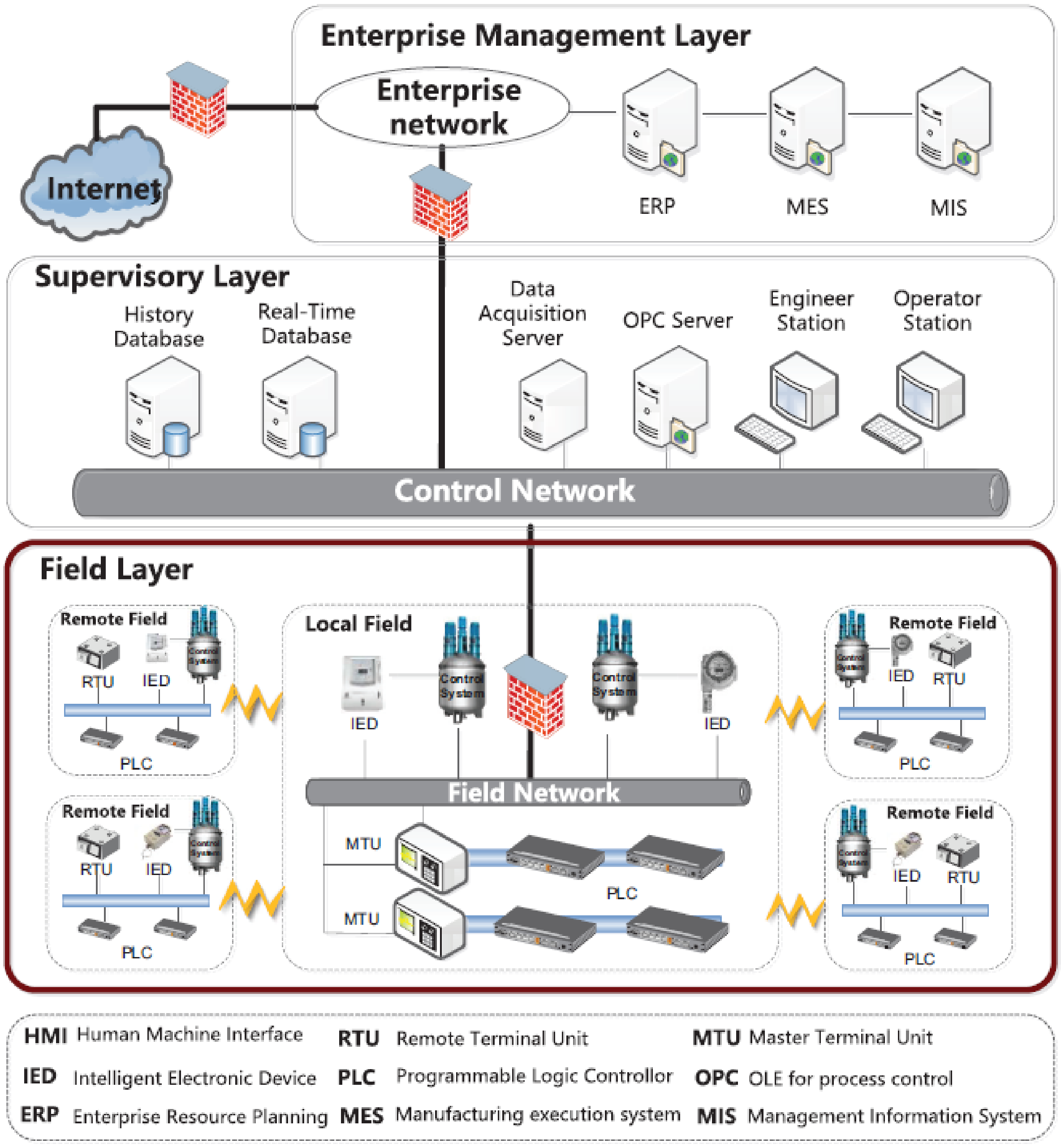


*Figure 1: Architecture of an industrial control system (Hu et al. 2018)*

The most common data historian technologies include the OSIsoft PI system, AspenTech InfoPlus 21 as well as products by Honeywell, ABB, Aveva, Siemens and other companies in the process automation sector. Many companies also build their own data historians which are usually SQL server applications. Since the amount of stored data is continuously growing, cloud platforms have been developed and resulted in products like Amazon AWS, Google Cloud Platform and Microsoft Azure. (Risse 2019)

## 2.2 PID Controllers

A control loop contains four main elements, namely the controller, actuator, process, and sensor as illustrated in Figure 2. The control loop aims to steer the process variable (PV) to the setpoint (SP). The process variable is continuously measured by the sensor and compared to the setpoint. The setpoint is usually set by the operator for regulatory control purposes. It can be adjusted infrequently or continuously. Cascade control and model predictive control (MPC) are examples where the setpoint is continuously adjusted. The controller calculates the error, which represents the difference between the process variable and the setpoint. Depending on the controller structure (e.g. PI, PID), the controller interprets the error and sends a controller output (OP) to the actuator, which in turn can influence the process via the manipulated variable (MV). The control loop is characterised by this closed signal process, whereby the controller function is to eliminate the error as quickly as possible or to keep it small.

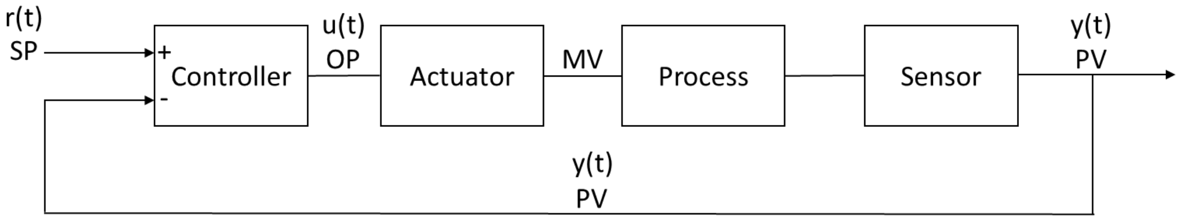


*Figure 2: Block diagram of a closed control loop*

The most common controller in the process industry is the proportional-integral-derivative (PID) controller due to its simplicity and satisfactory performance for a wide range of processes (Visioli 2006, 1).

$$e(t) = r(t) - y(t) \tag{2.1}$$

$$u(t) = K_c \cdot e(t) \tag{2.2}$$

The control error e(t) is defined in (2.1) as the difference between the setpoint r(t) and the measured process variable y(t). In proportional control, the controller output u(t) is proportional to the error e(t) as shown in (2.2). Depending on the controller gain $K_C$ the controller output is more or less sensitive to the error signal. The sign of $K_C$ determines whether u(t) increases or decreases with increasing error e(t). Proportional control has one significant disadvantage which is the inability to eliminate offset after a setpoint change or sustained load disturbance. The steady-state error cannot be reduced to zero regardless of the value of the controller gain. (Seborg et al. 1989, 185ff.)

$$u(t) = u_b + K_c \cdot e(t) \tag{2.3}$$

The steady-state error was the motivation to introduce the bias $u_b$ in (2.3) which can be a constant value or can be manually adjusted by the operator until the steady-state error is zero. Since the controller output equals the bias when the error is zero, the bias is set so that the controller output assumes its nominal steady-state value. In a flow control loop, for example, the bias is set so that the volume flow is at its nominal steady-state value. Therefore, the bias must be readjusted when a setpoint change occurs. This is inconvenient because operator intervention is required. For this reason, proportional control is usually used in combination with integral control. (Seborg et al. 1989, 186f.; Visioli 2006, 5)

$$u(t) = \frac{1}{\tau_I} \int e(t)\,dt \tag{2.4}$$

In integral control, the controller output depends on the integral of the error signal over time. The integral time $\tau_I$ is an adjustable control parameter in commercial controllers. Integral control overcomes the drawback of proportional control which is unable to eliminate offset. Considering equation (2.4) the controller output for integral control will change with time until the error signal is zero. While integral control has the advantage of eliminating offset, it has the disadvantage of integral wind-up which can lead to controller saturation and will be explained more detailed in Section 2.3.3. Proportional and integral control are mostly used in a combined way in a PI controller because integral control only reacts when the error has persisted for some time while proportional control takes immediate action. (Seborg et al. 1989, 188f.)

$$u(t) = \tau_D \frac{de}{dt} \tag{2.5}$$

Whilst integral control assesses the past changes in the error signal, derivative control provides information on future changes in the error by calculating its rate of change. In equation (2.5) $\tau_D$ is the derivative time. Since no control action occurs when the error is constant (de/dt = 0), derivative control cannot be used on its own but only in combination with proportional or proportional-integral control. (Seborg et al. 1989, 190)

$$u(t) = K_c \cdot \left( e(t) + \frac{1}{\tau_I} \int e(t)dt + \tau_D \frac{de}{dt} \right) \tag{2.6}$$

The ideal PID controller is formed by summing proportional, integral and derivative control as shown in equation (2.6).

$$u(t) = K_c \cdot \left( e(t) + \frac{1}{\tau_I} \int e(t)dt + \tau_D \frac{dy}{dt} \right) \tag{2.7}$$

For a setpoint change, the derivative term in (2.6) becomes very large and leads to a derivative kick. To avoid the derivative kick, the derivative control is based on the measurement of the process variable which gives equation (2.7). (Seborg et al. 1989, 191)

## 2.3 Saturation and Constraints

In a control loop, there are several components that can reach a limit because they have a minimum and maximum value as visualised in Figure 3. This phenomenon is commonly referred to as saturation (Bauer et al. 2016; Jelali 2013; Tarbouriech and Turner 2009). However, it is important to differentiate between saturation and constraints in the context of this work. Therefore, saturation and constraints in control loops will be defined in Sections 2.3.1 and 2.3.2. Causes for controller saturation will be given and explained in Section 2.3.3.

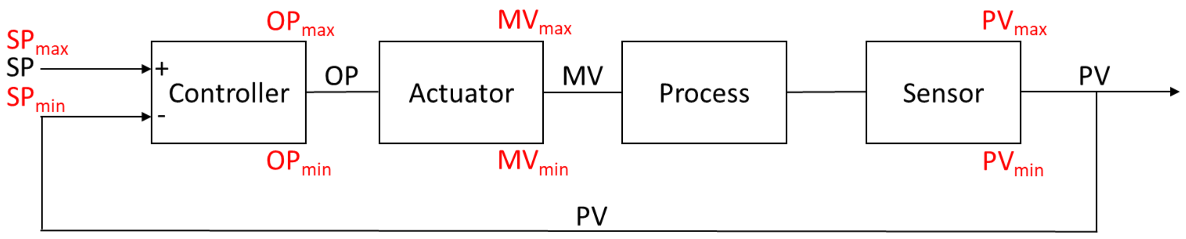


*Figure 3: Block diagram of a closed control loop with constraints*

### 2.3.1 Definition of Constraints in Control Loops

A constraint exists if a component in the control loop has inherent limits. This applies to the actuator, the sensor and in special cases also to the setpoint.

The actuator, which transfers the control signal to the process, has physical limits ($MV_{min}$ and $MV_{max}$ in Figure 3) and accordingly has inherent constraints. For example, a control valve can be fully open or fully closed. In the literature, it is referred to as actuator saturation or input saturation when the actuator reaches a constraint (Tarbouriech and Turner 2009; Saberi et al. 2000; Biswas et al. 2009). The constraints of the actuator must be considered during the control system design to avoid negative impact on the control loop performance (Tarbouriech and Turner 2009; Bernstein and Michel 1995).

The sensor has a defined measuring range, which in turn has a minimum and a maximum ($PV_{min}$ and $PV_{max}$ in Figure 3). Consequently, the sensor has constraints due to its inherent measuring range. In the literature, it is called sensor saturation or output saturation when the sensor reaches a constraint (Tarbouriech and Turner 2009; Kreisselmeier 1996). A common reason for sensor saturation is the choice of a cheap sensor with an inadequate measuring range to save costs (Tarbouriech and Turner 2009). Sensor saturation introduces a non-linearity into the control loop and can cause a deterioration in control loop performance and stability (Tarbouriech and Turner 2009). In addition, sensor saturation poses a safety risk because no reliable information about the process is available when the process variable lies outside of the sensor's measuring range.

Normally the setpoint of a control loop is a constant value if the controller is in automatic mode. If the controller is in manual mode, steps in the time trend can be seen at times when an operator has changed the setpoint. In these two cases, there are no constraints in the setpoint. In certain control structures, however, constraints can be implemented for the setpoint ($SP_{min}$ and $SP_{max}$ in Figure 3). This is for example the case with constrained MPC (Wade 2004, 329). Constrained MPC considers the constraints of actuators and process variables in order to control the process within predefined limits (Wade 2004, 329). This means that the setpoints specified by the MPC controller can also be constrained. In the process industry, it is often the case that the most economic point of operation is a constraint or a combination of constraints (Wade 2004, 330). MPC can be used to drive the system towards these constraints for optimal economic operation. As a result, the setpoint will often reach a constraint. This in turn leads to the process variable also looking constrained. This behaviour can be intentional if the limit of the process variable represents the operational optimum (as already described above). However, it can also be disadvantageous or even unsafe to drive process variables to their limits.

### 2.3.2 Definition of Saturation in Control Loops

Controller saturation occurs when the actuator reaches a constraint and consequently causes the controller to become saturated. The limits are therefore not inherent to the controller but are caused by the constrained actuator. The theoretical controller output can assume any value. However, the actual controller output is limited to 4-20 mA or 0-10 V since the actuator cannot interpret higher or lower values. The behaviour of the actual controller output is illustrated for a proportional controller in Figure 4 where the controller output is shown in dependence of the error. If the error becomes very large or very small, the theoretical controller output would exceed the limits of the actuator. Therefore, the actual controller output is limited to $OP_{min}$ and $OP_{max}$. This results in a non-linear relationship between the error and the actual controller output (Seborg et al. 1989, 186f.).

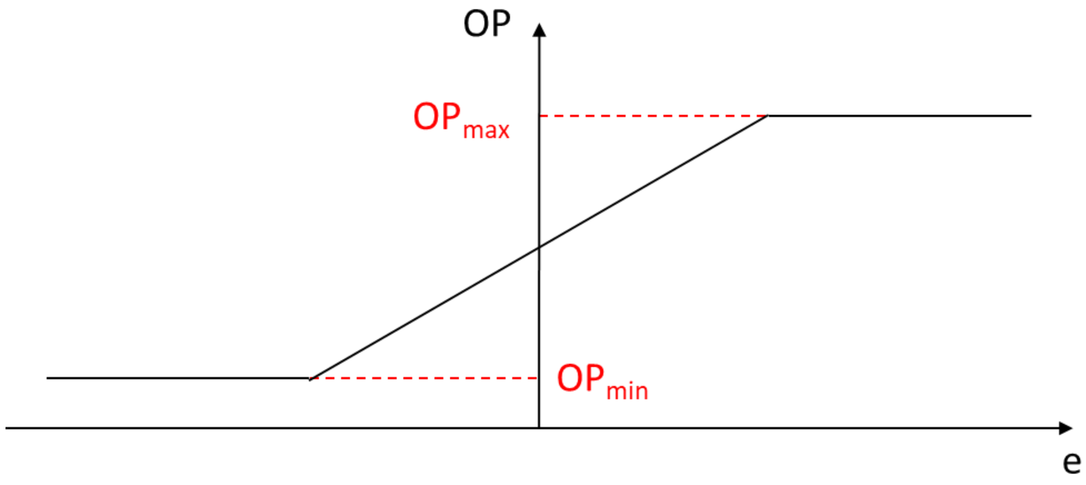


*Figure 4: Proportional controller non-linearity (Seborg et al. 2004)*

In the context of this work, saturation is defined as controller saturation and is characterised by the controller output being at its minimum $OP_{min}$ or maximum $OP_{max}$ and only deviating from there for short time periods (Bauer et al. 2016; Jelali 2013, 256; Matsuo et al. 2004). During a survey in the process industry 30 % percent of the participants stated that controller saturation occurs frequently (Bauer et al. 2016). Although controller saturation is a common phenomenon, it receives little attention in the literature. This is possibly because controller saturation is not always considered as a problem but is an intended operating mode of the plant which is explained in more detail in Section 2.3.3. Nevertheless, there are cases where controller saturation indicates poor control loop or process performance, and action is required to reduce or eliminate controller saturation. These cases are addressed in Section 2.3.3. In summary, the extent to which the effect of saturation needs to be accounted for in the control system design process depends on the required control system performance in relation to the capacity of the actuators as well as the level of expected disturbances (Bernstein and Michel 1995).

### 2.3.3 Causes for Saturation

In the following, the causes for controller saturation will be explained and illustrated with examples. Based on the process data, it is not possible to distinguish between the different causes for saturation because the individual control loop and the underlying process strongly influence the data. Therefore, no recommendation can be given on how to identify the cause of saturation based on the process data alone since detailed process knowledge is required for the cause identification.

**Poor Actuator Dimensioning**

Poorly dimensioned actuators can cause controller saturation when the process requirements are beyond the range of the manipulated variable (Jelali 2013, 256; Bauer et al. 2016; Jämsä-Jounela et al. 2003). In this case, the physical limits of the actuator prevent the controller from reducing the error to zero (Seborg et al. 1989, 189). Consequently, the controller saturates. Inadequate actuators should be replaced to increase control loop performance and fulfil the process specifications. An example for poorly dimensioned actuators could be a control valve in a flow control loop which is too small for the design specifications. The valve is too small to supply the volume flow specified by the controller and eventually the controller saturates due to the persisting error.

**External Disturbances**

Large disturbances push the process into a state it was not designed for and can lead to saturation (Bernstein and Michel 1995; Bauer et al. 2019). The sequence of events which lead to controller saturation is similar to the case of poorly dimensioned actuators because the process operation and the actuator dimensioning do not match. The actuator saturates due to its physical limits and consequently the controller saturates as the error cannot be reduced to zero. An example could be a cooling circuit that continuously cools a medium. If the ambient temperature is extremely high, this represents an external disturbance for the cooling circuit. When the cooling capacity of the circuit reaches the maximum limit without reaching the target temperature of the medium, saturation occurs.

**Change of operation away from design specifications**

When a system is operated away from its original design specifications, the control loop is no longer capable of fulfilling the requirements placed on it. Eventually, this can lead to controller and actuator saturation (Bauer et al. 2019), or even sensor saturation. In this case, the control loop has to be adapted to the new requirements in terms of controller tuning, actuator dimensioning and sensor choice.

**Poor controller tuning**

Poor controller tuning can lead to controller saturation (Jelali 2013, 256; Bauer et al. 2016). When the equation for a PID controller in (2.7) is considered, large values for the controller gain $K_c$ or small values for the integral time $\tau_I$ can cause a very large or small controller output even when the error is small. As a result, the controller output jumps to its minimum $OP_{min}$ or maximum $OP_{max}$ for small control errors and the controller saturates. If poor tuning settings cause the controller to become saturated, the tuning settings should be readjusted.

**Intended Saturation**

All the above-mentioned reasons for saturation are causes for evaluating the control loop and the plant operation and adjusting the system if necessary. On the other hand, saturation is often a deliberate mode of plant operation. In such cases no action has to be taken to reduce the occurrence of saturation.

Mid-range control is an example where saturation occurs frequently and does not represent a problem. Mid-range control is exemplarily illustrated in Figure 5 for a pump and a control valve in a flow control loop. The control loop has two manipulated variables, the pump speed and the control valve position, and one process variable which is the flow through the pipe. Both manipulated variables are used to regulate the flow. The pump makes large adjustments to the flow and the control valve makes small adjustments. To avoid unnecessarily high pressure losses, the valve should always be as wide open as possible. Consequently, the controller output for the control valve will saturate often. (Hägglund 2019)

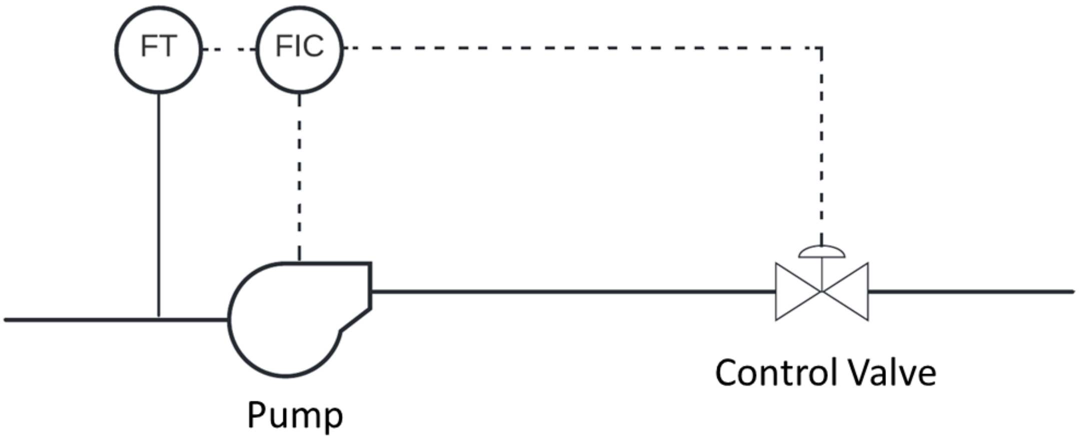


*Figure 5: Mid-range control*

**Integral wind-up**

Engineers often associate controller saturation with integral wind-up (Jelali 2013, 256; Bauer et al. 2016). However, this association is wrong because modern distributed control systems (DCS) include anti wind-up schemes which prevent integral wind-up (Seborg et al. 1989, 189f.). Therefore, integral wind-up cannot occur in modern DCS. In addition, integral wind-up does not cause controller saturation but increases the time the controller remains saturated when the actuator reaches a constraint. Nevertheless, integral wind-up will be explained in the following to clarify the connection to controller saturation.

Integral wind-up requires that the controller structure includes an integral term and that no anti wind-up scheme is implemented. When a sustained error occurs (e.g. after a large setpoint change or start-up of a batch process) the integral term becomes large and continues building up until the error changes sign and the integral term starts to decrease. During the build-up of the integral term the controller output can become saturated. The controller stays saturated until the integral term becomes zero. (Seborg et al. 1989, 189f.)

Integral wind-up is illustrated in Figure 6 as a step response to a setpoint change. The manipulated variable reaches a constraint at $MV_{max}$=1.5 immediately after the setpoint change, as the integral term and therefore also the controller output shoot up. This illustrates how the controller saturation is initially caused by the constraint of the manipulated variable. The process variable increases and reaches the setpoint at time 15. At this point, however, the integral term is still so large that the controller output and the manipulated variable remain saturated. Therefore, the integral wind-up increases the time the controller remains saturated. Since the error has now changed sign, the integral term starts to become smaller. At time 25 the manipulated variable is no longer saturated and at time 45 the setpoint is reached. The figure illustrates how high overshoots and long settling times can be caused by integral wind-up and why saturation due to wind-up should be avoided.

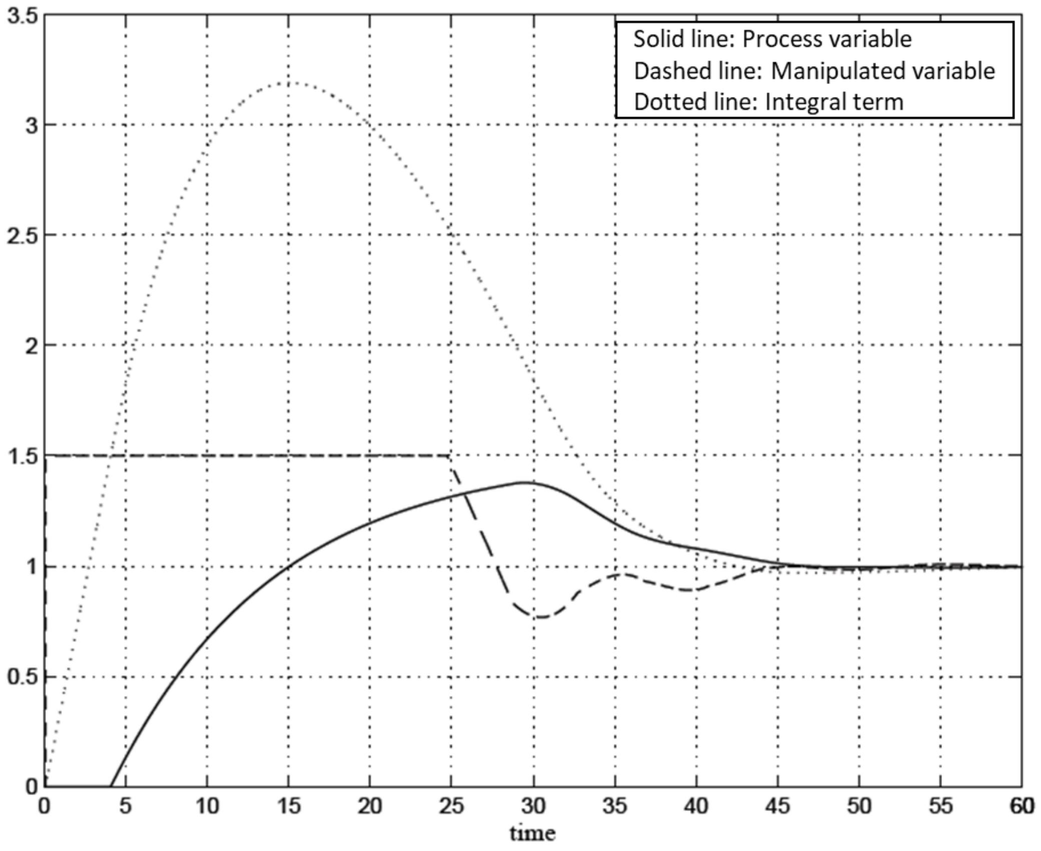


*Figure 6: Setpoint step response illustrating integral wind-up (Visioli 2006, 36)*

## 2.4 Data Analytics in Control Loop Performance Monitoring

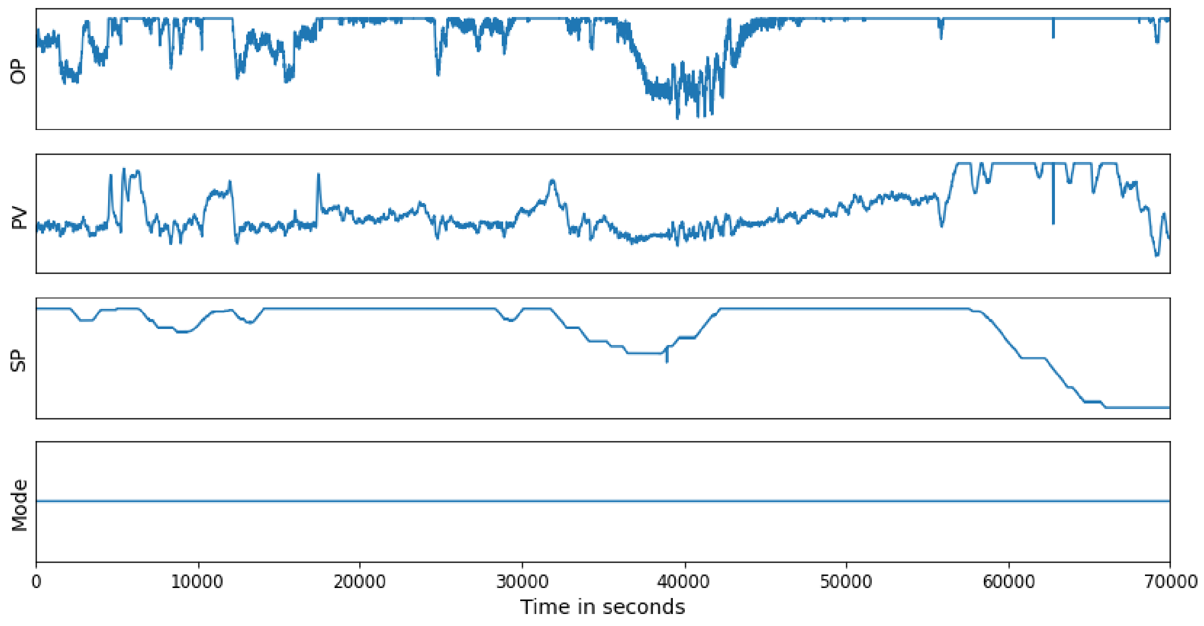


*Figure 7: Signals of a level control loop with saturation and constraints (normalised data)*

Control loop performance monitoring (CPM) applies statistical methods and fault detection methods to control loop data with the objective to identify poorly performing control loops and to develop improvement measures. An example for control loop data is given in Figure 7 including the controller output, process variable, setpoint and auto manual mode of a level control loop. The main reasons for poorly performing loops are linked to the controller (e.g. inadequate tuning), the actuator (e.g. valve stiction, other actuator faults) or the general control or process design (Jelali 2006; Bauer et al. 2016). These reasons were already present decades ago and are still present today (Bauer et al. 2016). Poor control loop performance can lead to increased process variability, increased energy consumption, increased operating costs, accelerated equipment wear and control system instability (Dambros et al. 2019; Forsman and Stattin 1999; Thornhill et al. 2003; Akavalappil and Radhakrishnan 2022; Bacci D. Capaci and Scali 2018). This created the need for CPM methods. The first method to identify poorly performing control loops was the Harris index proposed by Harris in 1989 (Harris 1989). Since then, many CPM methods, fault detection methods and commercially available CPM tools have been developed. CPM is executed by control engineers, with one control engineer being responsible for hundreds of control loops (Bauer et al. 2016). The high number of control loops per engineer reinforces the need for CPM methods and tools. Jelali proposed a CPM procedure consisting out of five steps (Jelali 2006):

1. Analysis of control loop data and calculation of performance figures
2. Selection and design of benchmark for performance assessment
3. Assessment and detection of poorly performing loops
4. Diagnosis of the underlying causes
5. Suggestion of improvement measures

For the first three steps, most engineers prefer simple mathematical statistics (Bauer et al. 2016). These include operating mode statistics, crossing of minimum and maximum thresholds, mean and standard deviation. But there are also more advanced techniques which include algorithmic indices (e.g. Idle index (see Section 2.4.3), integrated absolute error (see Section 2.4.1)) and multivariate statistics (e.g. principal component analysis, partial least squares) (Bauer et al. 2016). The diagnosis of the underlying causes of the poorly performing control loop is often "detective work", but there are many methods which detect specific faults in control data time trends and simplify/accelerate the diagnosis procedure (Jelali 2006; Bauer et al. 2019). Fault detection methods comprise amongst others oscillation detection, valve stiction detection, sluggish control loop detection and quantisation detection. These topics will be explained in Sections 2.4.1, 2.4.2, 2.4.3 and 2.4.4. The detection of saturation will be explained separately in more detail in Chapter 3 since it is the focus of this work.

### 2.4.1 Oscillation Detection

Oscillations are a frequent problem in the process industry and can lead to reduced plant productivity, increased process variability, energy consumption and operating costs and accelerated equipment wear (Dambros et al. 2019; Forsman and Stattin 1999; Thornhill et al. 2003; Zakharov and Jämsä-Jounela 2014; Miao and Seborg 1999). They can occur in the controller output, process variable and manipulated variable. Oscillations can be caused by incorrect controller tuning, external oscillatory disturbances, process non-linearities including valve stiction and multi-loop interactions (Dambros et al. 2019; Hägglund 1995; Miao and Seborg 1999). Since they occur frequently and have a negative impact on the process, oscillations should be detected, diagnosed and eliminated. For this reason, several methods for oscillation detection have been proposed and some of them will be explained in the following.

Oscillation detection methods can be divided into three categories which are time-domain methods (e.g. based on integrated absolute error (IAE) or autocorrelation function (ACF)), frequency-domain methods (e.g. fast Fourier transform or discrete cosine transform) and hybrid methods. Hägglund developed the first oscillation detection method based on the IAE of the control error e(t) between successive zero crossings (Hägglund 1995). High values for the IAE indicate an oscillation and if a threshold is exceeded, oscillation is detected. Thornhill and Hägglund enhanced the method by introducing a regularity factor to assess the oscillation period (Thornhill and Hägglund 1997). Forsman and Stattin considered the IAE and the time between zero crossings to detect oscillations (Forsman and Stattin 1999). In addition, their method differentiates between positive and negative IAEs to take asymmetric oscillations into account. If the values for consecutive oscillation fragments are similar, oscillation is present.

In 1999, a new method for excessive oscillations based on the ACF of the oscillation was introduced by Miao and Seborg (Miao and Seborg 1999). Their technique considers the decay ratio of the ACF. If the decay ratio exceeds a threshold, oscillation is detected. Thornhill et al. developed another method based on the ACF (Thornhill et al. 2003). The method evaluates the regularity of the oscillation period of the ACF and is able to detect multiple oscillations of different frequencies in the same signal. Srinivasan et al. use a hybrid method with empirical mode decomposition to decompose a signal into multiple components of different frequencies and apply a zero crossing technique to the resulting components (Srinivasan et al. 2007). Zakharov and Jämsä-Jounela identify oscillation periods by finding peaks in the time series signal (Zakharov and Jämsä-Jounela 2014). Subsequent oscillation periods are correlated, and an index is calculated based on the correlation. Oscillation is detected if the correlation index lies above a threshold.

Oscillation diagnosis is closely related to valve stiction detection since the oscillations caused by valve stiction have characteristic features. These features include a triangular or square waveform for the process variable and the controller output (Akavalappil and Radhakrishnan 2022). For other oscillation causes, the signals are mostly sinusoidal. Therefore, methods for oscillation diagnosis can differentiate between oscillations caused by valve stiction and oscillations caused by other reasons such as external disturbances or poor tuning. Consequently, oscillation diagnosis is a question of detecting valve stiction. Valve stiction detection is discussed in more detail in the following Section 2.4.2.

### 2.4.2 Valve Stiction Detection

Valve stiction is the term used to describe static friction of the valve stem (Choudhury et al. 2005). When the control valve receives an input to change the valve stem position, the stiction will cause an initial resistance to the applied force so that the valve stem does not move. This is followed by an abrupt movement of the valve stem called slip-jump. After the slip-jump, the valve stem moves smoothly. Valve stiction leads to oscillations and the associated negative impacts as already discussed in 2.4.1. The negative impacts include quality variations, accelerated equipment wear and control system instability which can ultimately affect the plant economy (Akavalappil and Radhakrishnan 2022; Bacci D. Capaci and Scali 2018). Therefore, valve stiction must be diagnosed as early as possible and appropriate actions must be taken by the plant operators to eliminate stiction (Bacci D. Capaci and Scali 2018; Akavalappil and Radhakrishnan 2022). An industrial example is given in Figure 8 for a pressure control loop where the characteristic features of stiction can be seen in the time trend. The OP signal shows a triangular wave form, and the PV measurement shows abrupt changes caused by the slip-jump of the valve stem.

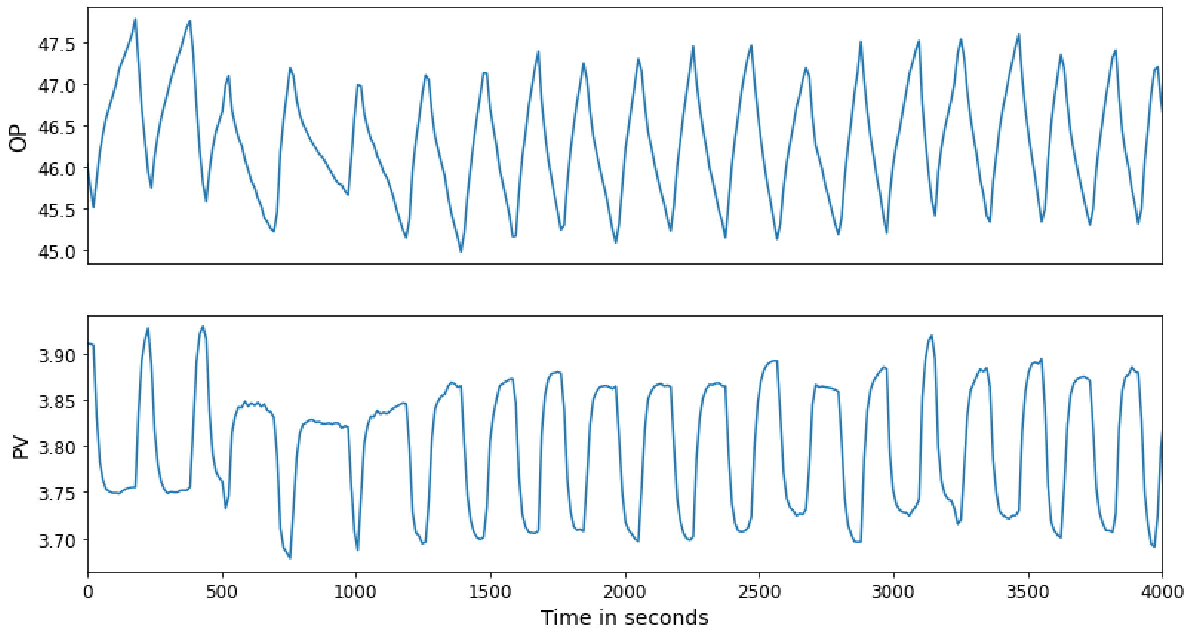


*Figure 8: Valve stiction in a pressure control loop (South African Council for Automation and Control)*

Many methods for valve stiction detection have been developed and they can be classified into five main categories: Cross-correlation function-based methods, limit cycle pattern-based methods, non-linearity detection-based methods, waveform shape-based methods and artificial neural network or machine learning methods (Bacci D. Capaci and Scali 2018; Akavalappil and Radhakrishnan 2022). Cross-correlation function-based methods calculate the cross-correlation function between the process variable and the controller output (Horch 1999). The cross-correlation function is an odd function when valve stiction is present and an even function when the oscillation is caused by another reason. Limit cycle pattern-based methods use characteristic features in the MV/OP plot or in the PV/OP plot to detect stiction (Kano et al. 2004; Yamashita 2006; Scali and Ghelardoni 2008). In a previous thesis at the Hamburg University of Applied Sciences, a limit cycle pattern-based method was developed based on the shape of the Error/OP plot (Essinger). Non-linearity detection-based methods take advantage of the fact that valve stiction is a process non-linearity and detect stiction by detecting non-linearities (Choudhury et al. 2004; Thornhill 2005). Waveform shape-based methods use the relationship between measured values and a reference waveform (triangular wave function, square wave function, sinusoidal function) (He et al. 2007; Srinivasan et al. 2005; Rossi and Scali 2005).

In addition to the high number of valve stiction detection methods, there are several methods and publications on the topics of valve stiction quantification and valve stiction modelling. In the context of this work, it is too extensive to discuss these methods. A detailed review can be found in (Bacci D. Capaci and Scali 2018) and (Akavalappil and Radhakrishnan 2022).

### 2.4.3 Sluggish Control Loop Detection

Incorrect controller tuning is one of the main reasons for poor control loop performance. Wrong tuning settings can result in oscillations if the tuning is too aggressive or in sluggish behaviour if the controller was tuned too conservative (Bauer et al. 2016; Hägglund 1999). Conservative controller tuning does not lead to any oscillations or overshoots but to sluggish behaviour which results in long and large deviations from the setpoint when a load disturbance occurs (Hägglund 1999).

Hägglund introduced the Idle index to detect sluggish control loops (Hägglund 1999). This method was improved by Horch and Kühl by developing a signal pre-treatment method to make the Idle index more reliable (Horch and Kühl 2003). For the calculation of the Idle index the following requirements must be met:

- The controller gain sign must be known.
- Noise-filtered controller output and process variable data must be available.
- The control loop is only subjected to load disturbances while the setpoint remains constant.

The Idle index $I_i$ evaluates the time periods where the correlation of the OP and PV signal increments is positive and negative.

$$
\begin{aligned}
t_{pos} &= \begin{cases} t_{pos} + h & if\ \Delta OP \Delta PV > 0 \\ t_{pos} & if\ \Delta OP \Delta PV \ \leq 0 \end{cases} \\
t_{neg} &= \begin{cases} t_{neg} + h & if\ \Delta OP \Delta PV < 0 \\ t_{neg} & if\ \Delta OP \Delta PV \ \geq 0 \end{cases}
\end{aligned}
\tag{2.8}
$$

$$
I_i = \frac{t_{pos} - t_{neg}}{t_{pos} + t_{neg}} \tag{2.9}
$$

The time periods with positive correlation $t_{pos}$ and negative correlation $t_{neg}$ are calculated as shown in (2.8) and updated for every sampling period h. The Idle index $I_i$ is then calculated as shown in (2.9). The value for the Idle index can lie between -1 and 1. For a positive controller gain, a value close to 1 means that the control loop is sluggish. A value close to -1 indicates a well-tuned loop or an oscillating loop. Since the index cannot differentiate between well-tuned and oscillating loops, Hägglund suggests combining the Idle index with an oscillation detection method (Hägglund 1999). Furthermore, the Idle index is sensitive to noise because the signal increments are considered. To overcome the noise sensitivity, the controller output and process variable must be pre-treated. Horch and Kühl developed a pre-treatment method consisting of three steps (Horch and Kühl 2003). In the first step, steady-state periods in the signals are detected and replaced with the mean of the corresponding period. In the second step, the signals are filtered using a low-pass filter. In the last step, quantisation is applied to the filtered signals to further reduce the noise.

### 2.4.4 Quantisation Detection

Quantisation can be seen as a sensor fault and is caused by sensors with low resolution or poorly adjusted sensors leading to step-like features in the time trend of the process variable (Bauer and Madolo 2007; Bauer et al. 2016; Bauer et al. 2019). An example for quantisation is shown in Figure 9 for a temperature measurement. The quantised time trend is a poor representation of the physical process and data analysis performed on quantised data can give misleading results (Bauer and Madolo 2007). For this reason, quantised measurements should not be used for data analysis and sensors with high quantisation levels should be re-calibrated. Bauer and Madolo developed a quantisation detection method which uses the regularity of peaks in the process variable histogram to measure the severity of quantisation (Bauer and Madolo 2007).

$$q_{\Delta x} = \frac{\mu_{\Delta x}}{\sigma_{\Delta x}} \tag{2.10}$$

First, the intervals Δx between two peaks in the histogram are calculated. In the next step, the regularity index $q_{\Delta x}$ is calculated from the mean $\mu_{\Delta x}$ and standard deviation $\sigma_{\Delta x}$ of the intervals Δx as shown in (2.10). The process variable is considered as quantised if $q_{\Delta x} > 3$. The quantisation level can be approximated by $\mu_{\Delta x}$.

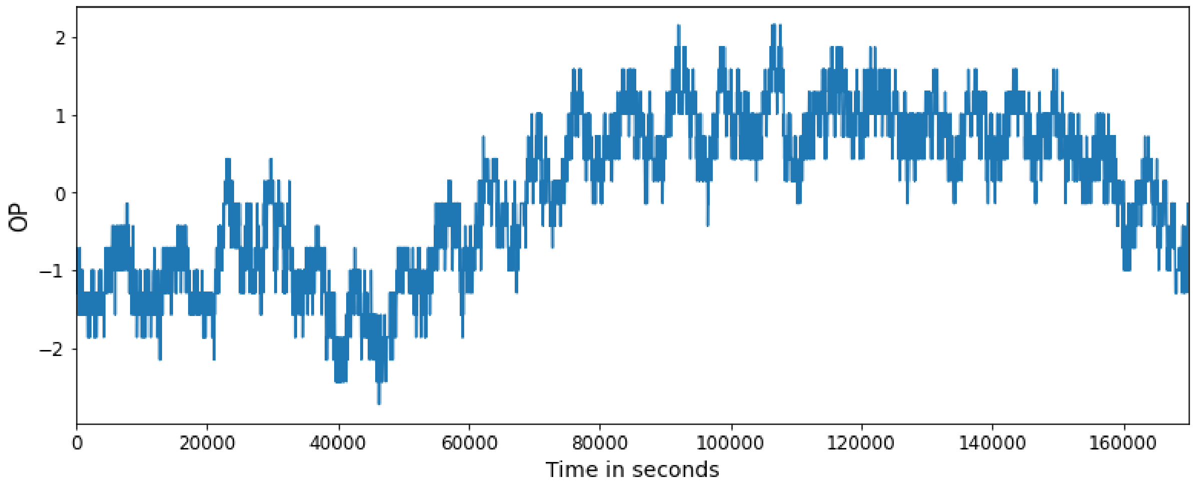


*Figure 9: Quantisation in a temperature measurement (normalised data) (South African Council for Automation and Control)*

# 3. Saturation Detection

## 3.1 Detection Methods

Controller saturation can be detected easily by visual inspection because the time trend of the controller output signal is flat and constrained (Jelali 2013, 256; Matsuo et al. 2004). Since visual inspection of hundreds of control loops is very time-consuming and therefore not feasible, saturation detection methods have been developed and implemented into industrial tools (Bauer et al. 2019; Matsuo et al. 2004). In contrast to the heavily researched fields of oscillation detection and valve stiction detection, only three methods for saturation detection have been proposed in the literature. This is because saturation detection does not require a complex method and is therefore less suitable as a research topic. The existing methods will be explained in the following and a modified method will be proposed.

### 3.1.1 Saturation Index by Bauer et al.

Bauer et al. developed a simple saturation detection method which considers the number of samples that lie close to the minimum $OP_{min}$ or maximum $OP_{max}$ of the controller output (Bauer et al. 2019).

$$X_{min} = min\{[x_1 \dots x_L]\} \tag{3.1}$$

$$X_{max} = max\{[x_1 \dots x_L]\} \tag{3.2}$$

The minimum $X_{min}$ and maximum $X_{max}$ of the investigated data series of length L are determined as in equation (3.1) and (3.2). The individual samples are referred to as $x_i$ where $x_1$ is the first sample and $x_L$ is the last sample of the data series.

$$\Delta_{sat} = \epsilon \cdot (|X_{max} - X_{min}|) \tag{3.3}$$

A sample lies close to the maximum or minimum if it is within the saturation band $\Delta_{sat}$ below the maximum or above the minimum as shown in equation (3.3). The factor ϵ was chosen to be 0.005 by Bauer et al. (Bauer et al. 2019).

$$X_{max} - \Delta_{sat} < x_i \leq X_{max} \tag{3.4}$$

$$X_{min} \leq x_i < X_{min} + \Delta_{sat} \tag{3.5}$$

Following from that any sample $x_i$ which fulfils one of the conditions shown in (3.4) and (3.5) is considered as a sample which lies close to the controller output limits. The range $\Delta_{sat}$ as well as the conditions (3.4) and (3.5) are visualised in Figure 10. Saturation $sat_n$ is detected if at least l=6 consecutive samples fulfil the same condition (3.4) or (3.5). Periods of time which comprise at least six samples close to the upper or lower controller output limit are defined as extended periods of saturation. The minimum length l=6 of the extended period of saturation was chosen by Bauer et al. because it gave the lowest number of false results when they first tested the method on industrial data (Bauer et al. 2019).

$$SI = \frac{\sum sat_n}{L} \tag{3.6}$$

The saturation index (SI) is calculated as the number of samples of all periods of extended saturation divided by the total length L of the investigated measurement as shown in (3.6). Therefore, the calculated index lies between 0 and 1 where index 1 would describe a constant value and index 0 describes a signal without saturation.

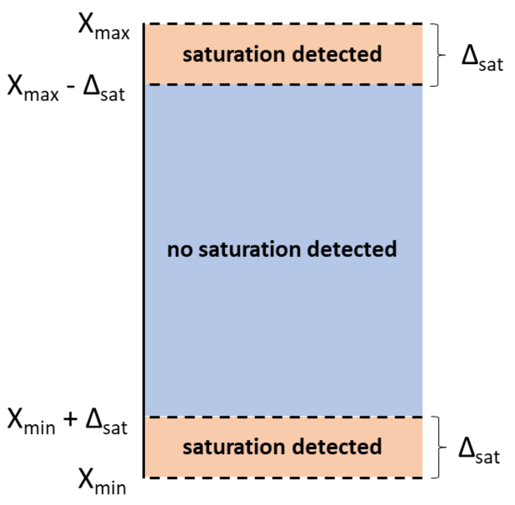


*Figure 10: Visualisation of the saturation detection method by Bauer et al. (Bauer et al. 2019)*

### 3.1.2 Modified Version of Saturation Index by Bauer et al.

Bauer et al. reported false positives when they first tested their method on industrial data (Bauer et al. 2019). The false positives occurred in data which contained a setpoint change. The corresponding OP signals are shown in Figure 11. Both signals have local extrema which are misinterpreted as saturated periods by Bauer's method. To eliminate the false positives, Bauer et al. filtered out data with setpoint changes. Since filtering out all data which contains a setpoint change is inconvenient for a detection method, a modification of Bauer's method will be proposed.

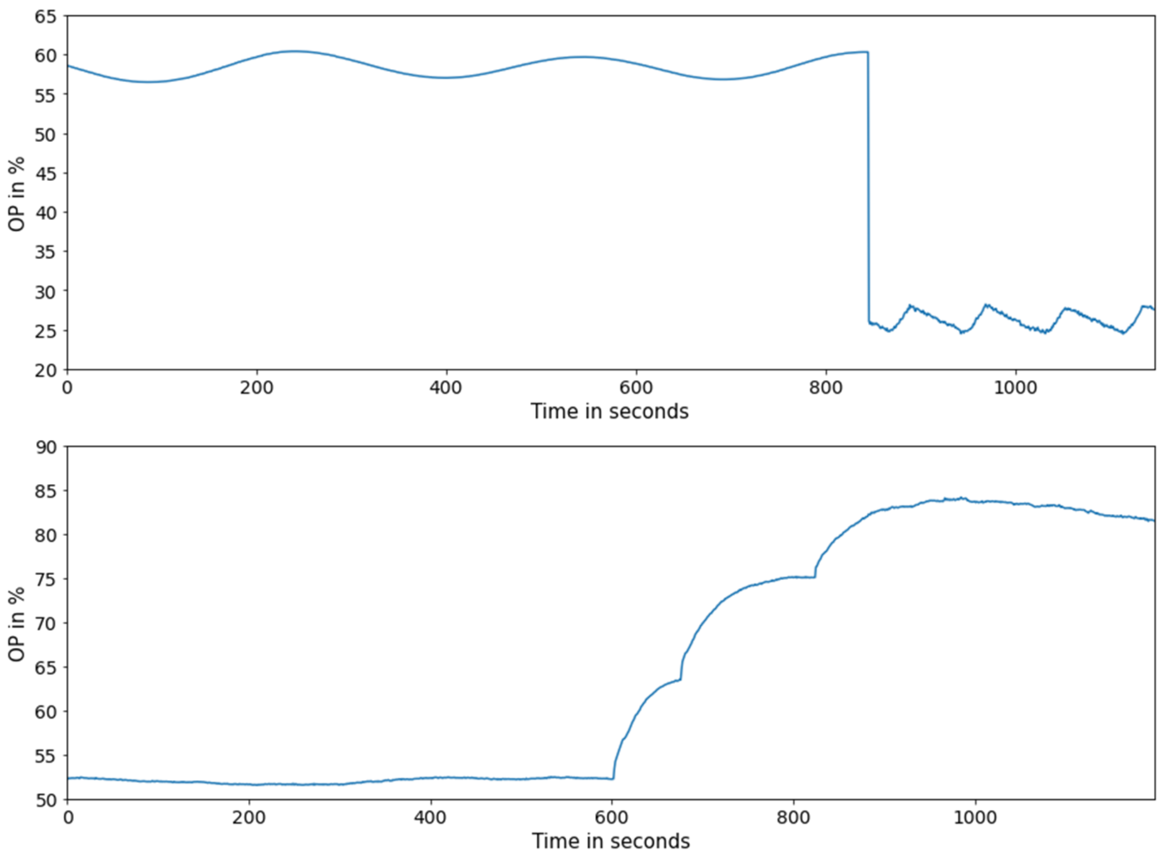


*Figure 11: OP signals with false positive results for method by Bauer et al.*

The conditions (3.4) and (3.5) will be extended by the additional statement that the derivative of the signal at sample $x_i$ must be zero.

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2d} \tag{3.7}$$

$$X_{max} - \Delta_{sat} < x_i \leq X_{max} \;\; and \;\; \frac{x_{i+1} - x_{i-1}}{2d} = 0 \tag{3.8}$$

$$X_{min} \leq x_i < X_{min} + \Delta_{sat} \;\; and \;\; \frac{x_{i+1} - x_{i-1}}{2d} = 0 \tag{3.9}$$

For the approximation of the signal derivative the central difference from numerical mathematics is used as shown in (3.7) where d is the step size between two samples (Heister et al. 2019, 58). The derivative that results from the central difference is illustrated in Figure 12. The modified conditions for a saturated sample are shown in (3.8) and (3.9). This ensures that the signal is actually constrained at the absolute extremum and not just at a local extremum in the investigated time period.

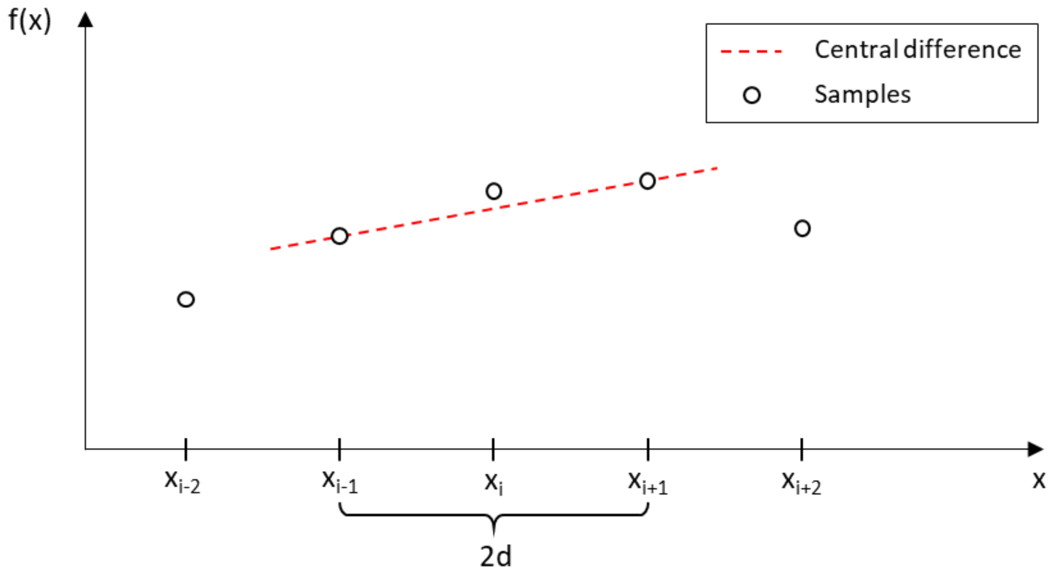


*Figure 12: Illustration of the numerical first order derivative using the central difference*

### 3.1.3 Kolmogorov-Smirnov Two-Sample Test by Matsuo et al.

Matsuo et al. used the D-statistic of the Kolmogorov-Smirnov Two-Sample test to compare a short sample of the OP signal to a test signal where all values are either $OP_{min}$ or $OP_{max}$ (Matsuo et al. 2004). The Kolmogorov-Smirnov Two-Sample test is a goodness of fit test which evaluates whether two samples $S_1$ and $S_2$ come from the same population (Massey 1951). For the verification, the cumulative distribution functions $F_1$ and $F_2$ of the two samples are used (Massey 1951). The cumulative distribution function $F(y_i \leq y)$ is defined as the fraction of values smaller or equal to y.

$$D = max(|F_1(y_i \leq y) - F_2(y_i \leq y)|) \tag{3.10}$$

The D-statistic is the maximum of the absolute differences between the cumulative distribution functions as shown in (3.10) (Matsuo et al. 2004; Massey 1951). The value of the D-statistic can range between 0 and 1. A graphical representation of the D-statistic is given in Figure 13 for two independent samples where the D-statistic is D=0.36.

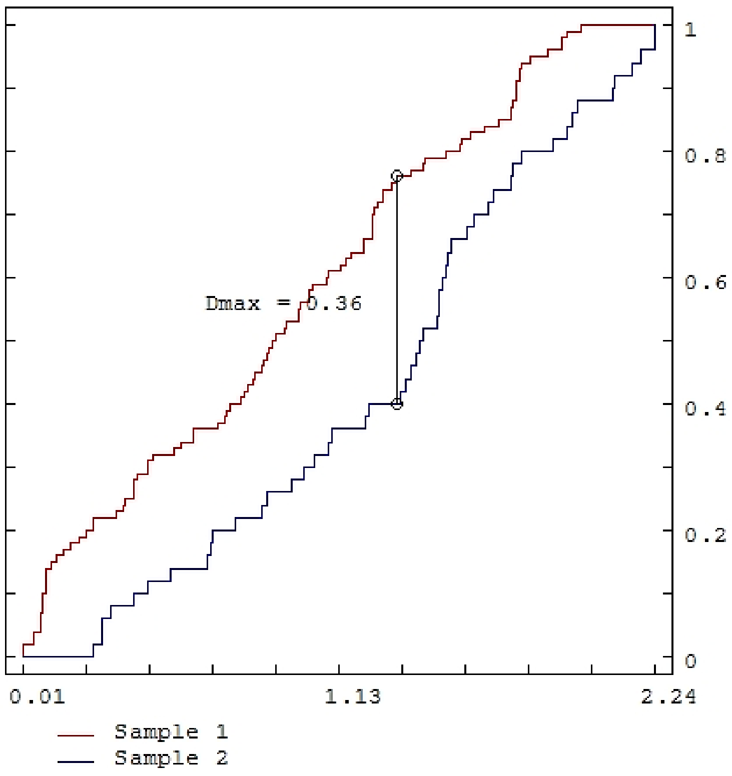


*Figure 13: Cumulative distribution functions for Kolmogorov-Smirnov Two-Sample Test (Statext)*

The D-statistic is compared to $D_{crit}$ to determine whether the two compared samples come from the same population.

$$D_{crit} = c(\alpha) \cdot \sqrt{\frac{m+n}{m \cdot n}} \tag{3.11}$$

$D_{crit}$ can be calculated as shown in equation (3.11) where α is the significance level, m is the sample length of sample $S_1$, and n is the sample length of sample $S_2$ (Zaiontz). The expression c(α) is the inverse Kolmogorov distribution at significance level α. Values for $D_{crit}$ can also be taken from tables in the literature for given sample lengths and significance levels (Massey 1951). The meaning of $D_{crit}$ can be interpreted as follows for α=0.05, m=10 and n=10: For 5 % of the random samples with a length of 10, the maximum absolute difference of the cumulative distribution functions for the considered samples will be at least 0.410 (Massey 1951).

The null hypothesis for the Kolmogorov-Smirnov Two-Sample test is that $S_1$ and $S_2$ were sampled from the same distribution. The null hypothesis is verified if D is smaller than or equal to $D_{crit}$. If D is larger than $D_{crit}$ the null hypothesis is rejected (Duller 2008, 157). In the context of saturation detection, that means that saturation is detected when $D \leq D_{crit}$ (Matsuo et al. 2004).

$$S_1 = \{3.1, 4.6, 3.3, 0.8, 0.2\} \tag{3.12}$$

$$S_2 = \{OP_{min}, OP_{min}, OP_{min}, OP_{min}, OP_{min}\} = \{0, 0, 0, 0, 0\} \tag{3.13}$$

In the following, the Kolmogorov-Smirnov Two-Sample test will be demonstrated on an example. $S_1$ and $S_2$ should be defined as shown in (3.12) and (3.13). $S_1$ should be a sub sample of the OP signal and $S_2$ is a test signal where all values are $OP_{min}$. Under the assumption that the OP signal varies between 0 and 100, the cumulative distribution functions are calculated in Table 1. The D-statistic for the example is D=0.6. For a significance level of α=0.05 and m=5 and n=5, the critical value for D is $D_{crit}$=0.56237 (Massey 1951). Since D is larger than $D_{crit}$ saturation is not detected in this example.

*Table 1: Example Kolmogorov-Smirnov Two-Sample test*

| $S_1 = \{3.1, 4.6, 3.3, 0.8, 0.2\}$ | | $S_2 = \{0, 0, 0, 0, 0, 0\}$ | |
|---|---|---|---|
| $F_1(y_i \leq 1)$ | 0.4 | $F_2(y_i \leq 1)$ | 1 |
| $F_1(y_i \leq 2)$ | 0.4 | $F_2(y_i \leq 2)$ | 1 |
| $F_1(y_i \leq 3)$ | 0.4 | $F_2(y_i \leq 3)$ | 1 |
| $F_1(y_i \leq 4)$ | 0.8 | $F_2(y_i \leq 4)$ | 1 |
| $F_1(y_i \leq 5)$ | 1 | $F_2(y_i \leq 5)$ | 1 |
| $F_1(y_i \leq 6)$ | 1 | $F_2(y_i \leq 6)$ | 1 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $F_1(y_i \leq 100)$ | 1 | $F_2(y_i \leq 100)$ | 1 |

### 3.1.4 Saturation Index for Valve Monitoring by Jämsä-Jounela et al.

Jämsä-Jounela et al. developed a saturation index for valve monitoring where the index is defined as the time $t_{vc}$ a valve is more than 90 % open or less than 10 % open in relation to the time $\tau$ to carry out a setpoint change (Jämsä-Jounela et al. 2003). If the saturation index is close to zero, this indicates that the valve is sized adequately. Values close to one indicate poor actuator dimensioning.

$$SI = \frac{\int_0^\tau t_{vc}\, dt}{\tau} \tag{3.14}$$

$$t_{vc} = \begin{cases} 0, p \in [0.1 \dots 0.9] \\ 1, p < 0.1 \vee p > 0.9 \end{cases} \tag{3.15}$$

The expression $t_{vc}$ is defined in (3.15) where p describes the valve position. Since the method by Jämsä-Jounela et al. is used for the detection of poorly dimensioned valves and not for the general detection of constraints and saturation, this method is not considered in the further course. Furthermore, MV data is required for the application of this method, which is often not available in practice.

## 3.2 Comparison of the Detection Methods

To decide which method is best suited for implementation in the constraint detection add-on, the existing methods should be compared and, if possible, optimised. Since the method developed by Jämsä-Jounela et al. is a valve monitoring method and requires MV data which is often not available, only the method by Bauer et al., the modified method by Bauer et al. and the method by Matsuo et al. are going to be compared in the following Sections.

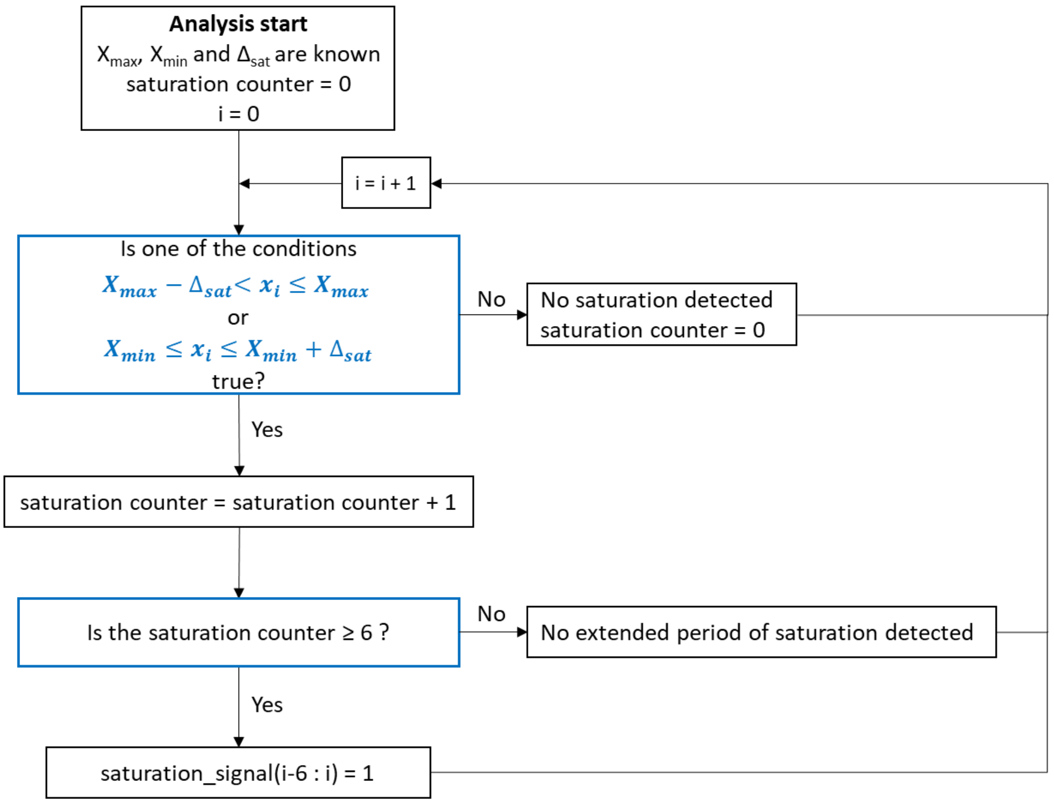
### 3.2.1 Implementation

All three methods were implemented in Python and calculate a saturation signal which is 1 if saturation is present and 0 if saturation is not present. The sample length of the saturation signal is set to be equal to the sample length of the analysed signal and all values are initially set to 0. Individual values in the saturation signal are then set to 1 if saturation is detected.

$$SI = \frac{sum(saturation\ signal)}{L} \cdot 100\ \% \tag{3.16}$$

From the saturation signal, a saturation index SI is calculated as the sum of all values in the saturation signal divided by the sample length L as shown in (3.16). The saturation index can range between 0 % and 100 %, where 0 % is a completely unsaturated signal and a value close to 100 % means that a signal sticks to its extrema most of the time. It must be noticed that a horizontal line with a constant value has an SI of 0 % since a horizontal line has no extrema and cannot be interpreted as saturation by any of the methods. Every method is implemented as a for loop that runs through every sample in the analysed signal and detects saturation based on if-statements. A short explanation of the implementation for every method will be given below including flowcharts.

**Implementation of the Saturation Detection Method by Bauer et al.**



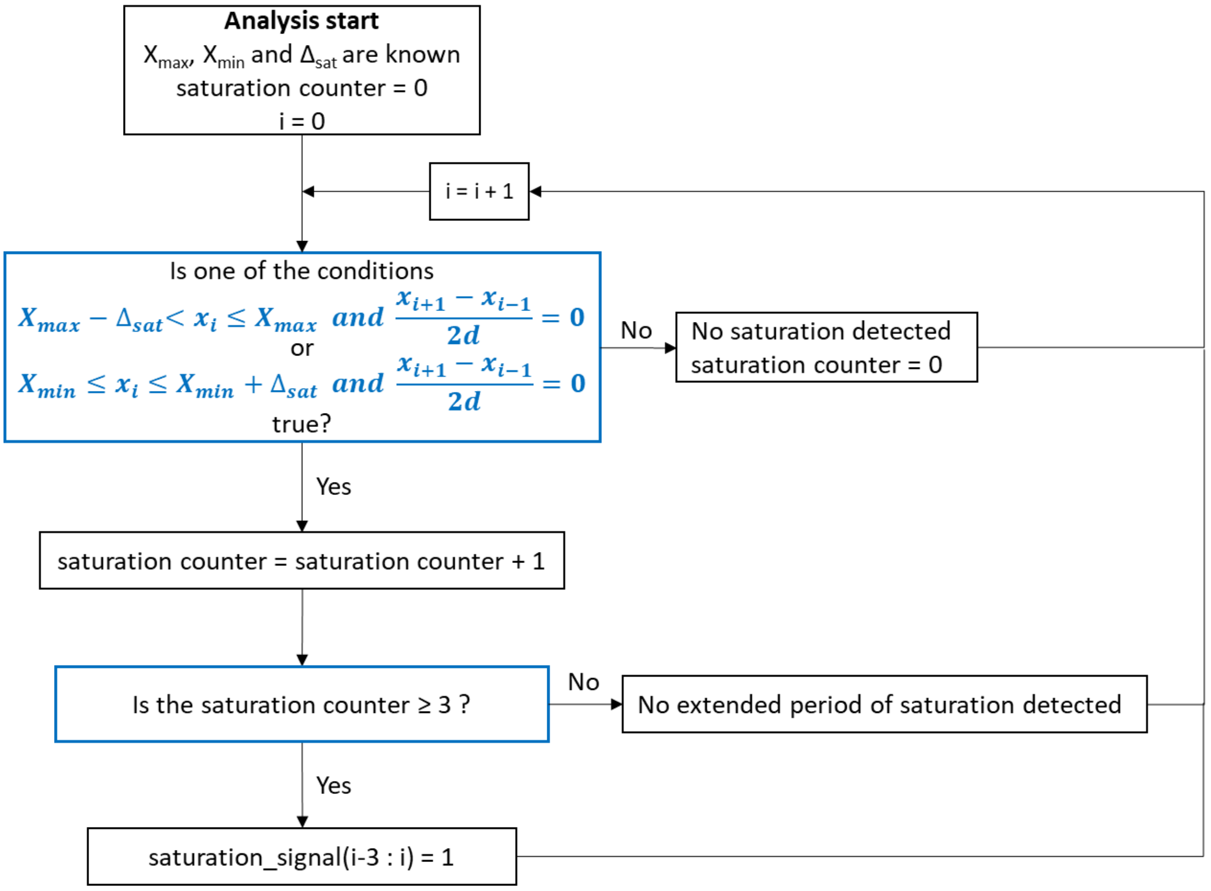
Analysis start
$X_{max}$, $X_{min}$ and $\Delta_{sat}$ are known
saturation counter = 0
i = 0
i = i + 1
Is one of the conditions
$X_{max} - \Delta_{sat} < x_i \leq X_{max}$
or
$X_{min} \leq x_i \leq X_{min} + \Delta_{sat}$
true?
No
No saturation detected
saturation counter = 0
Yes
saturation counter = saturation counter + 1
Is the saturation counter ≥ 6 ?
No
No extended period of saturation detected
Yes
saturation_signal(i-6 : i) = 1

*Figure 14: Flowchart of the Python code for the detection method by Bauer et al.*

For the detection method by Bauer et al., which is illustrated in Figure 14, the Python code assesses the minimum, maximum and the saturation band of the analysed signal. A saturation counter is defined with the initial value of 0. The saturation counter is used to detect when an extended period of saturation is present. The first if-statement in the loop tests whether the sample $x_i$ lies in one of the two saturation bands. If the statement is true, the saturation counter is raised by 1. If the statement is false, the saturation counter is reset to 0. The next if-statement assesses whether the saturation counter has reached the extended length of saturation l=6. The value l=6 was taken from Bauer et al. (Bauer et al. 2019). If an extended period of saturation is present, the saturation signal is set to 1 for the current sample and the five previous samples.

**Implementation of the Modified Saturation Detection Method by Bauer et al.**

The modified method works analogously to the original method by Bauer et al. and is visualised in Figure 15. The first if-statement additionally tests whether the derivative is 0 at the current sample. Because the additional condition ensures that the analysed signal is flat and constrained at an extremum, the extended length of saturation was reduced to l=3.



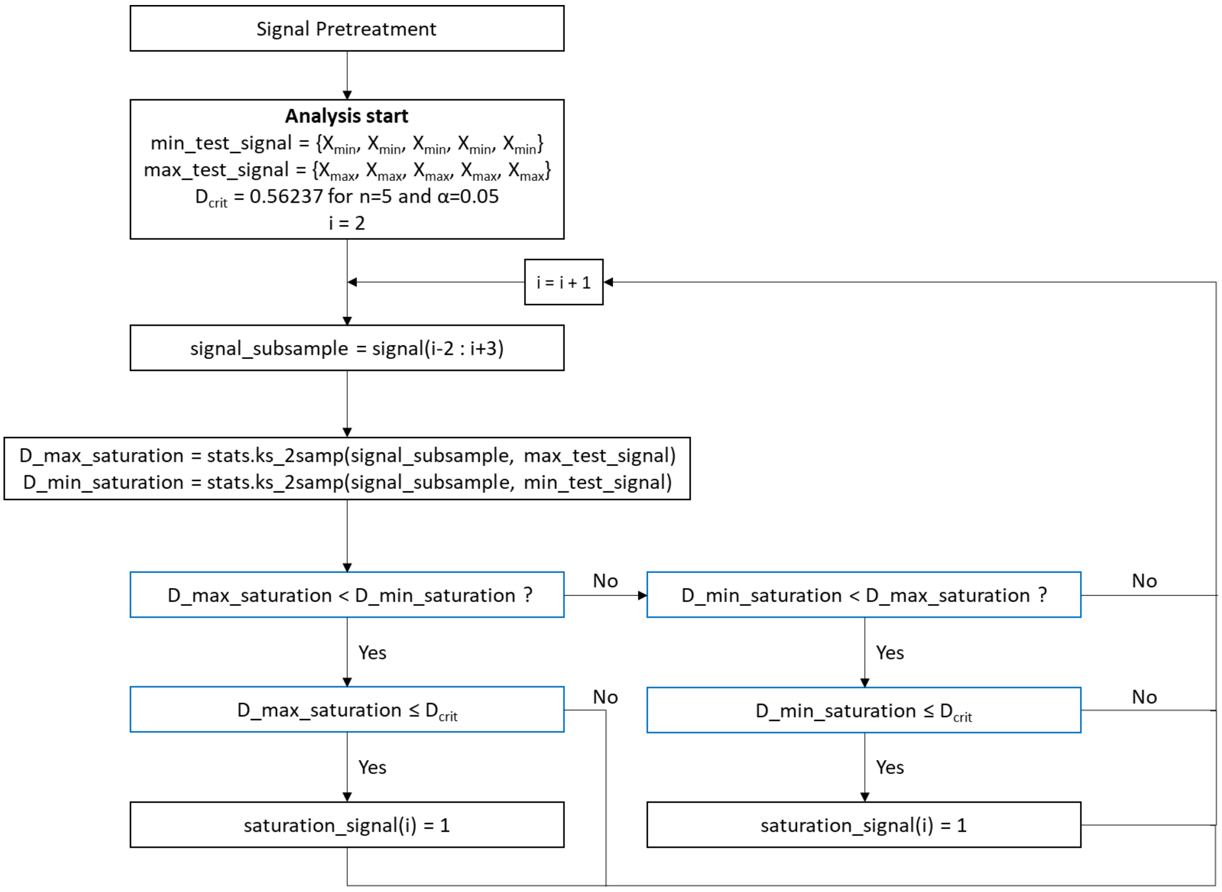
Analysis start
$X_{max}$, $X_{min}$ and $\Delta_{sat}$ are known
saturation counter = 0
i = 0
i = i + 1
Is one of the conditions
$X_{max} - \Delta_{sat} < x_i \leq X_{max}$ and $\frac{x_{i+1} - x_{i-1}}{2d} = 0$
or
$X_{min} \leq x_i \leq X_{min} + \Delta_{sat}$ and $\frac{x_{i+1} - x_{i-1}}{2d} = 0$
true?
No
No saturation detected
saturation counter = 0
Yes
saturation counter = saturation counter + 1
Is the saturation counter ≥ 3 ?
No
No extended period of saturation detected
Yes
saturation_signal(i-3 : i) = 1

*Figure 15: Flowchart of the Python code for the modified method by Bauer et al.*

**Implementation of the Saturation Detection Method by Matsuo et al.**

For the method by Matsuo et al. in Figure 16, two test signals are initiated. The min_test_signal contains 5 values which are equal to $X_{min}$ and the max_test_signal contains 5 values which are equal to $X_{max}$. The critical D-statistic is set to $D_{crit}$=0.56237. This is the corresponding value for n=5 and α=0.05 where the sample length n=5 was taken from Matsuo et al. and the significance level α=0.05 was chosen because it is commonly used in the Kolmogorov-Smirnov Two-Sample test. In the first step of the for loop, a signal subsample of length n=5 is created. Afterwards, two Kolmogorov-Smirnov Two-

Sample tests are performed. The first one compares the max_test_signal with the signal subsample and the second one compares the min_test_signal with the signal subsample. For the Kolmogorov-Smirnov Two-Sample test, the ks_2samp function from the Scipy Python library is used. The function gives the D-statistic as a result. In the next step, it is evaluated whether the D-statistic for saturation at $X_{min}$ or saturation at $X_{max}$ is smaller. This is done because a smaller D-statistic corresponds to a higher probability that saturation is present. It also prevents that saturation can be detected at the minimum and at the maximum at the same time. The smaller D-statistic is compared to $D_{crit}$. If the D-statistic is smaller or equal to $D_{crit}$, saturation is detected and the saturation signal for the current sample is set to 1. If the D-statistic is larger than $D_{crit}$, saturation is not present.



Signal Pretreatment
Analysis start
min_test_signal = {Xmin, Xmin, Xmin, Xmin, Xmin}
max_test_signal = {Xmax, Xmax, Xmax, Xmax, Xmax}
Dcrit = 0.56237 for n=5 and α=0.05
i = 2
i = i + 1
signal_subsample = signal(i-2 : i+3)
D_max_saturation = stats.ks_2samp(signal_subsample, max_test_signal)
D_min_saturation = stats.ks_2samp(signal_subsample, min_test_signal)
D_max_saturation < D_min_saturation ?
No
D_min_saturation < D_max_saturation ?
No
Yes
Yes
D_max_saturation ≤ Dcrit
No
D_min_saturation ≤ Dcrit
No
Yes
Yes
saturation_signal(i) = 1
saturation_signal(i) = 1

*Figure 16: Flowchart of the Python code for the method by Matsuo et al.*

In contrast to the methods based on Bauer et al., the method by Matsuo et al. requires two pre-treatment steps. The first pre-treatment step is filling in missing values. This is linked to the ks_2samp function which misinterprets missing values and outputs a wrong D-statistic. The wrong D-statistic can lead to false detection results. Therefore, the analysed signal is checked for missing data and the fillna function is used to fill missing values with the next valid value in the signal.

$$signal\_subsample = \{99.99, 99.99, 99.99, 99.99, 99.99\} \tag{3.17}$$

$$max\_test\_signal = \{100, 100, 100, 100, 100\} \tag{3.18}$$

The second pre-treatment step is linked to the Kolmogorov-Smirnov Two-Sample test itself. As an example, the signal_subsample and the max_test_signal in (3.17) and (3.18) are considered. For these samples no saturation would be detected although the signal is saturated. This occurs because the D-statistic is 1 and therefore always higher than $D_{crit}$. The D-statistic is 1 because it is calculated as the difference of the cumulative distribution functions of the samples. For this reason, the data is rounded in the second pre-treatment step to avoid false negatives.

### 3.2.2 Detection Method Requirements

To perform the comparison of the detection methods, requirements must be defined which a method has to fulfil to be considered as "good". The most important requirement is the method robustness. The robustness will be assessed with the number of false positive and false negative results. A low number of false results indicates a high robustness. Furthermore, it must be taken into account that the method will be implemented into an add-on which in turn will be used by an engineer. The method must be understandable for the end user to make the detection results easy to interpret. Ideally all hard-coded parameters should be kept constant so that the user does not have to make a choice on parameters which are not fully understood by the user. The hard-coded parameters include the extended length of saturation l for Bauer et al. and the sample length n, the significance level α and the corresponding $D_{crit}$ for Matsuo et al. Another important factor is the computation time because in the add-on the detection method will be applied not only to a single signal but more likely to a hundred signals or more. The computation time should be low so that the add-on works fast and efficient. A short and compact list of the requirements is given below.

- **Robustness**: Low number of false positives and false negatives
- **Understandability**: Detection method is easy to understand for the user
- **Constant hard-coded parameters**: Detection method works sufficiently well always using the same parameter settings
- **Low computation time**

### 3.2.3 Comparison of the Method Robustness

The comparison of the methods will be structured as follows. In this Section, the robustness of the methods will be evaluated on two data sets. Afterwards the influence of the hard-coded parameters on the detection results will be investigated in Section 3.2.4. Lastly, a summary will be given in Section 3.3 which contains the results of the robustness and hard-coded parameter evaluation, numbers on the computation time and an evaluation on the method understandability. Based on the summary, the detection method, which is most suitable for implementation in the add-on, will be chosen.

For an initial overview of the method performances, four OP signals from the SACAC data set are used. The SACAC data set can be found here (South African Council for Automation and Control). The data set contains industrial data for faults which cause poor control loop performance such as valve stiction, poor tuning, saturation, quantisation, plant-wide disturbances, sensor faults and other unknown faults.

*Table 2: Description of signals from the SACAC data set*

| Name | Description | Sampling rate | Length |
|---|---|---|---|
| Saturation-L-minerals | saturation in a level control loop in a minerals and mining process | 10 | 3241 |
| Saturation-T-oilgas | saturation in a temperature control loop in an oil and gas process | 20 | 1441 |
| Tuning-L-paper | tuning fault in a level control loop in a pulp and paper process | 1 | 1147 |
| Other-F-paper | unknown fault in a flow control loop in a pulp and paper process | 1 | 1198 |

The SACAC data set contains two examples for saturation which will be referred to as saturation-L-minerals and saturation-T-oilgas. Bauer et al. reported two false positives when they first tested their saturation detection method on the SACAC data set. The false positives occurred for tuning-L-paper and other-F-paper. A description of the data is given in Table 2. The saturation index for every signal will be calculated and the time trends with the saturation signals will be compared in the following.

Since four OP signals are not sufficient to evaluate the method robustness, the methods will additionally be tested on another industrial data set which will be referred to as the SA data set. The SA data set contains 1042 normalised control signals (including controller outputs, process variable, setpoints and auto manual modes) with a sampling rate of 10 seconds in the period from 20.01.2017 to 20.02.2017. From the SA data set, a subset of 50 signals was selected which includes 30 OP signals, 10 PV signals and 10 SP signals. This data subset will be used to test the detection methods for false results.

Table 3 shows the calculated saturation indices for the four signals from the SACAC data set. All three detection methods are able to detect saturation in saturation-L-minerals and saturation-T-oilgas. It is noticeable that the modified method by Bauer et al. gives the lowest indices for both signals and the method by Matsuo et al. the highest. Regarding tuning-L-paper an other-F-paper, Bauer's method gives false positive results as Bauer et al. already reported in their paper (Bauer et al. 2019). Matsuo's method gives a false positive result for tuning-L-paper, but not for other-F-paper. The modified version of Bauer's method does not give any false positives and therefore interprets all four signals correctly. Based on Table 3, it could be argued that either the modified method by Bauer et al. or the method by Matsuo et al. show the best performance. In any case, there are significant differences in the method performances. To visualise the underlying reasons for the different detection results, the time trends will be considered in Figure 17 to Figure 22.

*Table 3: Saturation index for method by Bauer et al., modified method by Bauer et al. and method by Matsuo et al.*

| | **Saturation Index**<br>Extended Period of Saturation Test<br>l=6<br>Bauer et al. | **Saturation Index**<br>Extended Period of Saturation Test with Derivative=0<br>l=3<br>Bauer et al. | **Saturation Index**<br>Kolmogorov-Smirnov Two-Sample Test<br>n=5, α=0.05<br>Matsuo et al. |
|---|---|---|---|
| Saturation-L-minerals | 32.1 % | 24.7 % | 38.4 % |
| Saturation-T-oilgas | 20.6 % | 14.9 % | 31.7 % |
| Tuning-L-paper | 5.2 % | 0.0 % | 1.8 % |
| Other-F-paper | 10.8 % | 0.0 % | 0.0 % |

In Figure 17 and Figure 18, the detection results for saturation-L-minerals are shown. The time periods where the saturation signal is 1 are visualised as coloured time capsules. In Figure 17, only the first 2500 samples of saturation-L-minerals are shown to improve the visibility of the time capsules. The most noticeable difference is that the saturation time capsules for Matsuo's method are long and connected even when there is noise in the saturated periods. The saturation capsules for the methods based on Bauer et al. become short and interrupted for noisy, saturated periods which can be seen in more detail in Figure 18 where the samples 1850 to 2300 are shown and the y-axis shows OP values between 98.5 % and 100 %. The effect of short and interrupted time capsules is more dominant in the modified version of Bauer's method.

The reason for the different lengths of the time capsules is linked to the detection methods. While the methods based on Bauer et al. rely on the fact that all values in saturated periods lie close to the extremum, Matsuo's method evaluates a statistical similarity to a completely saturated test signal. If the similarity is high enough, saturation is detected. Therefore, Matsuo's method gives long saturation capsules even for noisy periods in saturation-L-minerals resulting in a higher saturation index. The methods based on Bauer et al. consequently have lower saturation indices. The additional condition in the modified method regarding the derivative, leads to short saturation capsules and some saturated periods are missed completely as can be seen in Figure 18 in more detail.

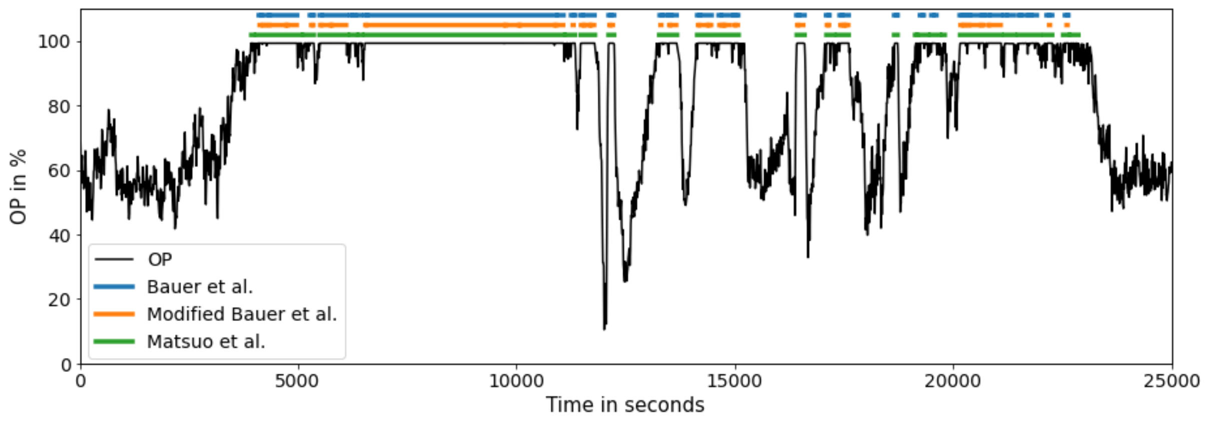


*Figure 17: Saturation detection results for saturation-L-minerals*

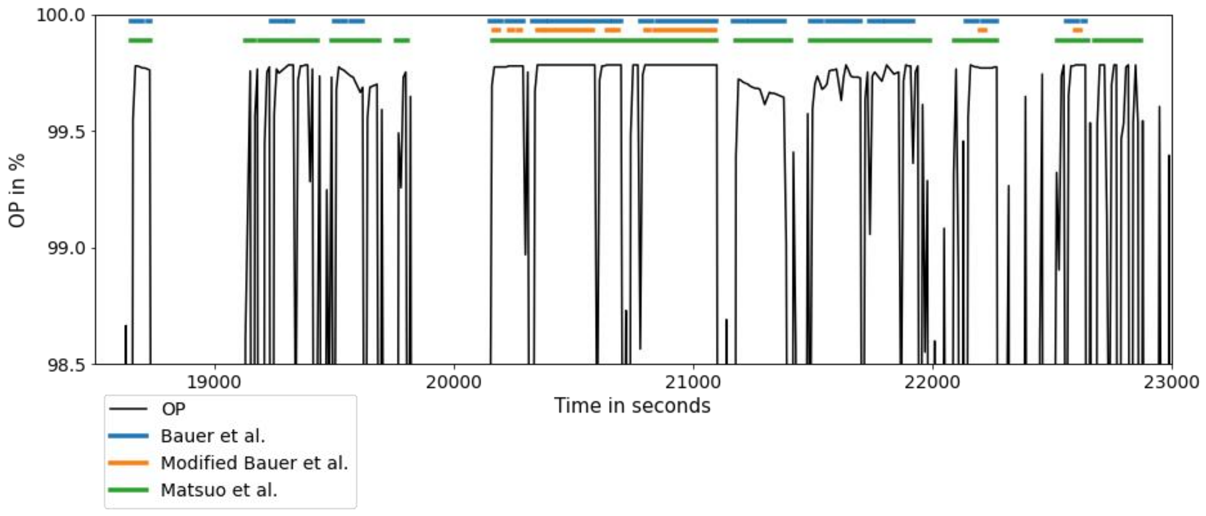


*Figure 18: Detailed view of saturation detection results for saturation-L-minerals (t = 18500-23000 s)*

Figure 19 and Figure 20 show the results for saturation-T-oilgas. The signal oscillates and the saturated periods are very short including only a few samples. Therefore, the methods by Bauer et al. do not detect all saturated periods because saturation is only detected when an extended period of saturation of length l=6 or respectively l=3 is present. Since the modified version of Bauer et al. requires the "derivative=0" condition to be met, this method has the fewest and shortest time capsules. A more detailed view of saturated periods that are missed by the methods based on Bauer et al. is given in Figure 20. The method by Matsuo et al. is able to detect these short, saturated periods so that the oscillation can also be seen in the saturation capsules. The fact that Matsuo's method does not require a minimum length of the saturated period is an advantage in this case.

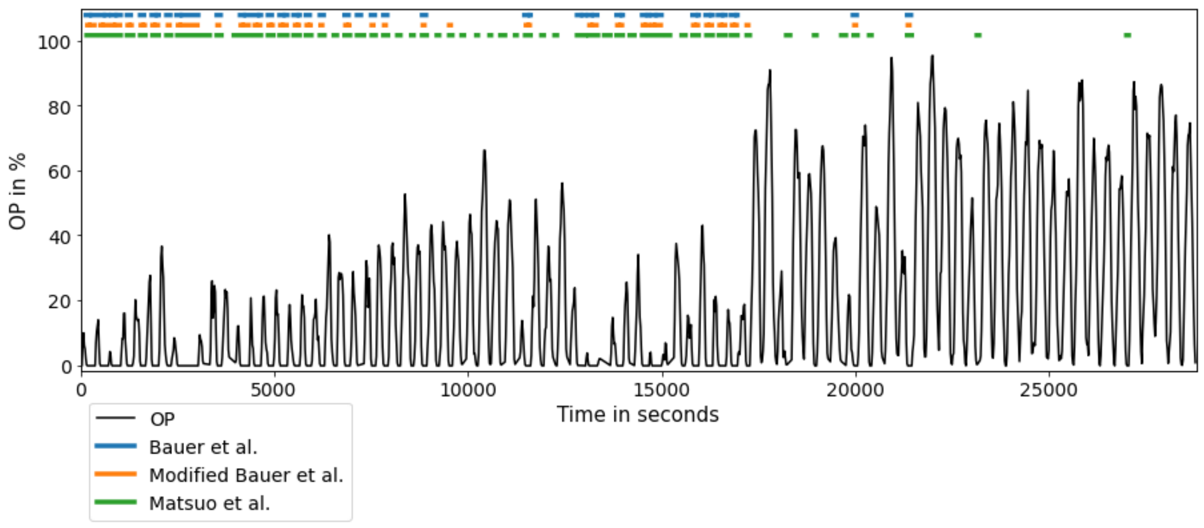


*Figure 19: Saturation detection results for saturation-T-oilgas*

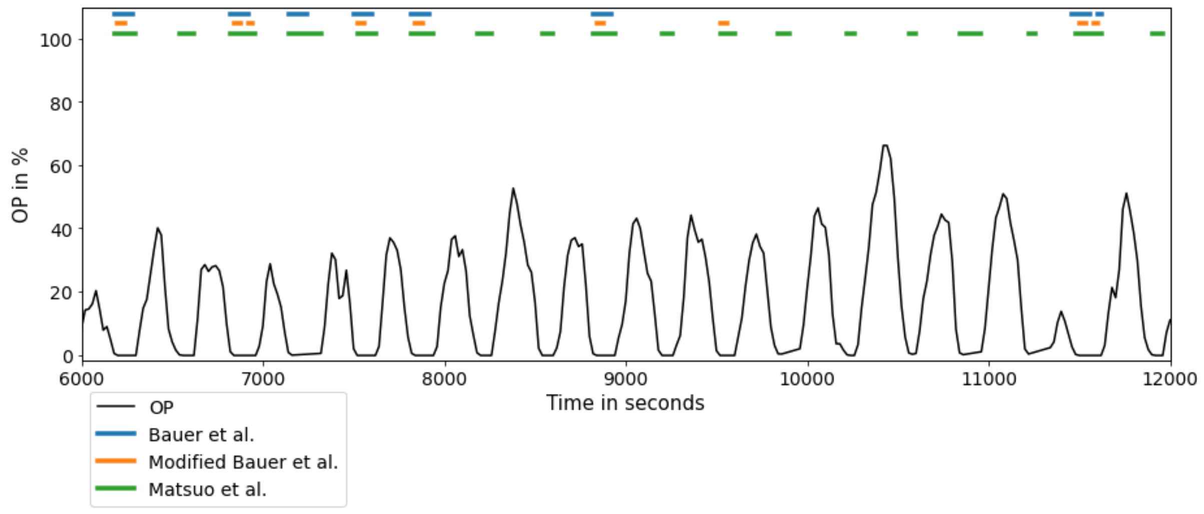


*Figure 20: Detailed view of saturation detection results for saturation-T-oilgas (t= 6000-12000 s)*

The time trend for tuning-L-paper is shown in Figure 21. Bauer's original method and Matsuo's method detect saturation at the maximum of the signal. While this is the only saturation capsule for the method by Matsuo et al., the method by Bauer et al. detects another saturated period shortly before a setpoint change occurred. The modified version of Bauer's method does not detect saturation because the derivative is not zero when the signal is at its maximum. For the original method by Bauer et al., the false saturation capsules can be eliminated by increasing the extended length of saturation significantly which would negatively influence the method's performance for short, saturated periods as it is the case for saturation-T-oilgas in Figure 19. Although the saturation capsule for Matsuo et al. is short, it is difficult to make a statement about whether it could be eliminated by adjusting the hard-coded parameters because of the method's complexity. Therefore, the influence of the hard-coded parameters on the detection methods is analysed in more detail in Section 3.2.4.

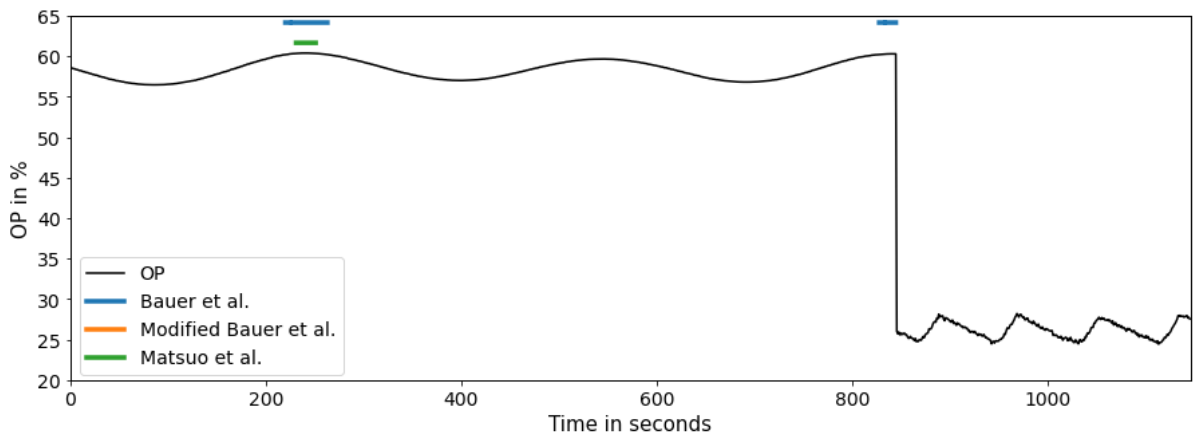


*Figure 21: Saturation detection results for tuning-L-paper*

In Figure 22, the results for other-F-paper are visualised. The OP signal remains at its minimum at around 52 % for 600 seconds and resembles saturation since the signal looks like a horizontal line. As a result, the method by Bauer et al. detects saturation because there are consecutive samples close to the signal minimum. The modified version of Bauer’s method does not detect saturation since the derivative is only close to zero but not exactly zero. For Matsuo’s method, the Kolmogorov-Smirnov Two-Sample test does not detect a similarity large enough between the OP signal and the minimum test signal so that no saturation is detected.

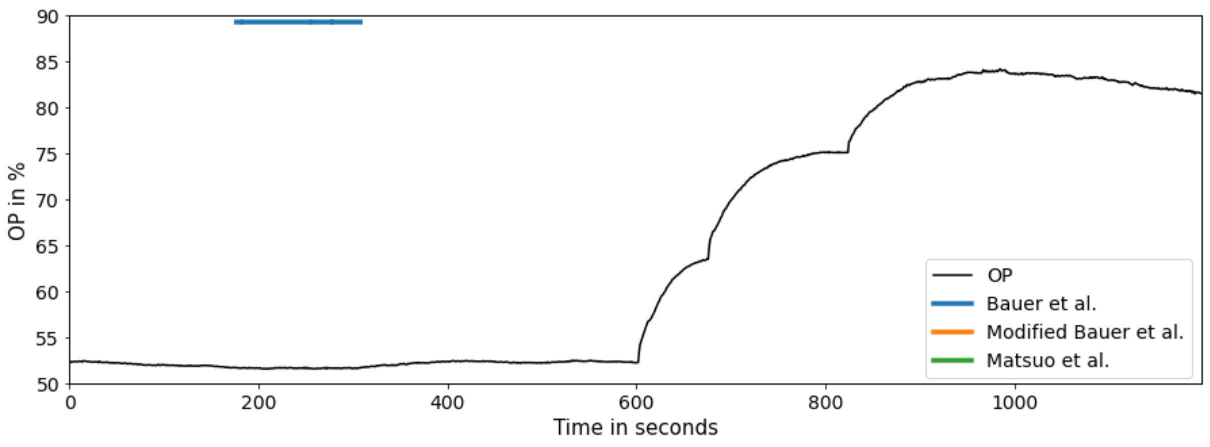


*Figure 22: Saturation detection results for other-F-paper*

Important conclusions can be drawn from the time trends. The original method by Bauer et al. can detect saturation but has some drawbacks including short saturation capsules in noisy, saturated periods, the inability to detect short, saturated periods during oscillations and misinterpreting flat time trends as saturation. The modified version of Bauer’s method has similar drawbacks for noisy, saturated periods and short, saturated periods but performs better when a signal has a flat time trend but is not saturated. Matsuo’s method shows a good performance on the SACAC data set. The uncertainty whether and how false positives could be removed is the only drawback which could be named for Matsuo’s method at this point.

For a more reliable assessment of the method robustness, the methods were tested on the subset of the SA data set with 50 signals. This data set includes OP, PV, and SP data because the detection method will not only be used for saturation detection but also for constraint detection in manipulated variables, process variables and setpoints. Table 4 gives an overview of the total number of analysed signals and the number of saturated signals per signal type. In total, 30 OP signals, 10 PV signals and

10 SP signals were analysed. All setpoints are set by an MPC controller. For each signal type, half of the signals are unsaturated/unconstrained to get an even overview of false positives and false negatives. All analysed signals contain the data of one day of operation with a sampling rate of 10 seconds and a total of 8641 samples. The original method by Bauer et al. gave 9 false results. All of them are false positives. The modified version gave 7 false results with 6 false positives and 1 false negative. The method by Matsuo et al. gave 20 false results with 19 false positives and 1 false negative.

*Table 4: False positive and false negative results for tested saturation detection methods*

| | Nr. of Signals | Nr. of Sat. Signals | | Extended Period of Saturation Test l=6 Bauer et al. | Extended Period of Saturation Test with Derivative=0 l=3 Bauer et al. | Kolmogorov-Smirnov Two-Sample Test n=5, α=0.05 Matsuo et al. |
|---|---|---|---|---|---|---|
| OP | 30 | 15 | False positives | 3 | 1 | 12 |
| | | | False negatives | 0 | 0 | 0 |
| PV | 10 | 5 | False positives | 1 | 0 | 2 |
| | | | False negatives | 0 | 1 | 1 |
| SP | 10 | 5 | False positives | 5 | 5 | 5 |
| | | | False negatives | 0 | 0 | 0 |
| Total | 50 | 25 | False results | 9 | 7 | 20 |

All three methods are more prone to false positives than to false negatives. In contrast to the performance on the SACAC data set, Matsuo's method performs poorly on this data set. In 40 % of the cases the method gives a false result which is unacceptable for a detection method. The false results are partly connected to the pre-treatment step of rounding the data. Not rounding the data enough or at all can lead to false negatives and rounding the data too much can lead to false positives. The methods based on Bauer et al. perform similarly when compared with each other. The added condition in the modified version leads to a slightly better performance. In general, the methods based on Bauer et al. give more reliable results when the analysed time interval is longer. This is the case because a longer time interval gives a higher probability that the absolute minimum and maximum of the signal are contained in the analysed period.

The high number of false positives for the SP signals in Table 4 is noticeable. The time trend in Figure 23 shows why all detection methods detect a constraint in every SP signal. When considering the time trend for the whole day, the setpoint is not constrained at any time but when a shorter time interval around the SP minimum is considered, the SP shows step-like features. The horizontal line at the minimum is misinterpreted as a constraint by the detection methods. Since all SP signals in the data set have similar step-like features, they all have a positive result for constraints.

The robustness analysis revealed important properties of the individual methods. The method by Bauer et al. gives short time capsules for noisy, saturated periods and does not detect all saturated periods when an oscillation is present. The method is prone to give false positive results when the time trend is flat but not saturated. This effect gets weaker as the analysed time interval becomes longer. The modified method based on Bauer et al. has the same drawbacks for noisy, saturated periods and short, saturated periods. Nevertheless, the method showed the least number of false results which shows that the "derivative = 0" condition is a valuable addition to the original method. The method by Matsuo et al. has the main advantage that it can handle noise in saturated periods and therefore gives high saturation indices. On the other hand, the method has the highest number of false results which makes it unsuitable for implementation in the add-on unless it can be improved significantly.

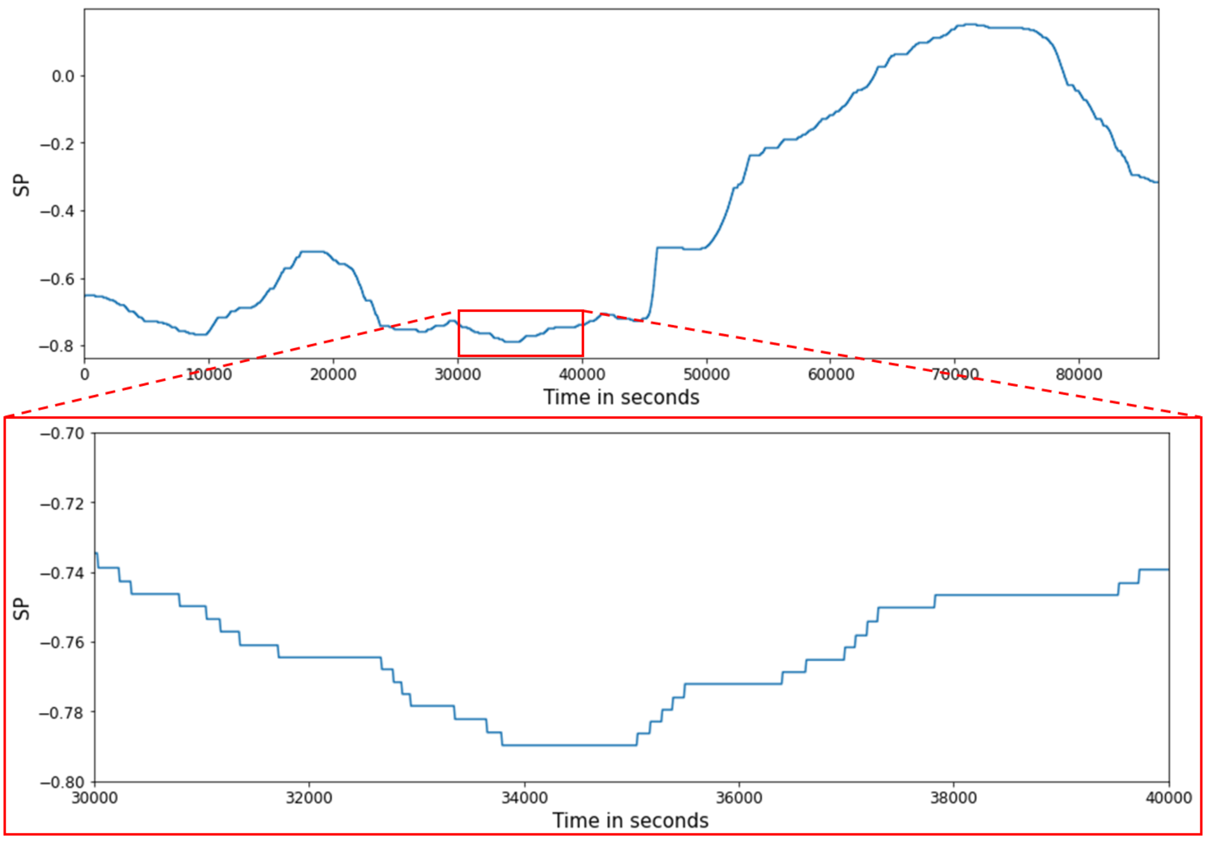


*Figure 23: Time trend of an unconstrained setpoint signal with MPC (normalised data)*

### 3.2.4 Influence of Hard-Coded Parameters on the Detection Methods

The results of the detection methods are significantly influenced by their hard-coded parameters. For the methods based on Bauer et al., the hard-coded parameter is the extended length of saturation l. For Matsuo et al., the detection results are influenced by the critical D-statistic $D_{crit}$ where $D_{crit}$ is calculated from the significance level α and the sample length n. An overview of the hard-coded parameters is given in Table 5. In the following, the saturation index is examined as a function of the hard-coded parameters to evaluate whether the method robustness can be improved by changing the parameters and whether the hard-coded parameters should be adjustable for the end user.

*Table 5: Overview of hard-coded parameters*

| **Detection method** | **Hard-coded parameters** |
|---|---|
| Extended Period of Saturation Test Bauer et al. | ▪ Extended length of saturation **l**<br>The extended length of saturation was set to l=6 in Section 3.2.3. |
| Extended Period of Saturation Test with Derivative=0 Bauer et al. | ▪ Extended length of saturation **l**<br>The extended length of saturation was set to l=3 in Section 3.2.3. |
| Kolmogorov-Smirnov Two-Sample Test Matsuo et al. | ▪ Significance level **α**<br>▪ Sample length **n** of the test signal and signal subsample<br>The significance level and sample length influence the critical D-statistic $\mathbf{D_{crit}}$. The significance level was set to α=0.05 and the sample length was set to n=5 which results in $D_{crit}$=0.56237 in Section 3.2.3. |

In Figure 24, the saturation index for Bauer et al. is shown as a function of the extended length of saturation. For this analysis, the SACAC data set was used. For all four signals the saturation index decreases when the extended length of saturation is increased. However, this trend varies in intensity for the four signals considered. For saturation-L-minerals, the saturation index decreases steadily but is still at 22.3 % for l=25. For saturation-T-oilgas in contrast, the saturation index decreases strongly so that for l=15 the saturation index is only 1.94 %. This is less than for the false positive results for tuning-L-paper and other-F-paper. The different behaviours for saturation-L-minerals and saturation-T-oilgas result from the fact that saturation-L-minerals has long, saturated periods and saturation-T-oilgas has very short, saturated periods due to the oscillation. For the false positive results in tuning-L-paper and other-F-paper, it can be observed that the saturation index decreases very little as a function of l. This means that false positives cannot be eliminated by increasing the extended length of saturation. On the contrary, the detection capability of the method deteriorates while the false positives are retained. In summary, the method by Bauer et al. works best for small values for l but false positives cannot be eliminated by changing this hard-coded parameter. For the end user, no benefit is provided if the extended length of saturation would be an adjustable parameter.

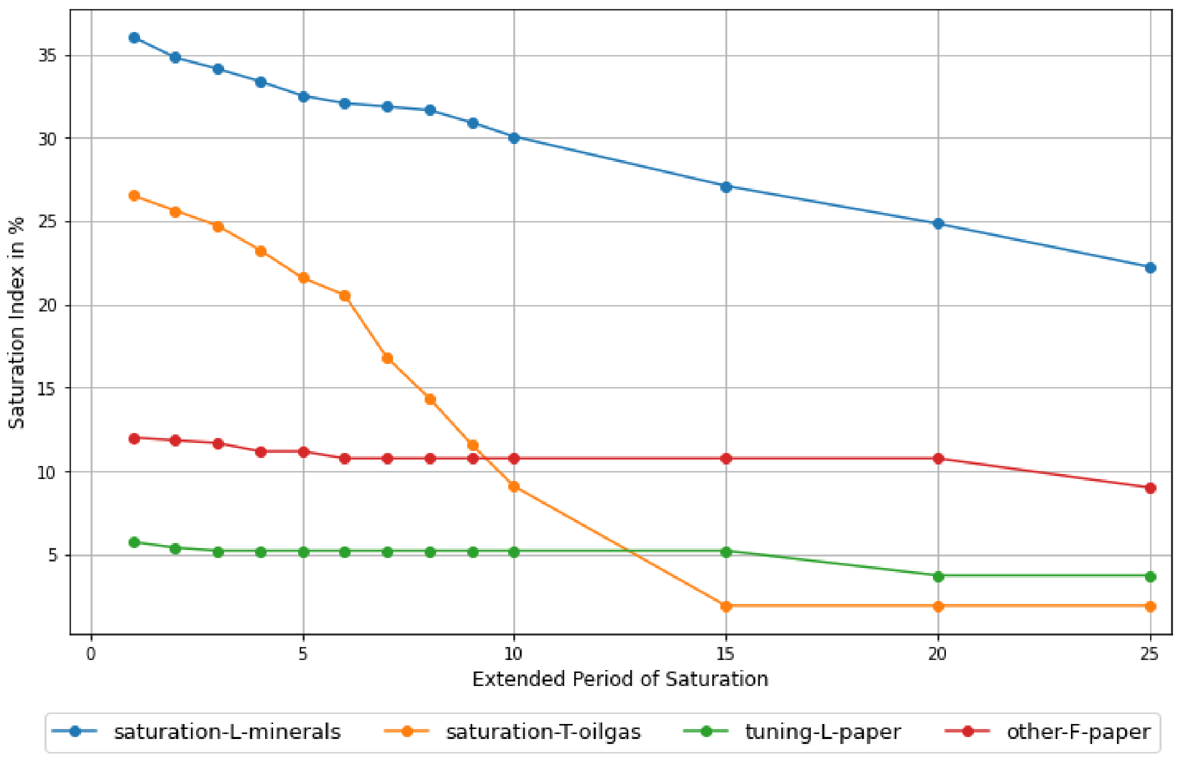


*Figure 24: Saturation index in dependence of the extended length of saturation for Bauer et al.*

In Figure 25, the saturation index for the modified version of Bauer et al. is shown as a function of the extended length of saturation. The saturation indices for saturation-L-minerals and saturation-T-oilgas are generally lower than in Bauer's original method but they decrease in a similar way. The saturation indices for saturation-L-minerals decrease slowly while the values for saturation-T-oilgas decrease rapidly and then stay constant at 1.74 % for l≥15.

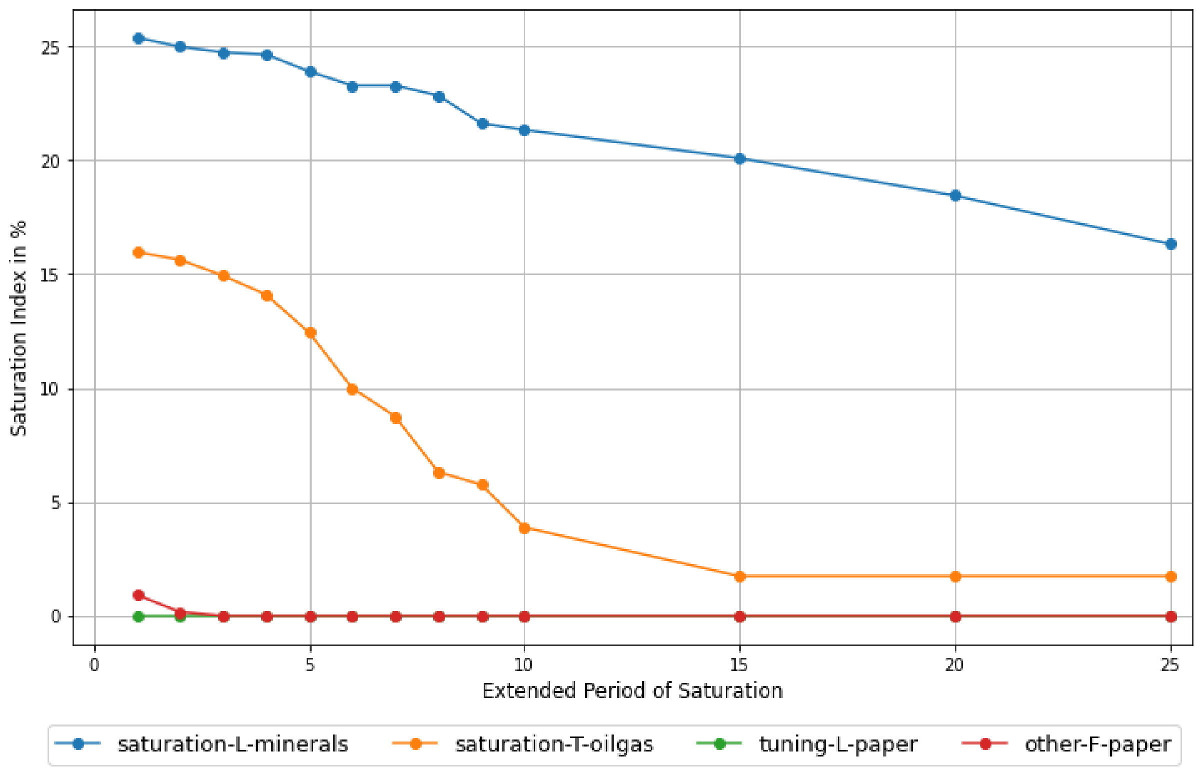


*Figure 25: Saturation index in dependence of the extended length of saturation for the modified version of Bauer et al.*

Regarding the unsaturated signals tuning-L-paper and other-F-paper, it can be concluded that the initial setting l=3 for the extended length of saturation was an appropriate choice since other-F-paper has a false positive result for l=1 and l=2. This indicates that the number of false results can be reduced by increasing the extended length of saturation. Therefore, the method was tested again with l=6 on the subset of the SA data set with 50 signals. The results are shown in Table 6. The number of false results stays the same since one false positive result was removed but a new false negative result was added. Consequently, the setting l=3 seems to be a good compromise between keeping a high saturation index for saturated signals and avoiding most false positives. The parameter should not be an adjustable parameter for the end user.

*Table 6: False positive and false negative results for modified version of Bauer et al. with l=3 and l=6*

| | Nr. of Signals | Nr. of Sat. Signals | | Extended Period of Saturation Test with Derivative=0 l=3 Bauer et al. | Extended Period of Saturation Test with Derivative=0 l=6 Bauer et al. |
|---|---|---|---|---|---|
| OP | 30 | 15 | False positives | 1 | 0 |
| | | | False negatives | 0 | 0 |
| PV | 10 | 5 | False positives | 0 | 0 |
| | | | False negatives | 1 | 2 |
| SP | 10 | 5 | False positives | 5 | 5 |
| | | | False negatives | 0 | 0 |
| Total | 50 | 25 | False results | 7 | 7 |

For the method by Matsuo et al. the saturation index was analysed as a function of the sample length n and as a function of the significance level α. In Figure 26, the analysis results for changes in the sample length of the signal subsample and the test signal for the Kolmogorov-Smirnov Two-Sample test are shown for α=0.05. The signal subsample and the test signal have the same length n in all test scenarios. The results for saturation-L-minerals and saturation-T-oilgas are displayed on the left y-axis and the results for tuning-L-paper and other-F-paper are displayed on the right y-axis.

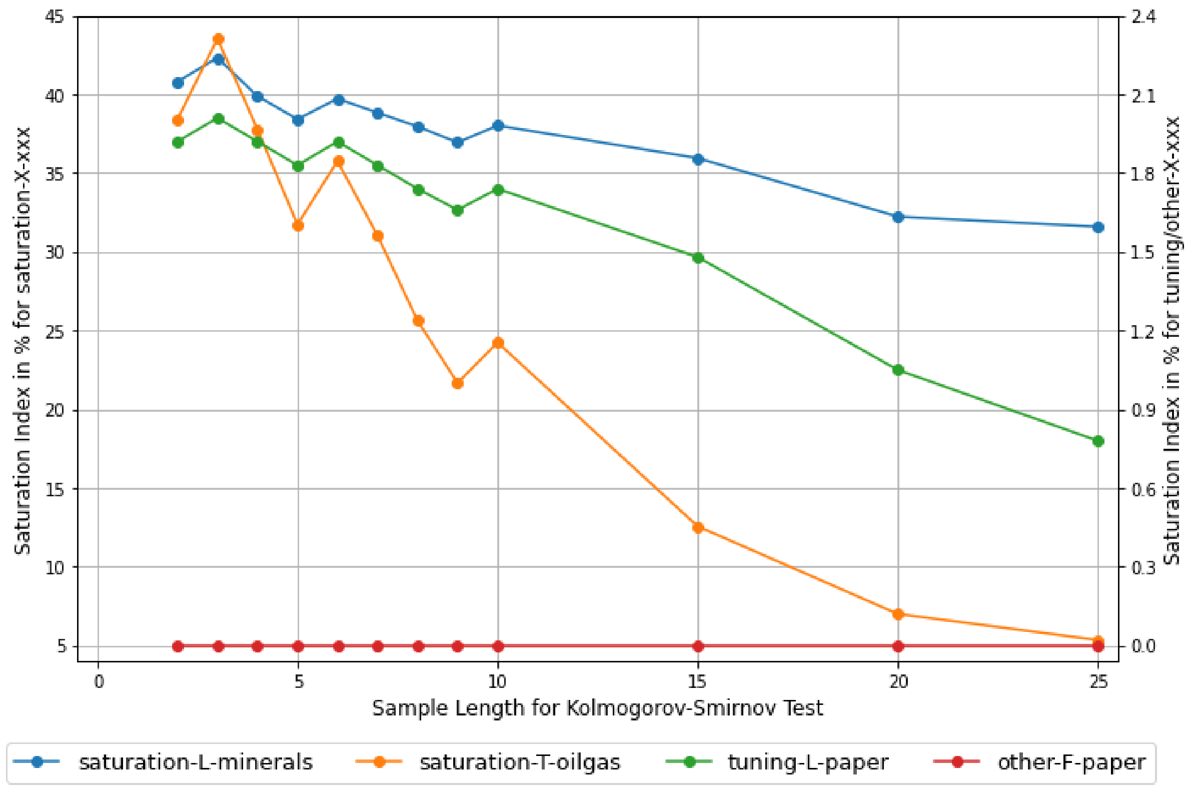


*Figure 26: Saturation index in dependence of the sample length for Matsuo et al. (α=0.05)*

The saturation index decreases with increasing sample length of the signal subsample and the test signal. This has two underlying reasons. The first reason is that the statistical similarity between the signal subsample and the test signal becomes smaller as the signals become longer. The second reason is that $D_{crit}$ decreases with increasing sample length. Consequently, the D-statistic will be larger than $D_{crit}$ more often and in this case no saturation is detected, and the saturation index becomes smaller. Figure 26 shows that for n≥9 the false positive result for tuning-L-paper becomes larger than the saturation index for saturation-T-oilgas. Furthermore, it is noticeable that all signals for which saturation is detected have their highest saturation index for a sample length of n=3. This indicates that the sample length for Matsuo's method should be kept low and that the sample length is not an appropriate parameter to improve the method robustness.

In Figure 27, the saturation index is shown as a function of the significance level for a sample length of n=5. Although $D_{crit}$ changes significantly in dependence of α the changes in the saturation index are only minor. $D_{crit}$ decreases when α is increased. This is because of the statistical meaning of $D_{crit}$ in the Kolmogorov-Smirnov Two-Sample test. For a certain significance level α and sample length n, the probability is α that the maximum absolute difference of the cumulative distribution functions of both samples of size n is at least $D_{crit}$. This means that at a constant sample length n, α decreases when $D_{crit}$ becomes larger. For this reason, the highest saturation indices occur for significance levels between 0.001 and 0.02.

From Figure 26 and Figure 27 it can be concluded that a sample length of n=3 and values for α between 0.001 and 0.02 give the highest saturation index. Nevertheless, the false positive result for tuning-L-paper remains for all tested scenarios so that the method robustness cannot be improved by adjusting the hard-coded parameters. Because of the complexity of Matsuo’s method, it is also not advantageous to let the end user set the hard-coded parameters. To avoid false positive results, a complementary method would have to be developed for this purpose only, which would further increase the complexity of the method.

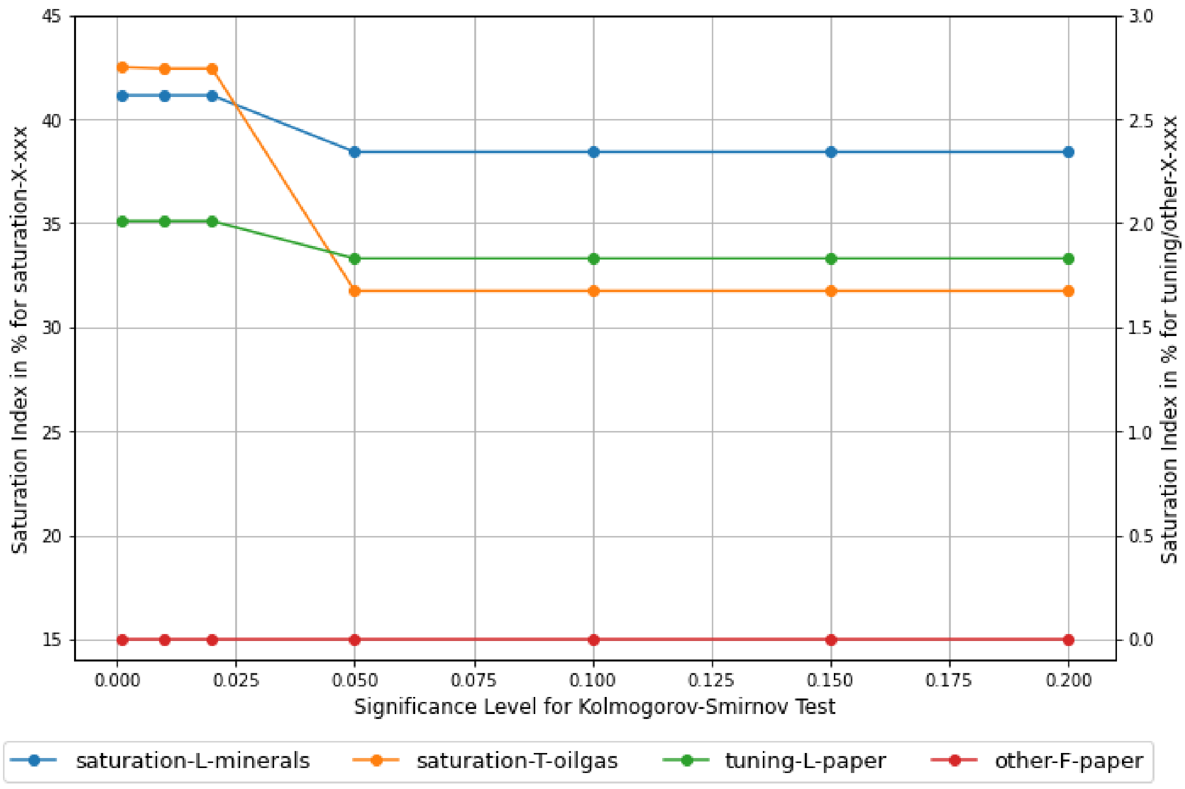


*Figure 27: Saturation index in dependence of the significance level for Matsuo et al. (n=5)*

## 3.3 Conclusion and Selection of the Detection Method

In this Section, the findings from Section 3.2.3 and Section 3.2.4 will be summarised and a detection method for implementation in the add-on will be selected based on the requirements defined in Section 0. Table 7 shows the performance of each method with reference to the requirements. The modified method based on Bauer et al. has the highest robustness with a percentage of correct results of 87.0 %. For the hard-coded parameters, no optimal setting with 100 % correct results could be found for any of the methods. Also, it was not found to be of advantage to make the hard-coded parameters adjustable for the end user. Therefore, no difference can be made between the methods about the hard-coded parameters. The methods based on Bauer et al. are easy to understand for the user which makes the detection results easier to interpret. This is of clear advantage for both methods. The Kolmogorov-Smirnov Two-Sample test is more difficult to understand and the underlying relationships of $D_{crit}$, sample length n and significance level α are complex. For the assessment of the computation time, the detection methods were tested on another subset of the SA data set with 444 signals with a duration of one week. The detection methods based on Bauer et al. take less than a minute to process the whole data set whereas Matsuo's method takes around 30 minutes. If the method by Matsuo et al. would be implemented in the add-on, most users would be likely to cancel the data analysis because of the slow process which makes it unsuitable for the implementation in the add-on. Furthermore, additional method properties have been identified and their influence on the detection results has been evaluated. In summary, the detection methods based on Bauer et al. fulfil more requirements than the method by Matsuo et al. Since the modified version of Bauer's method has a higher robustness, this method will be implemented in the add-on.

*Table 7: Comparison of the method performances (green=good performance, red=bad performance, black=neutral)*

| | Extended Period of Saturation Test<br>l=6<br>Bauer et al. | Extended Period of Saturation Test with Derivative=0<br>l=3<br>Bauer et al. | Kolmogorov-Smirnov Two-Sample Test<br>n=5, α=0.05<br>Matsuo et al. |
|---|---|---|---|
| Robustness (percentage of correct results) | 43/54 = 79.6 % | 47/54 = 87.0 % | 33/54 = 61.1 % |
| Hard-coded parameters | Can be set to a constant value | Can be set to a constant value | Can be set to a constant value |
| Understandability | Easy to understand | Easy to understand | Difficult to understand |
| Computation time | < 1 min for 444 signals | < 1 min for 444 signals | ca. 30 min for 444 signals |
| Other properties of the method | Time capsules become short and interrupted for noisy, saturated periods<br><br>Very short, saturated periods are missed<br><br>Flat time trends in unsaturated signals are misinterpreted as saturation | Time capsules become short and interrupted for noisy, saturated periods<br><br>Very short, saturated periods are missed | Requires pretreatment of the data<br><br>Can handle noise in saturated periods |

# 4. Data Analytics in Seeq

In the following Sections, the data analytics software Seeq will be introduced in terms of its architecture and its use for control loop performance monitoring. As it is the objective of this work to develop and implement a constraint detection add-on, it is important to understand the data analytics workflow in Seeq and which functions are useful in CPM because it is convenient to use similar functions to detect and visualise saturation and constraints.

## 4.1 Seeq Software Architecture

Seeq is a software to perform data analytics on time-series industrial process data. The software offers solutions for all phases involved in the data analytics process such as searching for data, visualising data, performing analytics with point-and-click tools, and creating reports. Seeq has three main applications which are Seeq Workbench, Seeq Organizer and Seeq Data Lab which are shown in Figure 28.

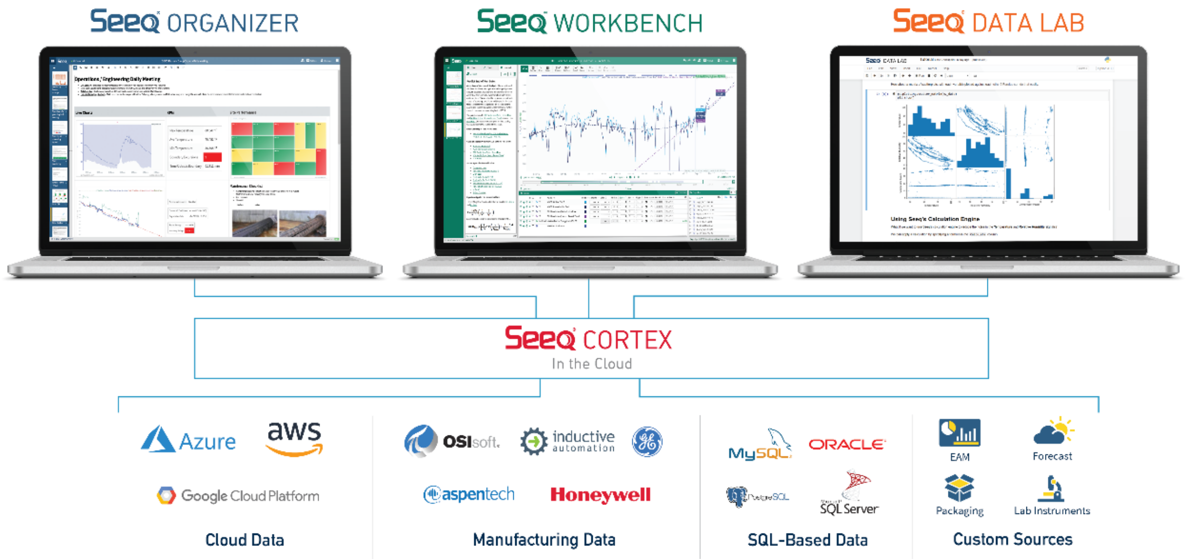


*Figure 28: Overview of the Seeq software architecture (Seeq Corporation)*

Workbench is a browser-based application where the user can search for data, analyse data and create visualisations with the analytics results (Seeq Corporation). Tools for diagnostic and predictive analytics as well as advanced analytics including pattern recognition and machine learning are provided. Workbench is extensible and provides users with the option to install add-ons with custom functions. Seeq customers can implement their own add-ons, but Seeq also offers a series of different add-ons e.g. for causality analysis, multivariate pattern search and valve stiction detection (Seeq Corporation). To perform data analytics, users must first create a workbook with Seeq Workbench. Within one workbook multiple worksheets can be created in which data analytics can be performed and visualisations can be created. Workbooks can be shared with colleagues so that every workbook has one workbook owner and potentially multiple users who access the workbook in shared mode. Organizer is also a browser-based application where the user can create reports based on the analytics which were previously created in Workbench (Seeq Corporation). Data Lab is based on jupyter notebook and provides the user with the Seeq Python library SPy (Seeq Corporation). Using SPy, the user can pull data from Workbench, perform analysis in a Python environment and push the results back to Workbench. Workbench, Organizer and Data Lab connect to the application backend Seeq Cortex which in turn connects to the customer's data source (Seeq Corporation). Seeq is able to connect to cloud data, manufacturing data and SQL-based data. Cloud data includes data from cloud platforms such as AWS, Microsoft Azure and Google Cloud Platform. Manufacturing data includes data historian technologies such as OSIsoft, AspenTech, Honeywell and others. SQL-based data includes data historians which are SQL server applications. In the ICS, Seeq is located in the enterprise management layer and connects to the data historian in the supervisory layer.

## 4.2 Control Loop Performance Monitoring in Seeq

Although Seeq is more targeted towards process engineers, it also provides tools and functions which are suitable for the use by control engineers in CPM (Cox 2021; Seeq Corporation).

In Seeq Workbench, the user can create event-based or periodic conditions which are visualised as coloured time capsules in the time trend as shown in Figure 29 for a temperature measurement. In Figure 29, a condition was created which looks for time periods where the temperature is above 95 °F. Time periods, in which the condition is true, are marked with red time capsules.

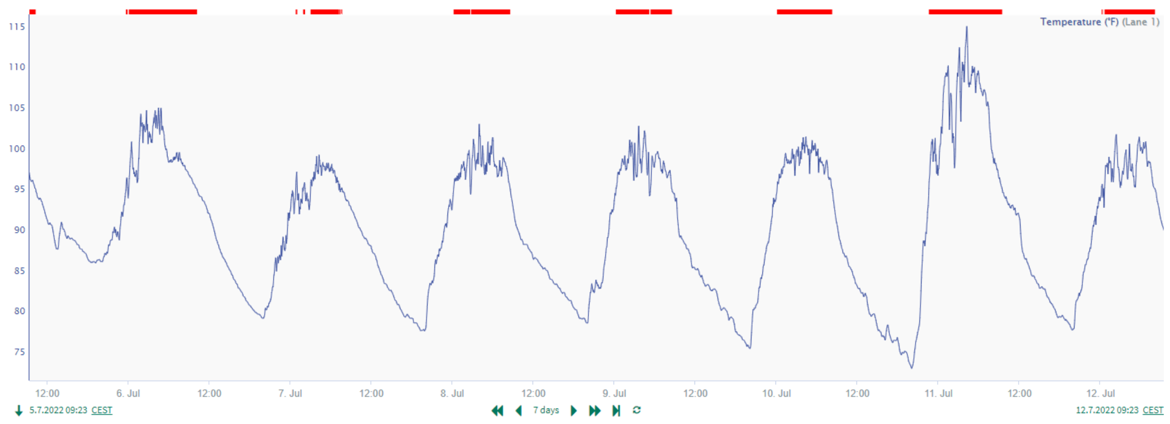


Figure 29: Time capsules for a temperature measurement in Seeq

In CPM, an example for an event-based condition could be if the controller is in manual or automatic mode. Periodic conditions are based on regular time periods such as days, weeks and months or operator shifts. Conditions and time capsules are Seeq's main feature, and many tools build up on conditions. From a condition, the user can calculate the "number of setpoint changes per controller and day" or the "percent-time spent in manual mode per week". Furthermore, it is also possible to define benchmarks and thresholds and to calculate basic statistics including mean, standard deviation, and variance. Since most control engineers use mode statistics and basic statistics for CPM, Seeq's most simple tools can be sufficient to execute basic CPM (Bauer et al. 2016).

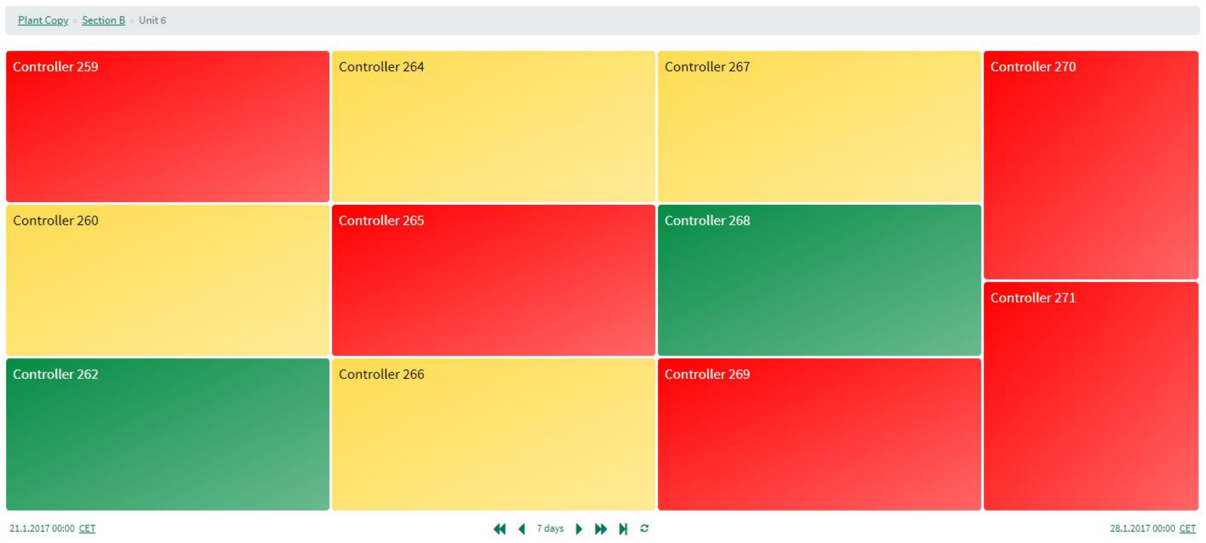
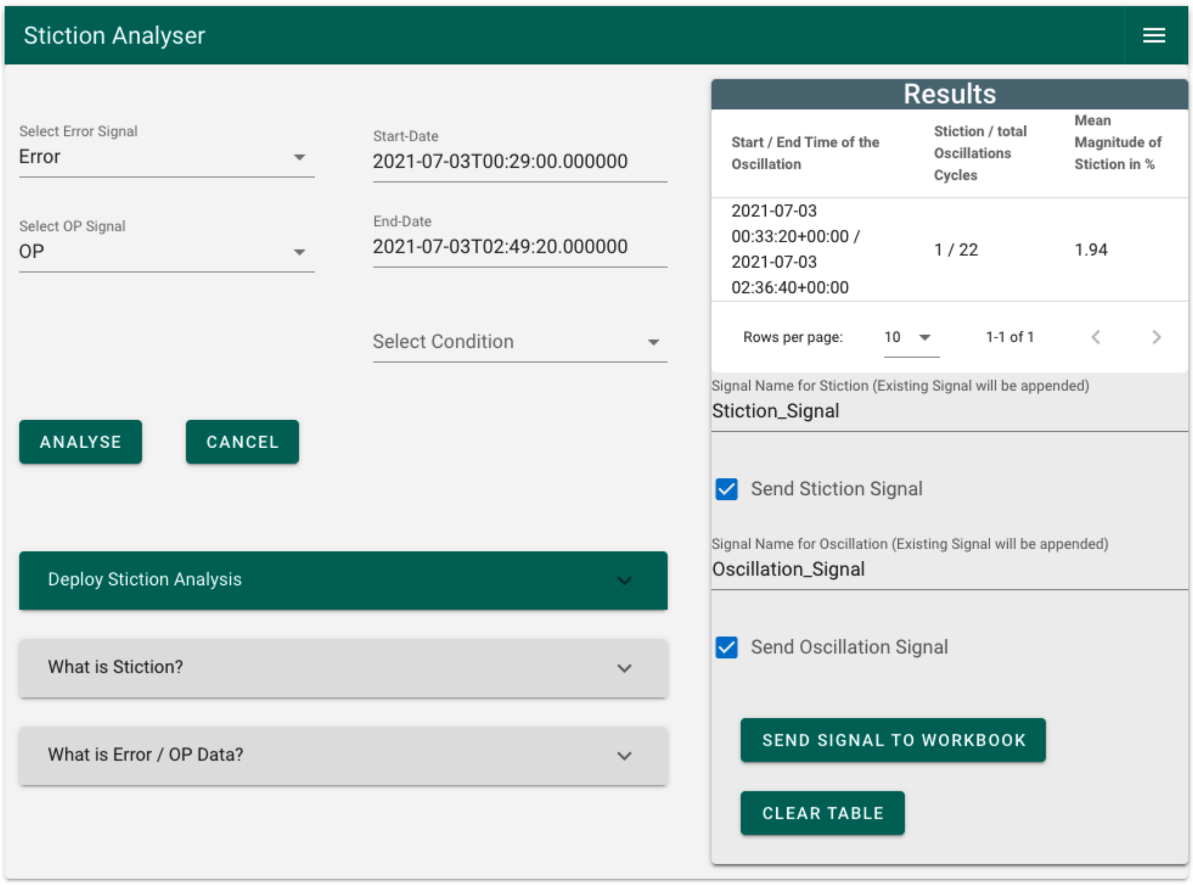


Figure 30: Treemap used for an asset tree with controllers

Another useful Seeq feature for CPM is the possibility to create an asset tree. An asset tree is a hierarchical structure that assigns signals to specific assets. For example, the controller output, process variable, setpoint, manipulated variable and auto manual mode can be assigned to a controller and the controller in turn can be assigned to a system component. If an asset tree for controllers has been defined, the user can use treemap view, shown in Figure 30, to get a plant-wide overview of controller performance based on user-defined conditions. If a controller panel in the treemap is clicked, the user is directed to the time trend of the corresponding controller signals.

Seeq also provides tools for advanced analytics which can be used for CPM (Cox 2021). These tools include data cleansing methods, signal alignment and dynamic modelling. In addition, multiple add-on tools can be installed for advanced CPM analytics including the causality add-on, the correlation add-on and the stiction analyser add-on. Add-ons developed by Seeq Corporation provide the user with a user interface (UI) like in Figure 31 for the stiction analyser add-on. The causality add-on creates causal maps with cause-and-effect relationships. The correlation add-on visualises pairwise correlations in a heatmap and takes time-lagged correlations into account. The causality add-on and the correlation add-on are useful to assess plant-wide disturbances or multi-loop interactions. The stiction analyser add-on imports an error signal and the corresponding controller output and performs oscillation detection and valve stiction detection.



Select Error Signal: Error

Select OP Signal: OP

Start-Date: 2021-07-03T00:29:00.000000

End-Date: 2021-07-03T02:49:20.000000

| Start / End Time of the Oscillation | Stiction / total Oscillations Cycles | Mean Magnitude of Stiction in % |
|---|---|---|
| 2021-07-03 00:33:20+00:00 / 2021-07-03 02:36:40+00:00 | 1 / 22 | 1.94 |

Rows per page: 10 1-1 of 1

Signal Name for Stiction (Existing Signal will be appended): Stiction_Signal

[✓] Send Stiction Signal

Signal Name for Oscillation (Existing Signal will be appended): Oscillation_Signal

[✓] Send Oscillation Signal

*Figure 31: User interface of the stiction analyser add-on (Essinger)*

# 5. Constraint Detection Add-on

The main objective of this work is the development and implementation of a constraint detection add-on which detects constraints and saturation in control loop data. In the following sections, the constraint detection add-on is presented. In Section 5.1, all specifications regarding the add-on function and other requirements including user requirements, detection method requirements and requirements from Seeq Corporation are explained. Section 5.2 explains the asset tree structure that is required for the add-on execution. In Section 5.3.1, the user interface, add-on function and workflow are introduced by means of an example workflow. Furthermore, Section 0 describes how the user can get support in the user interface and in the add-on documentation. Three use cases for the constraint detection add-on are presented in Section 5.4. Finally, the add-on is evaluated in Section 5.5 using the specifications and requirements from Section 5.1.

## 5.1 Add-on Specifications and Requirements

The Hamburg University of Applied Sciences and Seeq Corporation have jointly decided that an add-on for the detection of saturation and constraints in control loop data should be developed in this thesis using Python as the programming language. It was specified that the add-on should process an asset tree of control loop data. Furthermore, it should be possible to define a start and end time for the analysis period in the UI. If necessary, the UI should have further input options to make the tool more user-friendly.

As a result, the add-on should create an output asset tree that is a copy of the input asset tree in the analysis period. In addition to the input control signals, the output asset tree should also contain conditions with time capsules for saturated/constrained periods and a measure for the extent of saturation/constraints. Based on the output asset tree and the measure for the extent of saturation/constraints, a treemap should be generated automatically. In the treemap, controllers with a high extent of saturation/constraints should be coloured red, controllers with a medium extent of saturation/constraints yellow and controllers with a low or no extent of saturation/constraints green. The thresholds which define a “high extent of saturation/constraints”, or “medium extent of saturation/constraints” should be specified by the user. Table 8 summarises the specifications for the add-on inputs and outputs.

*Table 8: Input and output specifications for the constraint detection add-on*

| | |
|---|---|
| **Input** | ▪ Input asset tree<br>▪ Start and end time of analysis period<br>▪ Thresholds for visualisation in treemap<br>▪ Other input options for higher user-friendliness |
| **Output** | ▪ Output asset tree with:<br>    ▪ Input control signals<br>    ▪ Constraint/saturation signal<br>    ▪ Conditions with time capsules for saturated/constrained periods<br>    ▪ Measure for the extent of saturation/constraints<br>▪ Treemap coloured according to the extent of saturation/constraints<br>▪ Other outputs for higher user-friendliness |

In addition to the add-on input and output specifications, further requirements were formulated that the add-on should fulfil. These requirements can be divided into user requirements, detection method requirements and requirements from Seeq Corporation.

For the user, it is of high importance that the add-on gives correct, easy-to-interpret and meaningful results. In addition, the add-on should provide an intuitive UI and easy workflow with a short processing time. In order to make the add-on use as easy as possible, the add-on documentation should contain a comprehensible user guide and installation instructions with all user-relevant information. The documentation should also include multiple use cases to demonstrate how the add-on can be applied. Furthermore, it should be easy for the user to request support in case of problems or for special use cases.

The detection method requirements were specified in Section 3.2.2 and are related to the user requirement for correct results and a short processing time.

In addition to a fully functional add-on that fulfils all user requirements and detection method requirements, Seeq Corporation has further requirements. The Python code should be maintainable and follow coding standards so that other Seeq employees or students from the Hamburg University of Applied Sciences can work with the code in the future. The Python code should also be documented in the official add-on documentation and should be published on GitHub. To make the add-on an open-source Python package, which can be installed by every Seeq user, it should be uploaded on PyPI.

The requirements are summarised in Table 9. Table 8 and Table 9 will be used in Section 5.5 to assess the extent to which the add-on meets the specifications and requirements.

*Table 9: Add-on requirements*

| | |
|---|---|
| **User requirements** | ▪ Correct, easy-to-interpret, meaningful results<br>▪ Easy workflow and intuitive UI<br>▪ Short processing time<br>▪ Documentation with comprehensible user guide, installation instructions and use cases<br>▪ Simple way to get support |
| **Detection method requirements (from Section 0)** | ▪ Robustness<br>▪ Understandability<br>▪ Constant hard-coded parameters<br>▪ Low computation time |
| **Requirements from Seeq Corporation** | ▪ Maintainable code that follows coding standards<br>▪ Code documentation<br>▪ Publication of the add-on on GitHub<br>▪ Upload of the add-on as a Python package on PyPI |

## 5.2 Structure of the Input Asset Tree

The main add-on input is the input asset tree which contains the control loop data for the analysis. This asset tree requires a specific structure so that the add-on can process the data correctly. In order to distinguish the individual control signals, a certain naming is required. The control signals may only be called "Controller Output", "Process Variable", "Setpoint", "Manipulated Variable" and "Mode". Signals with other names cannot be processed by the add-on. Each signal must be assigned to a controller so that the visualisation of the analysis result in treemap functions correctly. For the controllers and other assets, no specific naming is necessary. To help the user generate the required asset structure correctly, template code is made available on GitHub (Tiedemann 2022c). The code gives a step-by-step explanation on how to generate the asset tree structure from data in a csv file or from data that is already in Seeq.

For the development and testing of the add-on, four asset trees have been generated from the SA data set which contains controller outputs, process variables, setpoints and auto manual modes. Since the process from which the SA data set originates is not known for reasons of anonymisation, the asset tree structures were freely invented. Three of the four asset trees comprise the same 20 controllers with four signals each which results in a total of 80 control signals. These asset trees are called “Plant 1”, “Plant 2” and “Plant 3” and differ in the structure of the assets to test whether the add-on can recognize and interpret different structures correctly. The different structures are shown in Figure 32 to Figure 34. The fourth asset tree is called “Plant” and comprises 111 controllers with four signals each which results in a total of 444 control signals. This asset tree was created to test the add-on’s capability of analysing a large number of signals and to develop the use cases in Section 5.4. The “Plant” asset tree is shown in Figure 35.

The “Plant 1” asset tree has two sections which are illustrated in Figure 32. Section A and Section B both have two units where each unit has five controllers assigned to it. Every controller in the asset tree has four control signals assigned to it which are named “Controller Output”, “Process Variable”, “Setpoint” and “Mode”.

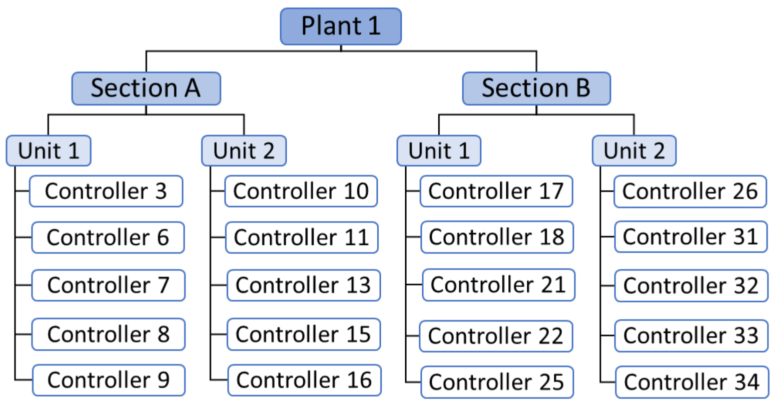


*Figure 32: Structure of the "Plant 1" asset tree*

The “Plant 2” asset tree also has two sections. Section A and Section B have two units and each unit in turn has four controllers assigned to it. Two controllers are on the same hierarchical level as the units as illustrated in Figure 33. The “Plant 2” asset tree is used in the example workflow in Section 5.3.1.

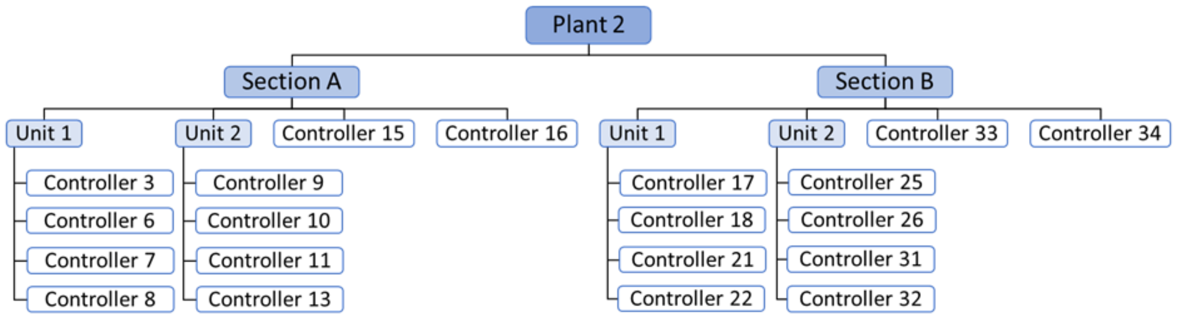


*Figure 33: Structure of the "Plant 2" asset tree*

In the “Plant 3” asset tree, all 20 controllers are directly assigned to the top level of the hierarchy as shown in Figure 34.

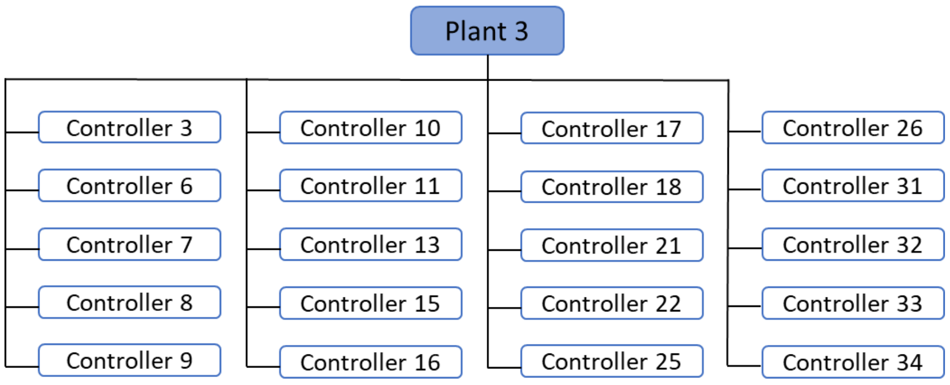


*Figure 34: Structure of the "Plant 3" asset tree*

The “Plant” asset tree in Figure 35 contains two sections where each section has six units. Every unit comprises between eight and eleven controllers. Every controller has four control signals assigned to it which are named “Controller Output”, “Process Variable”, “Setpoint” and “Mode”.

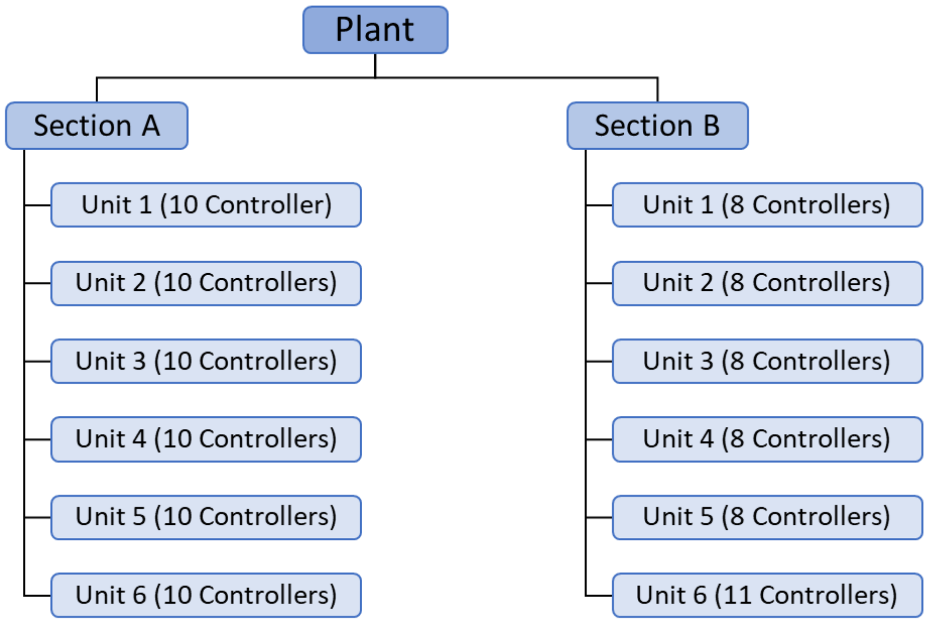
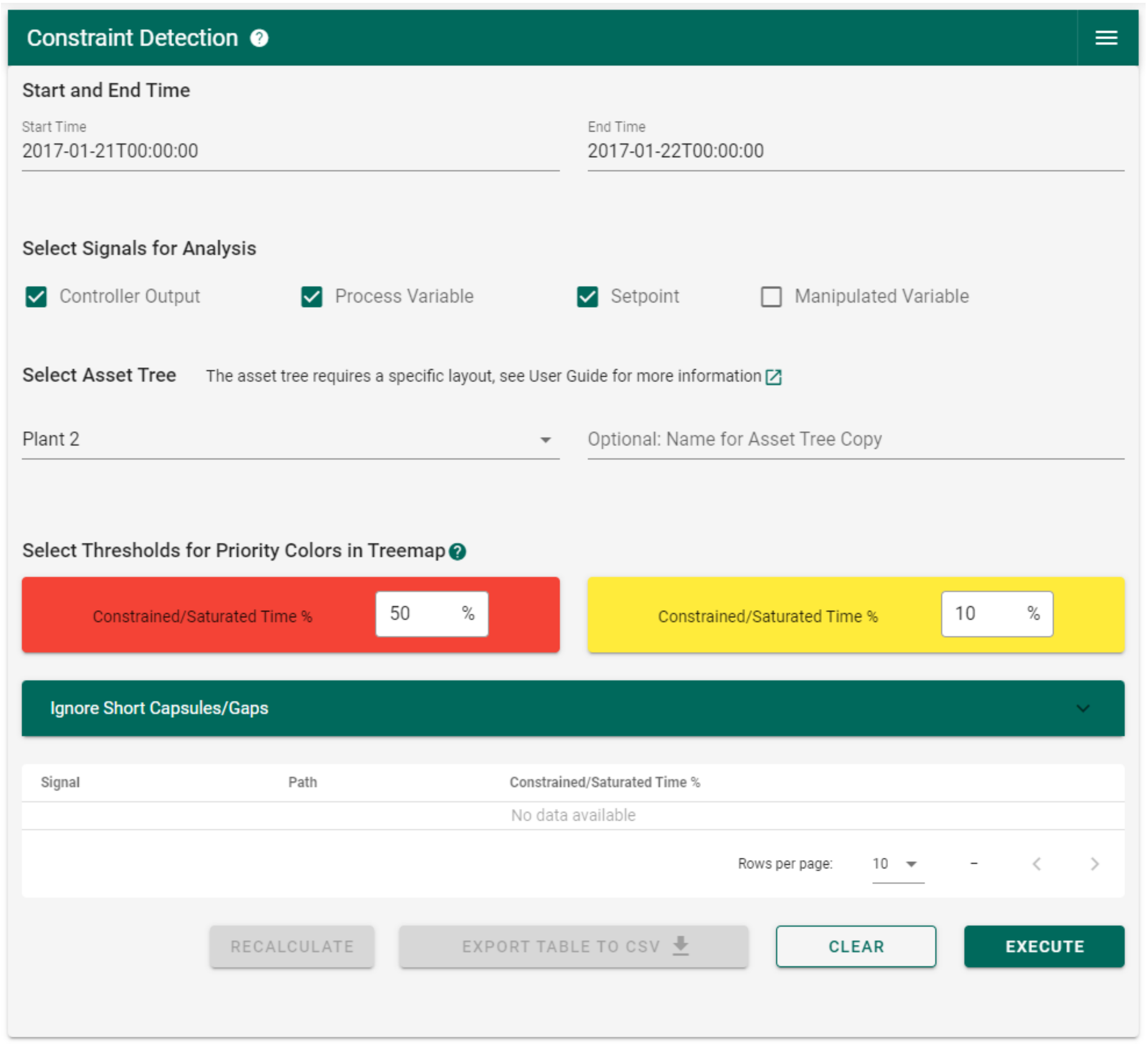


*Figure 35: Structure of the "Plant" asset tree*

## 5.3 Workflow and User Interface

### 5.3.1 Example Workflow

The UI of the constraint detection add-on is shown in Figure 36. The add-on function is explained by means of an example workflow in which all add-on inputs and outputs are presented. In this example workflow, all controller outputs, process variables and setpoints in the “Plant 2” asset tree are analysed for saturation and constraints in the period from 21.01.2017 to 22.01.2017. The “Plant 2” asset tree was explained in Section 5.2.



Constraint Detection

Start and End Time

Start Time: 2017-01-21T00:00:00

End Time: 2017-01-22T00:00:00

Select Signals for Analysis

[✓ Controller Output] [✓ Process Variable] [✓ Setpoint] Manipulated Variable

Select Asset Tree — The asset tree requires a specific layout, see User Guide for more information

Plant 2

Optional: Name for Asset Tree Copy

Select Thresholds for Priority Colors in Treemap

Constrained/Saturated Time %: 50 %

Constrained/Saturated Time %: 10 %

Ignore Short Capsules/Gaps

| Signal | Path | Constrained/Saturated Time % |
|---|---|---|
| | No data available | |

Rows per page: 10 –

*Figure 36: User interface of the constraint detection add-on*

**1)** In Seeq Workbench, the constraint detection add-on can be opened by clicking on “Constraint Detection” in the add-on menu in the tools panel. The add-on menu is shown in Figure 37. After clicking on “Constraint Detection” the UI opens in a new browser window. The add-on imports the display range from the active worksheet and all asset trees which are scoped to the workbook that the user is working in.

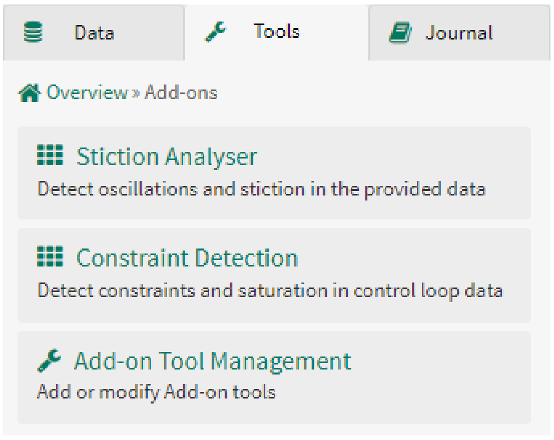

Data
Tools
Journal
Overview » Add-ons
Stiction Analyser
Detect oscillations and stiction in the provided data
Constraint Detection
Detect constraints and saturation in control loop data
Add-on Tool Management
Add or modify Add-on tools

*Figure 37: Add-on menu in the tools panel in Seeq Workbench*

**2)** The user must specify a start and end time for the analysis. By default, the add-on will insert the start and end of the display range of the active Seeq worksheet. The input fields provide a hint with the required date format YYYY-MM-DDTHH:MM:SS as can be seen in Figure 38. For the example workflow, the start time is defined as 2017-01-21T00:00:00 and the end time as 2017-01-22T00:00:00.

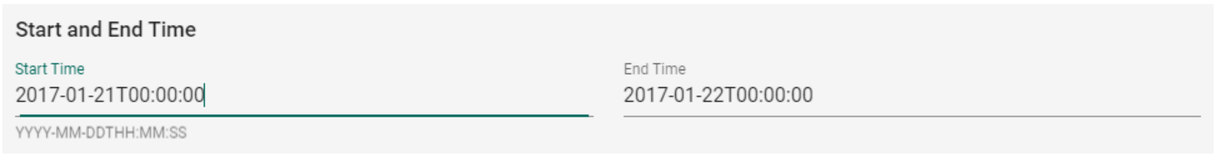

Start and End Time
Start Time 2017-01-21T00:00:00
YYYY-MM-DDTHH:MM:SS
End Time 2017-01-22T00:00:00

*Figure 38: Selection of the start and end time for the analysis*

**3)** The user must select the control signals to be analysed by activating checkboxes as can be seen in Figure 39. For example, if only the controller output signals are to be analysed, only the checkbox for 'Controller Output' must be activated. It is also possible to execute the analysis for multiple control signals at once. For every selected control signal, a new Seeq worksheet with a treemap is generated by the add-on. In this example, the add-on creates three worksheets with a controller output treemap, process variable treemap and setpoint treemap since these three signals are selected.

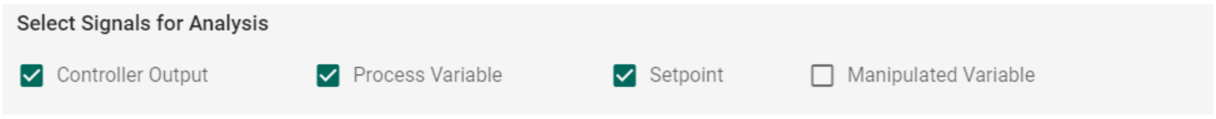

Select Signals for Analysis
☑ Controller Output ☑ Process Variable ☑ Setpoint ☐ Manipulated Variable

*Figure 39: Selection of control signals for the analysis*

**4)** The user must select an asset tree from the dropdown menu which is shown on the left side in Figure 40. In the dropdown menu, all asset trees appear which are scoped to the Seeq workbook. The "Plant 2" asset tree is selected for the example workflow. In Figure 40, the user is informed that the asset tree requires a certain layout to be processed by the add-on.

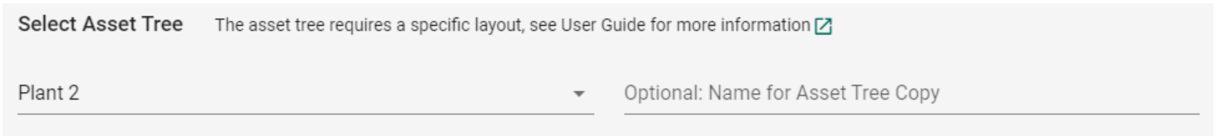
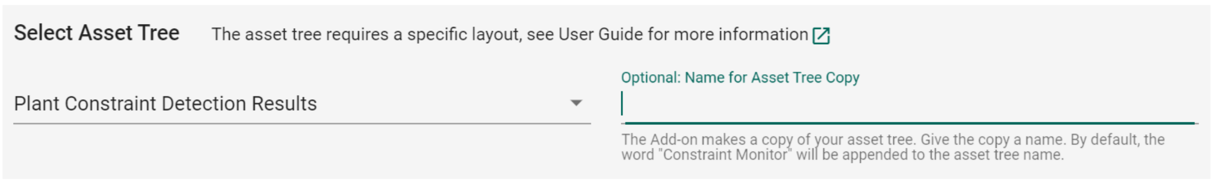

Select Asset Tree The asset tree requires a specific layout, see User Guide for more information
Plant 2
Optional: Name for Asset Tree Copy

*Figure 40: Selection of an asset tree for the analysis*

**5)** Optionally, a name for the asset tree copy can be specified. The constraint detection add-on generates a copy of the selected input asset tree in the analysis period. After the analysis, the asset tree copy with the specified name will show up in the data panel in Seeq Workbench. If the text field is left empty, the word ‘Constraint Monitor’ will be appended to the asset tree name. For example, if the asset tree name is “Plant 2”, the new asset tree will be named “Plant 2 Constraint Monitor”. The input field provides a hint which shortly explains its function as shown in Figure 41.


Select Asset Tree

The asset tree requires a specific layout, see User Guide for more information

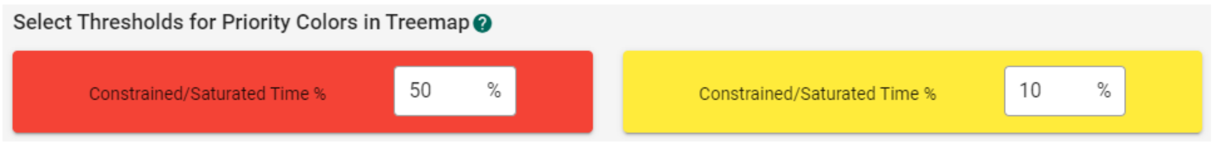
Plant Constraint Detection Results

Optional: Name for Asset Tree Copy

The Add-on makes a copy of your asset tree. Give the copy a name. By default, the word "Constraint Monitor" will be appended to the asset tree name.

*Figure 41: Specification of the output asset tree name*

**6)** In a treemap in Seeq, every asset is shown as a coloured panel. The colour of the panels is determined by user-specified conditions. In the constraint detection add-on, the colours in the treemap are set according to thresholds for the “Constrained Time %” or “Saturated Time %”. By default, the thresholds are set to 50 % and 10 % but they can be adjusted by the user which is shown in Figure 42.


Select Thresholds for Priority Colors in Treemap

Constrained/Saturated Time % 50 %

Constrained/Saturated Time % 10 %

*Figure 42: Specification of thresholds for the "Constrained/Saturated Time %"*

The “Constrained Time %” or “Saturated Time %” is the time-percentage a signal is constrained/saturated in the analysed time period. These terms describe the same statistic as the “Saturation Index” in Chapter 3 but it was replaced because it was found to be less intuitive than the term “Saturated Time %” or “Constrained Time %”. The term “Constrained Time %” is used for constraints in the setpoint, process variable and manipulated variable. The term “Saturated Time %” is used for saturation in the controller output. The colours for the panels in the treemap are determined as shown in Figure 43. If the “Constrained/Saturated Time %” is equal to or larger than the red threshold in Figure 42 than the “High Constraint/Saturation” condition is true, and the controller panel is coloured red. If the “Constrained/Saturated Time %” is smaller than the red threshold but equal to or larger than the yellow threshold, the “Medium Constraint/Saturation” condition is true, and the controller panel is coloured yellow. If none of the conditions is true, the controller panel is coloured green. The “High Constraint/Saturation” condition and “Medium Constraint/Saturation” condition also contain the information for the time capsules in the time trend.

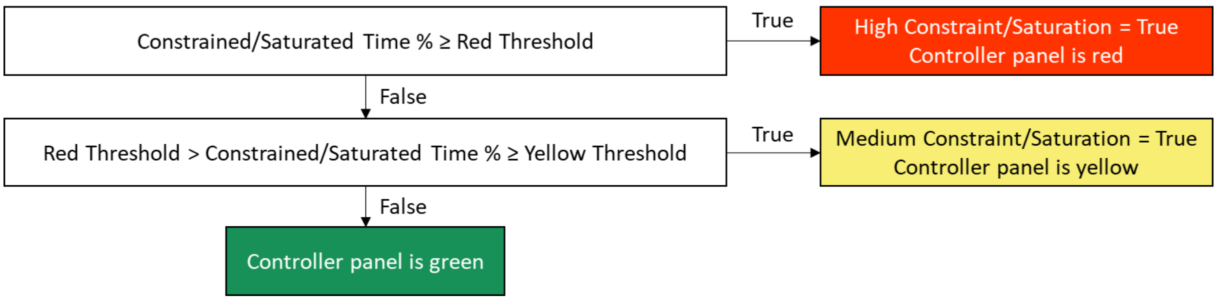
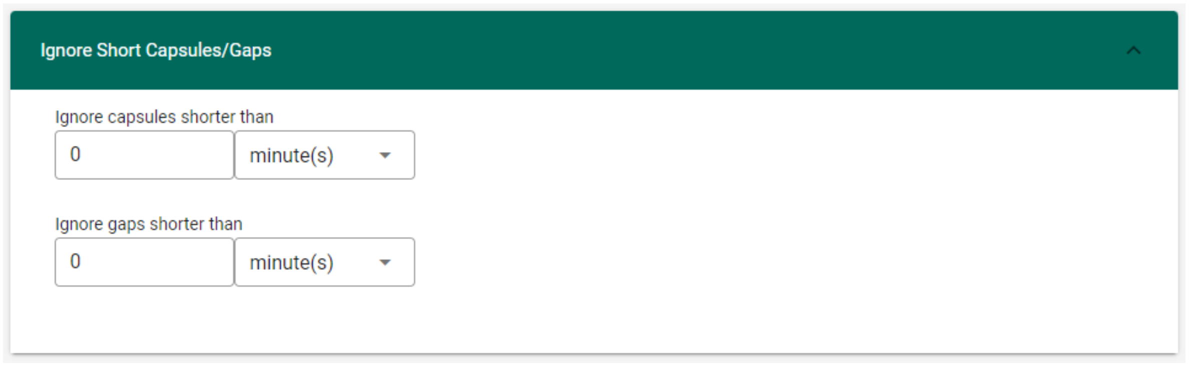

Constrained/Saturated Time % ≥ Red Threshold
True
High Constraint/Saturation = True
Controller panel is red
False
Red Threshold > Constrained/Saturated Time % ≥ Yellow Threshold
True
Medium Constraint/Saturation = True
Controller panel is yellow
False
Controller panel is green

*Figure 43: Logic for setting the colours in the treemap*

**7)** The constraint detection add-on generates time capsules for constrained/saturated periods in the analysed control signal. The time capsules can be seen in the time trend in Seeq Workbench. Depending on the analysed signal the time capsules can be very short or interrupted. Seeq provides a functionality in many of its tools which ignores short time capsules and closes short gaps between time capsules. This functionality was also implemented in the add-on as an optional input for the analysis. Short capsules and short gaps can be specified in the expansion panel which is shown in Figure 44. For example, if “5 minute(s)” is defined as a short capsule all detected constrained/saturated periods which are shorter than 5 minutes would not appear as time capsules in the time trend. The ignored capsules are also not considered for the calculation of the “Constrained/Saturated Time %”. In this workflow example, no short capsules or gaps are specified.


Ignore Short Capsules/Gaps

Ignore capsules shorter than: 0 minute(s)

Ignore gaps shorter than: 0 minute(s)

*Figure 44: Specification of short capsules and gaps*

**8)** The analysis can be started by clicking the “Execute” button. After clicking the button, a progress bar appears, and the UI is disabled so that the user cannot make changes to the input fields while the add-on is executing the analysis. The progress bar can be seen in Figure 45. If the input asset tree contains several hundred signals and the analysis is executed for a large time interval, then the analysis can take around 15 minutes. The progress bar should communicate to the user how far the analysis has progressed, and that the add-on is still processing. For the example workflow, the analysis takes less than a minute.

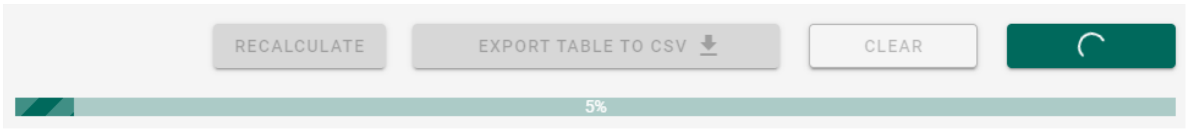
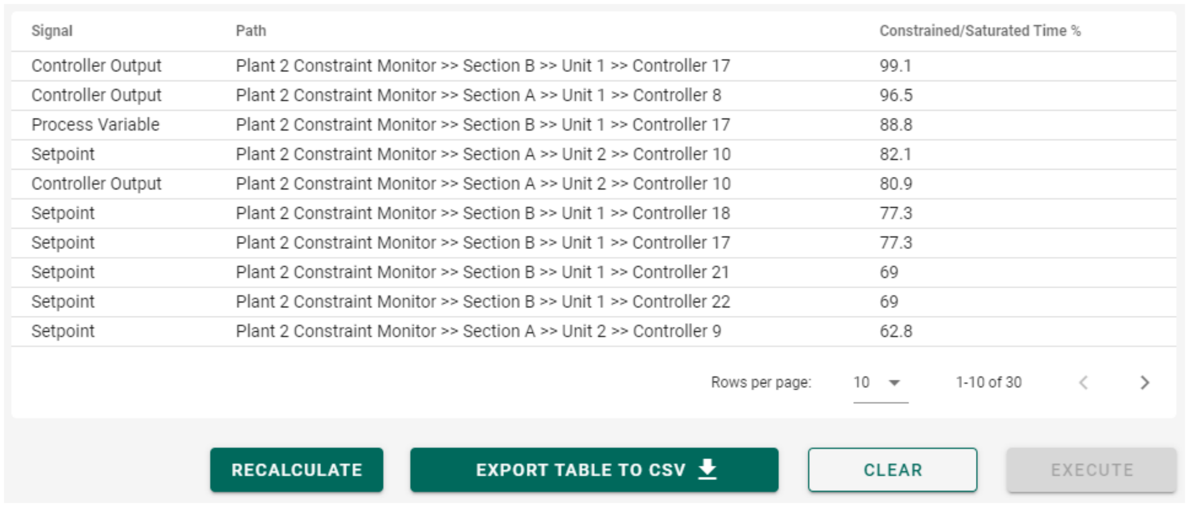


*Figure 45: Progress bar in the UI*

**9)** When the analysis is completed, the UI is partly enabled again. The table in the UI becomes populated and shows a top 30 of the signals with the highest “Constrained/Saturated Time %” in descending order.



| Signal | Path | Constrained/Saturated Time % |
|---|---|---|
| Controller Output | Plant 2 Constraint Monitor >> Section B >> Unit 1 >> Controller 17 | 99.1 |
| Controller Output | Plant 2 Constraint Monitor >> Section A >> Unit 1 >> Controller 8 | 96.5 |
| Process Variable | Plant 2 Constraint Monitor >> Section B >> Unit 1 >> Controller 17 | 88.8 |
| Setpoint | Plant 2 Constraint Monitor >> Section A >> Unit 2 >> Controller 10 | 82.1 |
| Controller Output | Plant 2 Constraint Monitor >> Section A >> Unit 2 >> Controller 10 | 80.9 |
| Setpoint | Plant 2 Constraint Monitor >> Section B >> Unit 1 >> Controller 18 | 77.3 |
| Setpoint | Plant 2 Constraint Monitor >> Section B >> Unit 1 >> Controller 17 | 77.3 |
| Setpoint | Plant 2 Constraint Monitor >> Section B >> Unit 1 >> Controller 21 | 69 |
| Setpoint | Plant 2 Constraint Monitor >> Section B >> Unit 1 >> Controller 22 | 69 |
| Setpoint | Plant 2 Constraint Monitor >> Section A >> Unit 2 >> Controller 9 | 62.8 |

*Figure 46: Table with top 30 of the most constrained/saturated signals*

The table can be downloaded by clicking the “Export Table to CSV” button. In addition to the “Constrained/Saturated Time %”, the table also shows the control signal type and the path which specifies where the constrained/saturated signal is located in the asset tree. Furthermore, the table is sortable by every column. For the example workflow with “Plant 2”, the signal with the highest “Saturated Time %” is a controller output which located in “Plant 2 Constraint Monitor >> Section B >> Unit 1 >> Controller 17” as can be seen in Figure 46. In the UI, the “Execute” button and all input fields which were explained in step 2 to 6 are disabled after the add-on execution. The expansion panel for the specification of short gaps and capsules is still enabled. The “Recalculate” button and “Export Table to CSV” button were enabled. The function of the “Recalculate” button is explained in step 13.

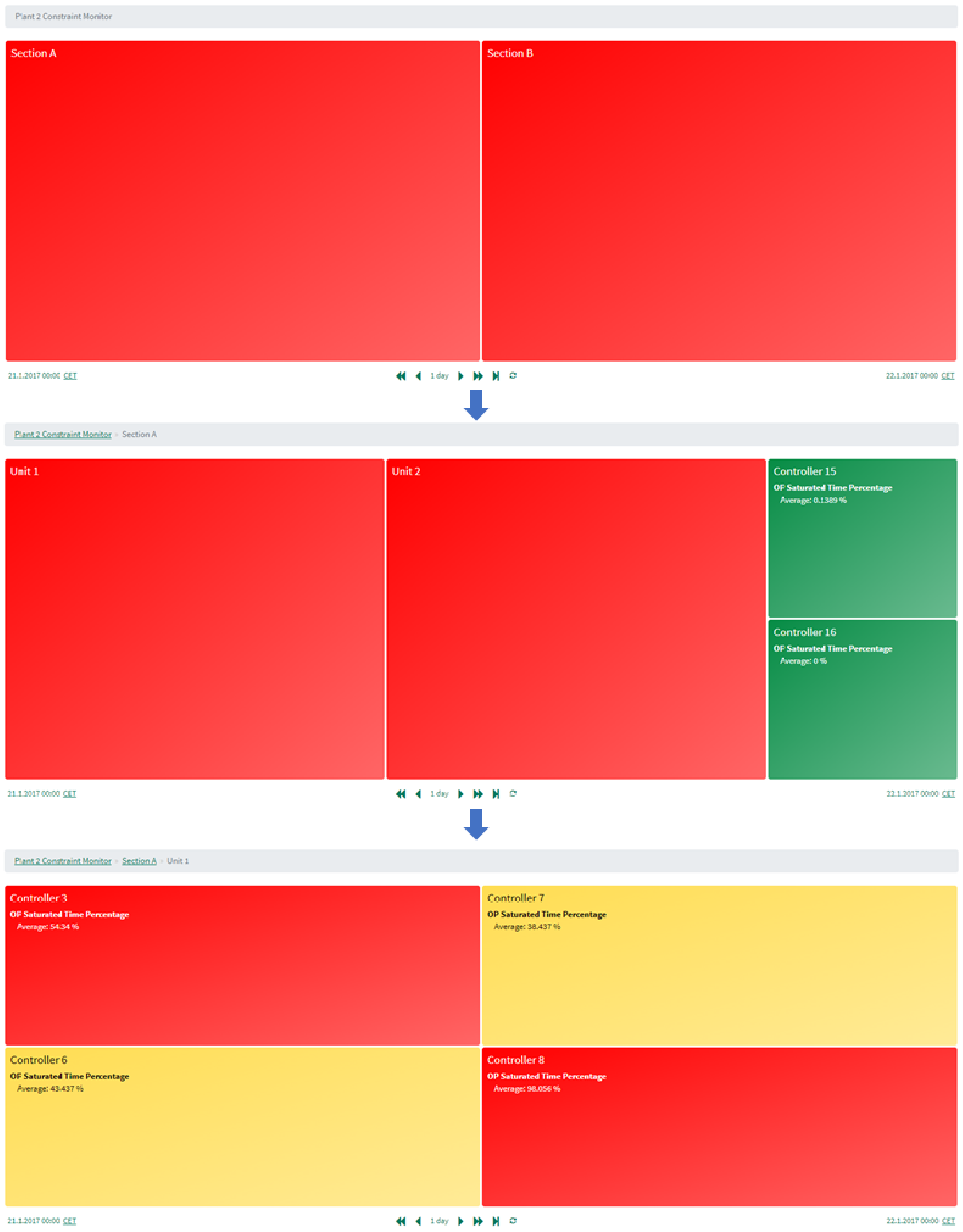
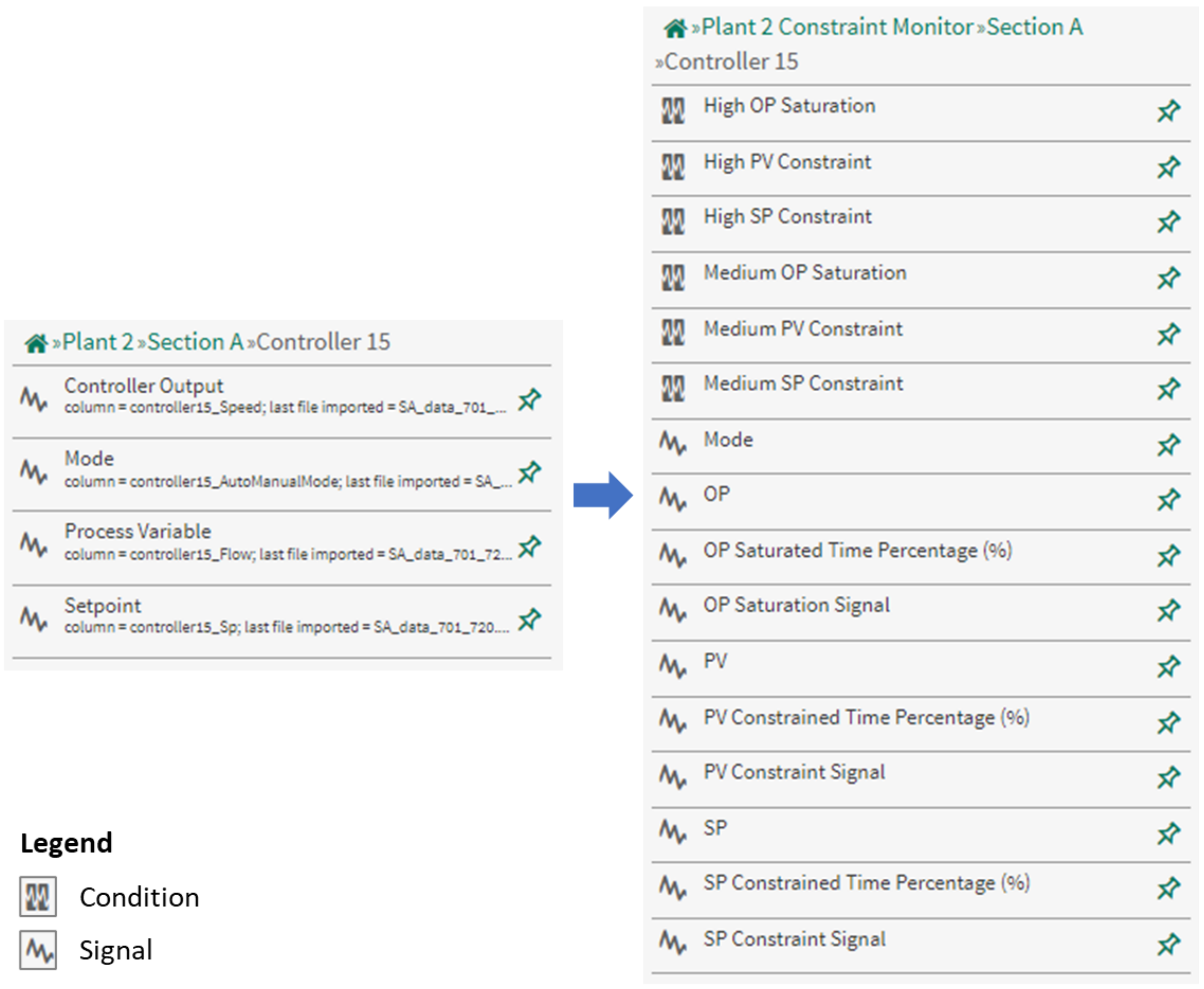


*Figure 47: Illustration of the use of a treemap in Seeq Workbench*

**10)** After the add-on execution, the web page with the Seeq workbook must be refreshed. For every signal checkbox that was activated in step 3, a new worksheet with a treemap was generated. For additional insights, the 'Constrained Time %' or 'Saturated Time %' can be selected as a statistic to be displayed in the treemap. The bottom of Figure 47 shows the treemap in the new worksheet called "OP Saturation Treemap" with the "Saturated Time %" for the four controllers in "Plant 2 Constraint Monitor >> Section A >> Unit 1".

**11)** The user can explore the treemaps and use the table in the UI as guidance. In the treemap, the user can move from the upper hierarchical levels to lower levels by clicking on a panel. This workflow is illustrated in Figure 47. The user can start on the top level of the asset tree with Section A and Section B. When the user clicks on the "Section A" panel, the treemap changes to "Plant 2 Constraint Monitor >> Section A". In the next step, the user can click on the "Unit 1" panel and the treemap changes to "Plant 2 Constraint Monitor >> Section A >> Unit 1". In this way, the user can move through the whole treemap. When the user clicks on a controller panel, the time trend of the corresponding controller opens.

**12)** In addition to the automatically generated treemaps, the newly generated asset tree is also of interest to the user. In Figure 48, the input asset tree "Plant 2" is shown on the left and the output asset tree "Plant 2 Constraint Monitor" is shown on the right for all signals and conditions that are assigned to controller 15.


»Plant 2 »Section A »Controller 15

Controller Output
column = controller15_Speed; last file imported = SA_data_701_...

Mode
column = controller15_AutoManualMode; last file imported = SA_...

Process Variable
column = controller15_Flow; last file imported = SA_data_701_72...

Setpoint
column = controller15_Sp; last file imported = SA_data_701_720....

»Plant 2 Constraint Monitor »Section A »Controller 15

High OP Saturation
High PV Constraint
High SP Constraint
Medium OP Saturation
Medium PV Constraint
Medium SP Constraint
Mode
OP
OP Saturated Time Percentage (%)
OP Saturation Signal
PV
PV Constrained Time Percentage (%)
PV Constraint Signal
SP
SP Constrained Time Percentage (%)
SP Constraint Signal

**Legend**

| | |
|---|---|
| ▨ | Condition |
| 〰 | Signal |

Figure 48: Signals and conditions in the input asset tree and output asset tree

In the input asset tree, controller 15 has four signals assigned to it. In the output asset tree, controller 15 has ten signals and six conditions assigned to it. For every analysed signal, two new signals and two new conditions were added. The added signals will be explained exemplarily for the controller output. The controller output signal is renamed to OP for more concise signal naming in the output asset tree. The add-on analyses the OP signal and generates the “OP Saturation Signal” which is 0 if no saturation is detected and 1 if saturation is detected (see Section 3.2.1). From the “OP Saturation Signal”, the “OP Saturated Time Percentage” is calculated. From the “OP Saturated Time Percentage”, it can be assessed whether the “High OP Saturation” condition or the “Medium OP Saturation” condition are true or false. The newly generated signals can be used for further analysis using other tools in Seeq Workbench.

**13)** Optionally, short capsules and gaps can still be defined after the first add-on execution by specifying them in the expansion panel and clicking on the “Recalculate” button. This affects the time capsules in the time trend and the calculated “Constrained/Saturated Time %”. This step can be executed iteratively. In Figure 49, the effect on the time capsules is shown. On the left side, short capsules and gaps are set to 0. On the right side, short capsules and gaps are both set to 5 minutes. Therefore, the left picture shows multiple shorter time capsules, and the right picture shows one single time capsule.

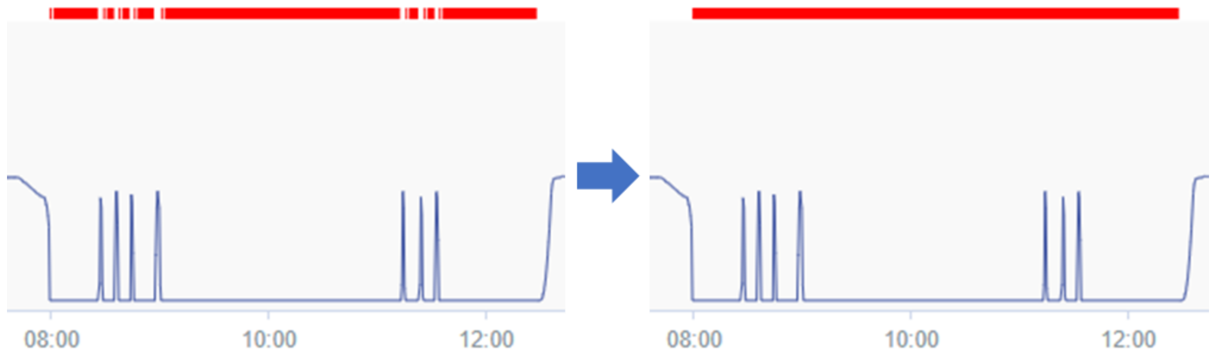


*Figure 49: Time capsules in the time trend*

**14)** If the user wants to start a new analysis, the “Clear” button can be used to reset the UI to its initial state as shown in Figure 36. Alternatively, the add-on can be closed by closing the browser window.

### 5.3.2 Summary of the Workflow

The inputs and outputs of the constraint detection add-on are summarised in Table 10. A video of the example workflow can be found in the user guide in the public add-on documentation (Tiedemann 2022e). The constraint detection add-on was tested successfully on all four asset trees which were presented in Section 5.2.

*Table 10: Inputs and outputs of the constraint detection add-on*

| | |
|---|---|
| **Input** | ▪ Input asset tree<br>▪ Start and end time of analysis period<br>▪ Thresholds for visualisation in treemap<br>▪ Other input options for higher user-friendliness:<br>    ▪ Selection of control signals to be analysed<br>    ▪ Short capsules and gaps (optional)<br>    ▪ Name for the output asset tree (optional) |
| **Output** | ▪ Output asset tree with:<br>    ▪ Input control signals<br>    ▪ Constraint/saturation signal<br>    ▪ Conditions with time capsules for saturated/constrained periods:<br>        ▪ “High Constraint/Saturation” condition<br>        ▪ “Medium Constraint/Saturation” condition<br>    ▪ Measure for the extent of saturation/constraints:<br>        ▪ “Constrained Time %” or “Saturated Time %”<br>▪ Treemap coloured according to the “Constrained/Saturated Time %”<br>▪ Other outputs for higher user-friendliness:<br>    ▪ Table in the UI with a top 30 of the most constrained/saturated periods<br>    ▪ Option to export the table to a CSV file |

### 5.3.3 Support in the User Interface and Documentation

In the UI of the constraint detection add-on, the user can find multiple options to get support. These options are marked in red in Figure 50. The UI provides two question mark icons. The question mark icon in the app bar on the top opens a dialog window with a general explanation of the add-on which is shown in Figure 51. The question mark icon in the threshold section opens a dialog window which explains the terms “Constrained Time %” and “Saturated Time %”, and how they are used for the different colours in the treemap. The dialog window is shown in Figure 52. The question mark icons with their corresponding dialog windows provide immediate support while the user is filling in the UI.

The menu in the top right corner provides a “Support” option and a “Documentation” option. The “Support” option redirects the user to the GitHub issues of the add-on (HAW Process Automation). In the GitHub issues, the user can open a new issue and ask for support or report errors. The Department of Process Automation at the Hamburg University of Applied Sciences will process the reported issues. The “Documentation” option in the menu redirects the user to the add-on documentation (Tiedemann 2022f). In the documentation, the user can find a user guide, installation instructions, use cases, release notes and other information on the add-on. In the “Select Asset Tree” section of the UI, a direct link to the user guide is provided where the required structure of the asset tree is explained (Tiedemann 2022a). The explanation also provides a link to template code that gives a step-by-step explanation on how to generate the required structure from data that is already in Seeq or from data in a csv file (Tiedemann 2022c). The template code should facilitate the use of the add-on. In summary, the links to the documentation and to the template code address Seeq users who are working with the constraint detection add-on for the first time. The “Support” option is for Seeq users who have already used the add-on but are experiencing problems that cannot be solved by reading the documentation.

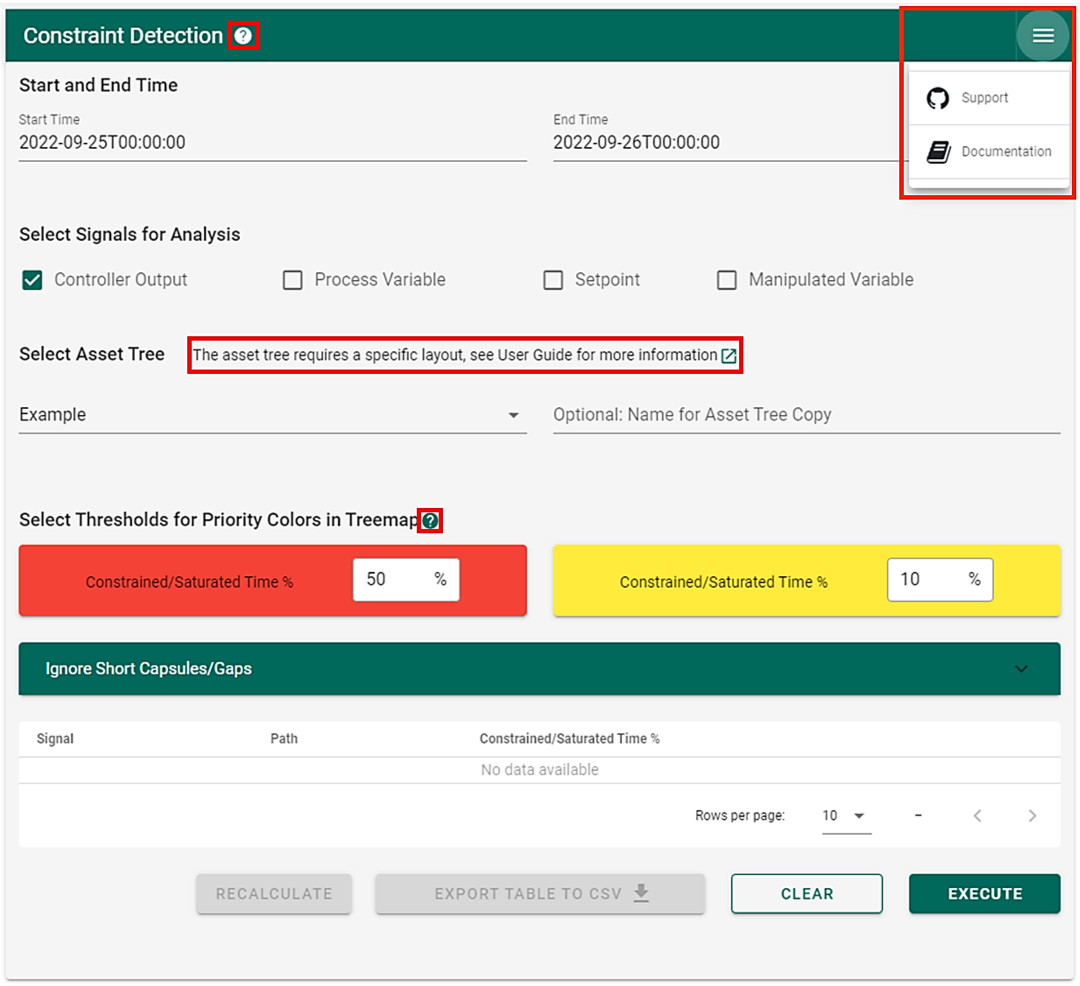
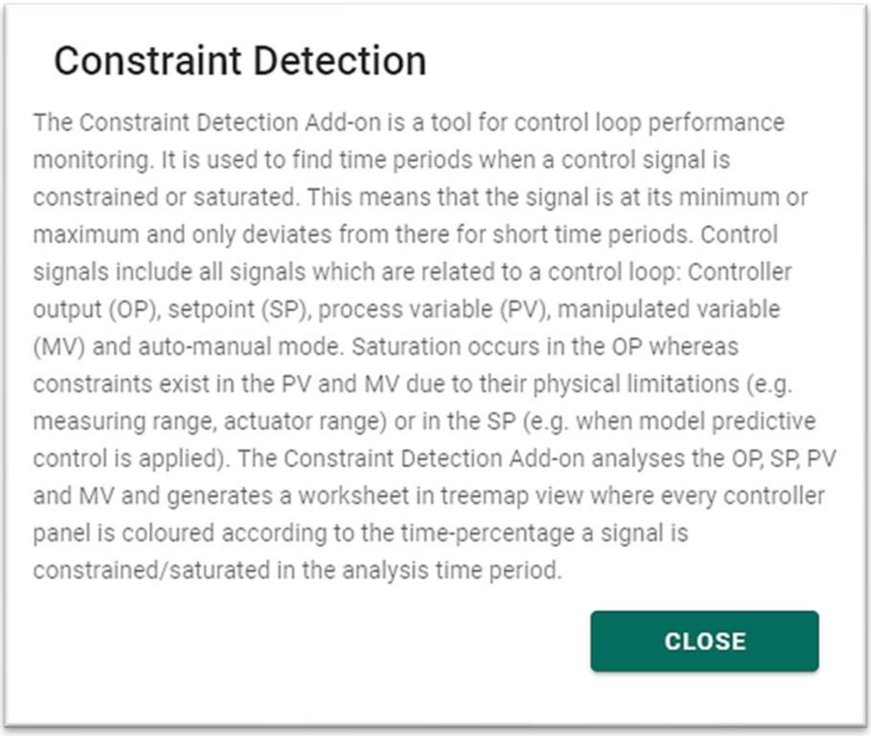

Constraint Detection
Support
Documentation
Start and End Time
Start Time
2022-09-25T00:00:00
End Time
2022-09-26T00:00:00
Select Signals for Analysis
Controller Output
Process Variable
Setpoint
Manipulated Variable
Select Asset Tree
The asset tree requires a specific layout, see User Guide for more information
Example
Optional: Name for Asset Tree Copy
Select Thresholds for Priority Colors in Treemap
Constrained/Saturated Time %
50 %
Constrained/Saturated Time %
10 %
Ignore Short Capsules/Gaps
Signal
Path
Constrained/Saturated Time %
No data available
Rows per page: 10
RECALCULATE
EXPORT TABLE TO CSV
CLEAR
EXECUTE

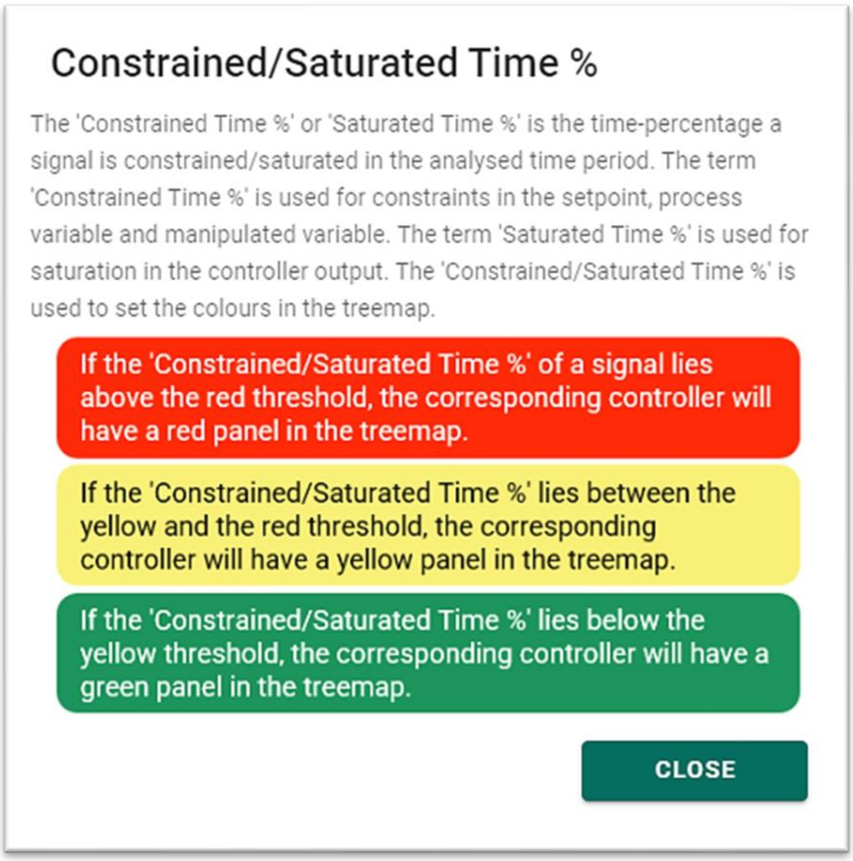
*Figure 50: Support options in the UI*



Constraint Detection

The Constraint Detection Add-on is a tool for control loop performance monitoring. It is used to find time periods when a control signal is constrained or saturated. This means that the signal is at its minimum or maximum and only deviates from there for short time periods. Control signals include all signals which are related to a control loop: Controller output (OP), setpoint (SP), process variable (PV), manipulated variable (MV) and auto-manual mode. Saturation occurs in the OP whereas constraints exist in the PV and MV due to their physical limitations (e.g. measuring range, actuator range) or in the SP (e.g. when model predictive control is applied). The Constraint Detection Add-on analyses the OP, SP, PV and MV and generates a worksheet in treemap view where every controller panel is coloured according to the time-percentage a signal is constrained/saturated in the analysis time period.

*Figure 51: Dialog window with explanation about the constraint detection add-on*


Constrained/Saturated Time %

The 'Constrained Time %' or 'Saturated Time %' is the time-percentage a signal is constrained/saturated in the analysed time period. The term 'Constrained Time %' is used for constraints in the setpoint, process variable and manipulated variable. The term 'Saturated Time %' is used for saturation in the controller output. The 'Constrained/Saturated Time %' is used to set the colours in the treemap.

If the 'Constrained/Saturated Time %' of a signal lies above the red threshold, the corresponding controller will have a red panel in the treemap.

If the 'Constrained/Saturated Time %' lies between the yellow and the red threshold, the corresponding controller will have a yellow panel in the treemap.

If the 'Constrained/Saturated Time %' lies below the yellow threshold, the corresponding controller will have a green panel in the treemap.

*Figure 52: Dialog window with explanation about the thresholds for the treemap visualisation*

## 5.4 Use Cases

In this section, three use cases for the constraint detection add-on are presented. All presented use cases are also publicly available for Seeq users in the add-on documentation (Tiedemann 2022d). The "Plant" asset tree, which was introduced in Section 5.2, is used in all three use cases and is analysed in the time span from 21.01.2017 to 28.01.2017. Due to the high number of signals and the analysis period of one week, the add-on execution takes around 15 minutes for every use case.

### 5.4.1 Plant-Wide Analysis of Control Loop Data

The first use case of the constraint detection add-on is the plant-wide analysis of control loop data. This use case largely corresponds to the example workflow in Section 5.3.1. In this use case, all controller outputs, process variables and setpoints in the "Plant" asset tree are analysed for constraints and saturation in the time span from 21.01.2017 to 28.01.2017. The output asset tree is named "Plant Constraint Detection Results" and the thresholds are set to their default values 50 % and 10 %. The UI with all specified inputs is shown in Figure 53.

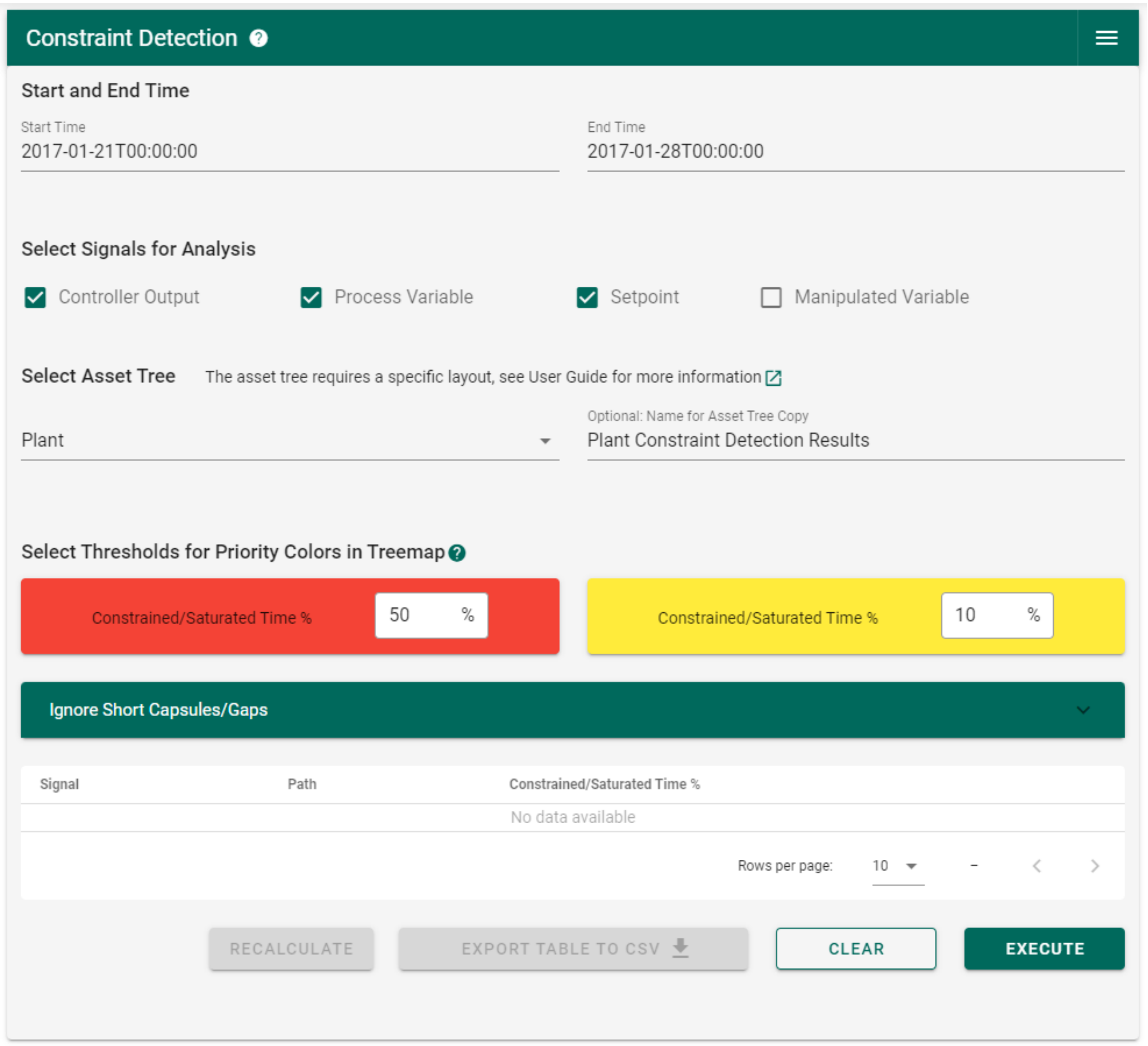
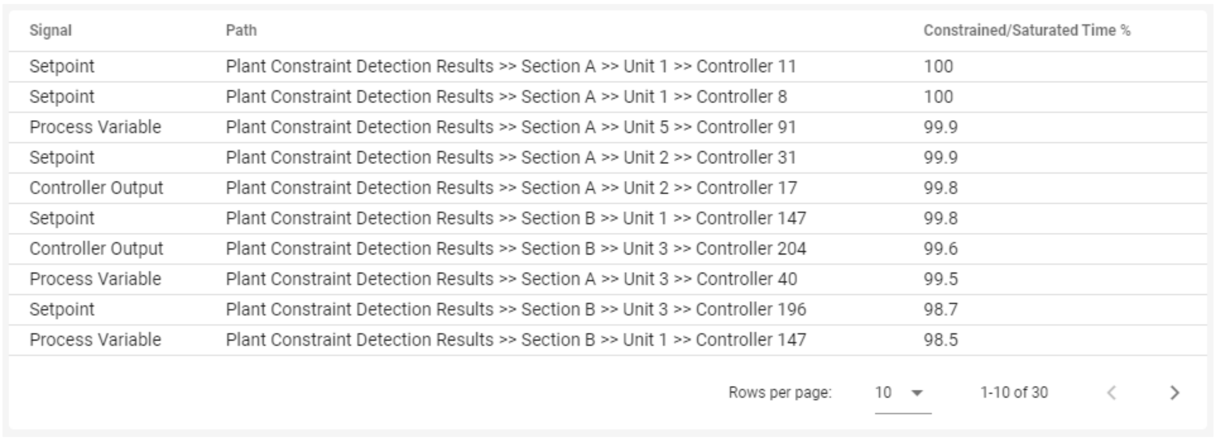

Constraint Detection
Start and End Time
Start Time
2017-01-21T00:00:00
End Time
2017-01-28T00:00:00
Select Signals for Analysis
✓ Controller Output
✓ Process Variable
✓ Setpoint
☐ Manipulated Variable
Select Asset Tree
The asset tree requires a specific layout, see User Guide for more information
Plant
Optional: Name for Asset Tree Copy
Plant Constraint Detection Results
Select Thresholds for Priority Colors in Treemap
Constrained/Saturated Time % 50 %
Constrained/Saturated Time % 10 %
Ignore Short Capsules/Gaps
Signal
Path
Constrained/Saturated Time %
No data available
Rows per page: 10
RECALCULATE
EXPORT TABLE TO CSV
CLEAR
EXECUTE

*Figure 53: UI for plant-wide analysis of control loop data*

After the add-on execution, the table in the UI becomes populated and shows the most constrained/saturated controller outputs, process variables and setpoints which is shown in Figure 54.

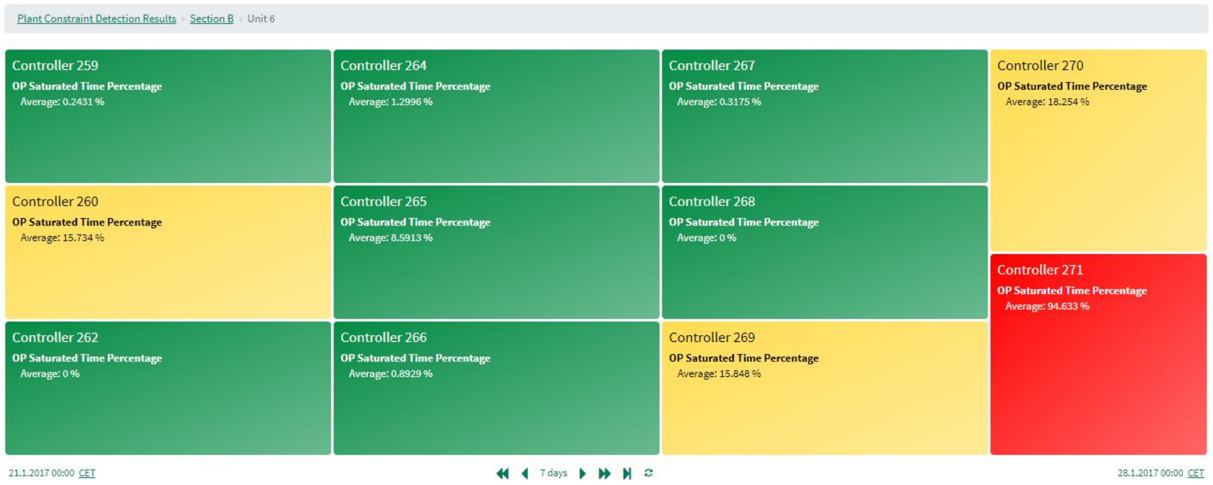
In Seeq Workbench, three new worksheets were generated with treemaps for the analysed control signals. The “Saturated Time %” was selected as a statistic to be displayed in the controller output treemap as illustrated in Figure 55 for “Plant Constraint Detection Results >> Section B >> Unit 6”. By clicking on the “Controller 269” panel the time trend of the corresponding controller can be opened as shown in Figure 56. This use case has a broad applicability and can be used in any plant as long as the requirements for the input asset tree are met. Furthermore, the selection of the control signals and the threshold values can be used to carry out an analysis which is customised for the user.



| Signal | Path | Constrained/Saturated Time % |
|---|---|---|
| Setpoint | Plant Constraint Detection Results >> Section A >> Unit 1 >> Controller 11 | 100 |
| Setpoint | Plant Constraint Detection Results >> Section A >> Unit 1 >> Controller 8 | 100 |
| Process Variable | Plant Constraint Detection Results >> Section A >> Unit 5 >> Controller 91 | 99.9 |
| Setpoint | Plant Constraint Detection Results >> Section A >> Unit 2 >> Controller 31 | 99.9 |
| Controller Output | Plant Constraint Detection Results >> Section A >> Unit 2 >> Controller 17 | 99.8 |
| Setpoint | Plant Constraint Detection Results >> Section B >> Unit 1 >> Controller 147 | 99.8 |
| Controller Output | Plant Constraint Detection Results >> Section B >> Unit 3 >> Controller 204 | 99.6 |
| Process Variable | Plant Constraint Detection Results >> Section A >> Unit 3 >> Controller 40 | 99.5 |
| Setpoint | Plant Constraint Detection Results >> Section B >> Unit 3 >> Controller 196 | 98.7 |
| Process Variable | Plant Constraint Detection Results >> Section B >> Unit 1 >> Controller 147 | 98.5 |

Rows per page: 10 1-10 of 30

*Figure 54: Top 30 table for plant-wide analysis of control loop data*



Plant Constraint Detection Results » Section B » Unit 6

Controller 259
OP Saturated Time Percentage
Average: 0.2431 %

Controller 264
OP Saturated Time Percentage
Average: 1.2996 %

Controller 267
OP Saturated Time Percentage
Average: 0.3175 %

Controller 270
OP Saturated Time Percentage
Average: 18.254 %

Controller 260
OP Saturated Time Percentage
Average: 15.734 %

Controller 265
OP Saturated Time Percentage
Average: 8.5913 %

Controller 268
OP Saturated Time Percentage
Average: 0 %

Controller 271
OP Saturated Time Percentage
Average: 94.633 %

Controller 262
OP Saturated Time Percentage
Average: 0 %

Controller 266
OP Saturated Time Percentage
Average: 0.8929 %

Controller 269
OP Saturated Time Percentage
Average: 15.848 %

21.1.2017 00:00 CET 7 days 28.1.2017 00:00 CET

*Figure 55: Controller output treemap for plant-wide analysis of control loop data*

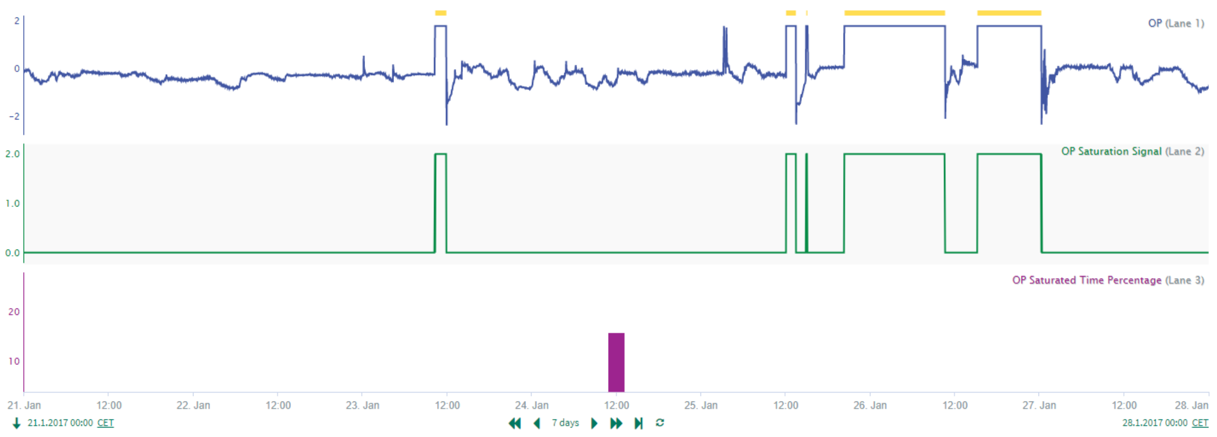
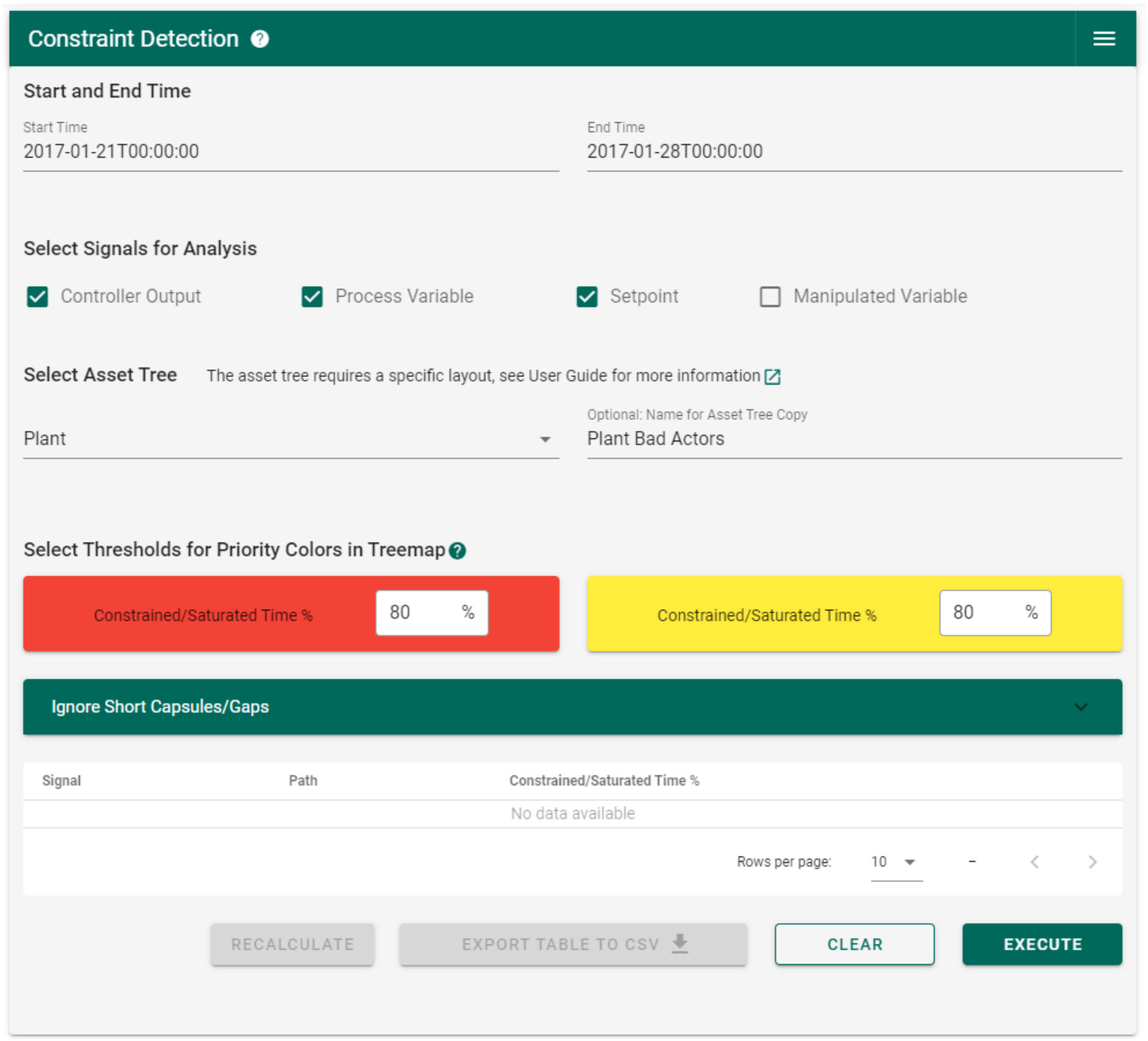


*Figure 56: Time trend for controller 269 for plant-wide analysis of control loop data*

### 5.4.2 Identifying Bad Actors

In this use case, the constraint detection add-on is used for the identification of bad actors. In the context of this use case, bad actors are control signals with a high “Constrained/Saturated Time %”. All controller outputs, process variables and setpoints in the “Plant” asset tree are analysed from 21.01.2017 to 28.01.2017. The output asset tree is named “Plant Bad Actors”. For the identification of bad actors, the thresholds are both set to 80 %. By setting both thresholds to the same value only red and green panels will appear in the treemap. This makes control signals with a high “Constrained/Saturated Time %” stand out more which can be seen in Figure 58 where controller 44 with a “Saturated Time %” of 91 % is eye-catching as it is the only controller with a red panel. A click on the “Controller 44” panel brings the user to the time trend in Figure 59. This use case is a variant of the plant-wide analysis of control loop data from Section 5.4.1 and should demonstrate how the treemap visualisation can be tailored to the user’s needs.


Constraint Detection

Start and End Time

Start Time: 2017-01-21T00:00:00

End Time: 2017-01-28T00:00:00

Select Signals for Analysis

✓ Controller Output ✓ Process Variable ✓ Setpoint ☐ Manipulated Variable

Select Asset Tree — The asset tree requires a specific layout, see User Guide for more information

Plant

Optional: Name for Asset Tree Copy: Plant Bad Actors

Select Thresholds for Priority Colors in Treemap

🟥 Constrained/Saturated Time % 80 %

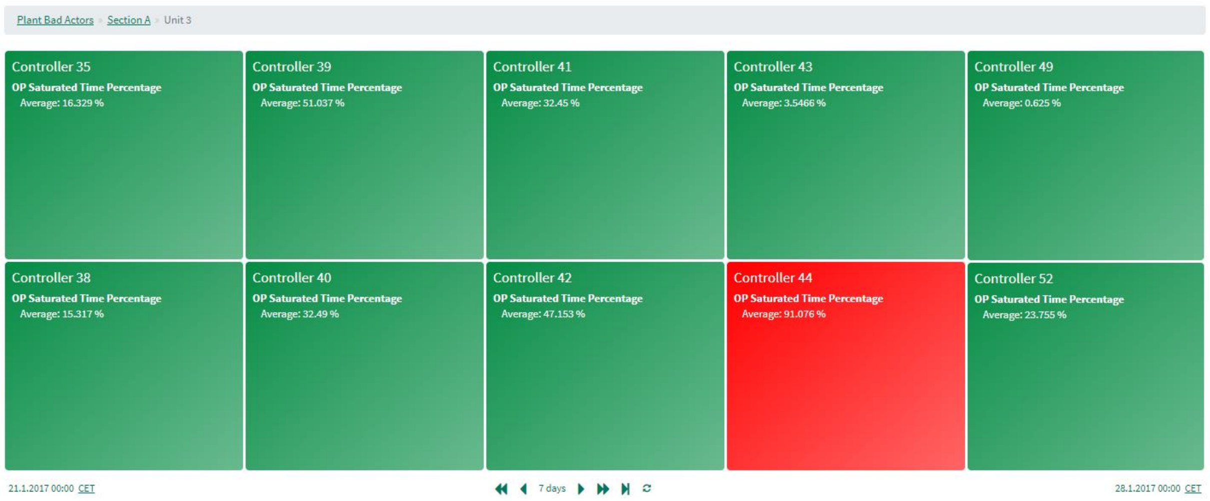
🟨 Constrained/Saturated Time % 80 %

Ignore Short Capsules/Gaps

| Signal | Path | Constrained/Saturated Time % |
|---|---|---|
| | No data available | |

Rows per page: 10 –

Figure 57: UI for identifying bad actors


Plant Bad Actors > Section A > Unit 3

Controller 35
OP Saturated Time Percentage
Average: 16.329 %

Controller 39
OP Saturated Time Percentage
Average: 51.037 %

Controller 41
OP Saturated Time Percentage
Average: 32.45 %

Controller 43
OP Saturated Time Percentage
Average: 3.5466 %

Controller 49
OP Saturated Time Percentage
Average: 0.625 %

Controller 38
OP Saturated Time Percentage
Average: 15.317 %

Controller 40
OP Saturated Time Percentage
Average: 32.49 %

Controller 42
OP Saturated Time Percentage
Average: 47.153 %

Controller 44
OP Saturated Time Percentage
Average: 91.076 %

Controller 52
OP Saturated Time Percentage
Average: 23.755 %

21.1.2017 00:00 CET 7 days 28.1.2017 00:00 CET

Figure 58: *Treemap for identifying bad actors*

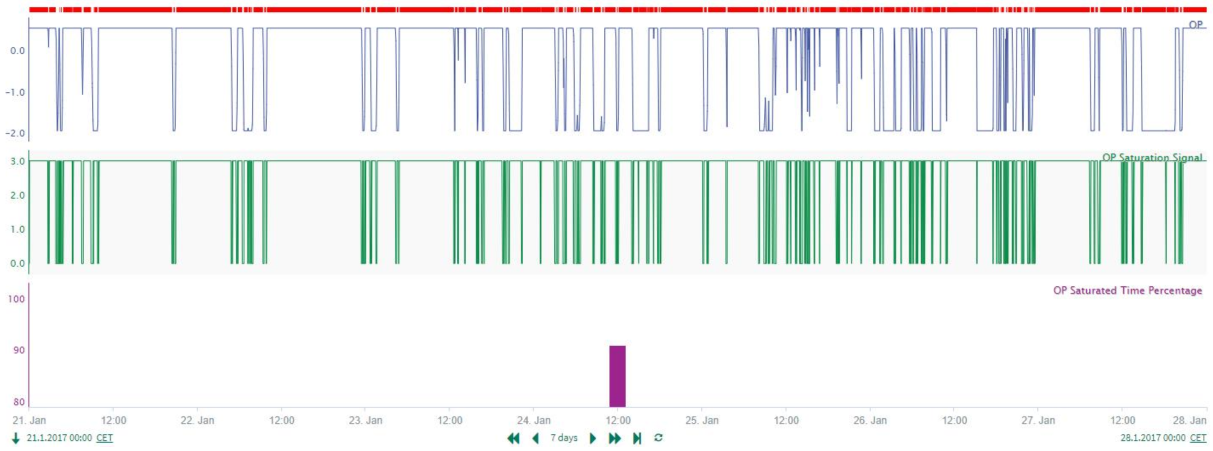
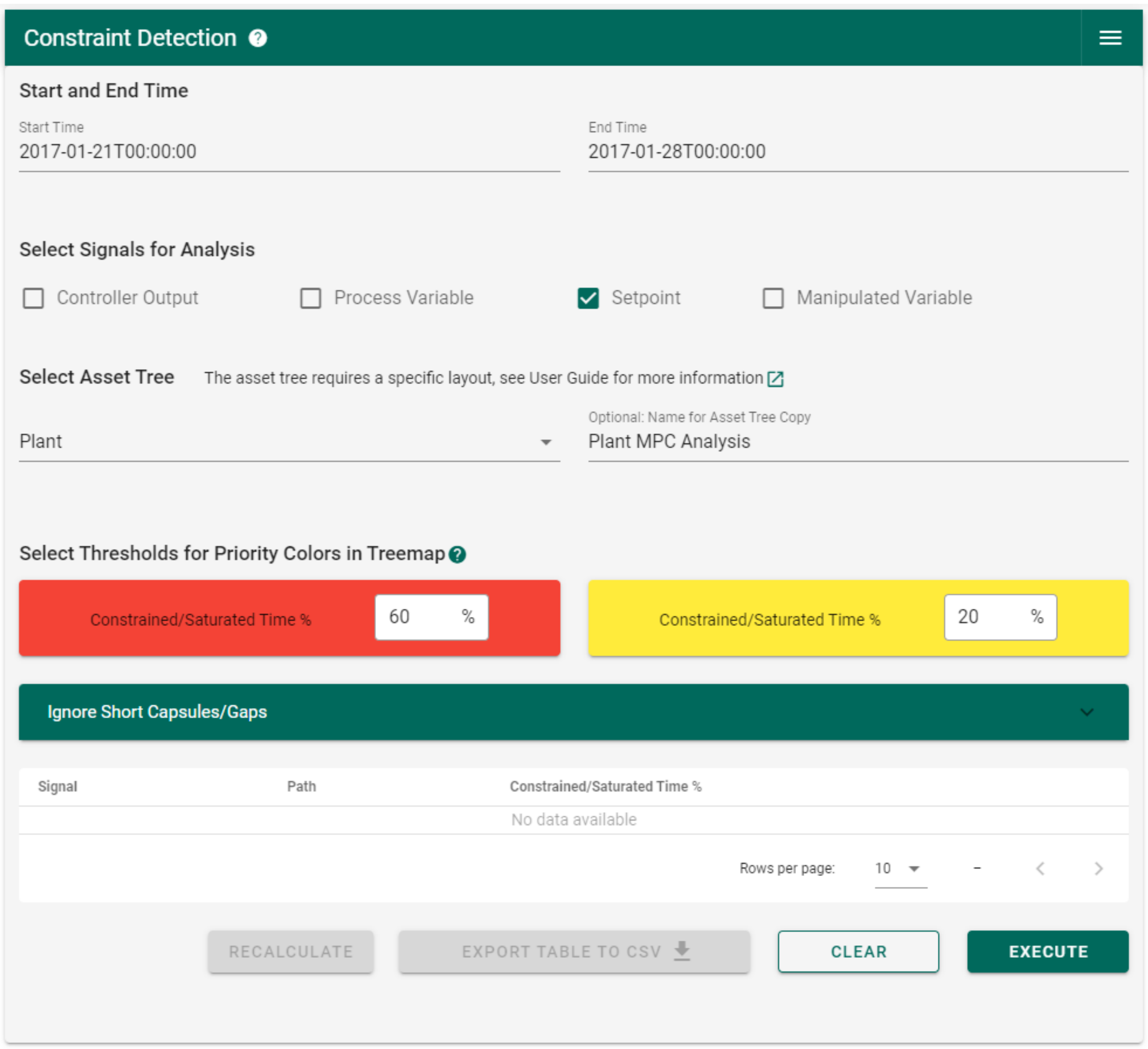


Figure 59: *Time trend of controller 44 for identifying bad actors*

### 5.4.3 Setpoint Analysis for Constrained Model Predictive Control

This use case shows how setpoints can be analysed for constraints when constrained MPC is applied. For this purpose, all setpoints in the “Plant” asset tree are analysed from 21.01.2017 to 28.01.2017. The output asset tree is named “Plant MPC Analysis” and the thresholds are set to 60 % and 20 %.


Constraint Detection

Start and End Time

Start Time: 2017-01-21T00:00:00

End Time: 2017-01-28T00:00:00

Select Signals for Analysis

☐ Controller Output ☐ Process Variable ✓ Setpoint ☐ Manipulated Variable

Select Asset Tree — The asset tree requires a specific layout, see User Guide for more information

Plant

Optional: Name for Asset Tree Copy: Plant MPC Analysis

Select Thresholds for Priority Colors in Treemap

🟥 Constrained/Saturated Time %: 60 %

🟨 Constrained/Saturated Time %: 20 %

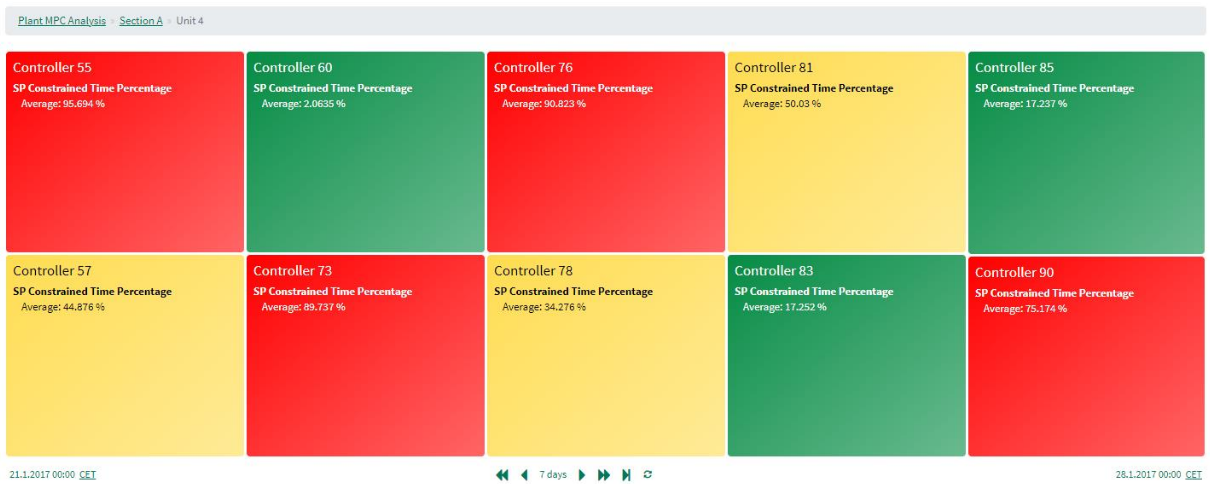
Ignore Short Capsules/Gaps

| Signal | Path | Constrained/Saturated Time % |
|---|---|---|
| | No data available | |

Rows per page: 10 –

*Figure 60: UI for setpoint analysis for constrained MPC*

The resulting treemap is shown in Figure 61 for “Plant MPC Analysis >> Section A >> Unit 4”. The “Constrained Time %” is displayed in the treemap. In constrained MPC, setpoint constraints can be intentional or unintentional. If a setpoint constraint represents a process optimum, then it is intended that the setpoint is constrained as often as possible. In this case, it is beneficial that the setpoint has a high “Constrained Time %”. On the other hand, it can be unintended that a setpoint sticks to its minimum or maximum. In this case, it is unfavourable that the setpoint has a high “Constrained Time %”. In Figure 62 and Figure 63, the setpoint time trends of controller 90 and controller 60 are shown. Both controllers are located under “Plant MPC Analysis >> Section A >> Unit 4” and the time trends were automatically opened by clicking on the controller panel in the treemap. The setpoint of controller 90 has a high “Constrained Time %” whereas the setpoint of controller 60 has a low “Constrained Time %”. In order to determine whether the behaviour of the setpoints is intentional further knowledge about the process is required. This use case should communicate to Seeq users that constraints in control loop data can be intentional or unintentional and that every controller must be assessed individually to determine whether the constraint is problematic and measures for constraint elimination need to be taken.


Plant MPC Analysis » Section A » Unit 4

Controller 55
SP Constrained Time Percentage
Average: 95.694 %

Controller 60
SP Constrained Time Percentage
Average: 2.0635 %

Controller 76
SP Constrained Time Percentage
Average: 90.823 %

Controller 81
SP Constrained Time Percentage
Average: 50.03 %

Controller 85
SP Constrained Time Percentage
Average: 17.237 %

Controller 57
SP Constrained Time Percentage
Average: 44.876 %

Controller 73
SP Constrained Time Percentage
Average: 89.737 %

Controller 78
SP Constrained Time Percentage
Average: 34.276 %

Controller 83
SP Constrained Time Percentage
Average: 17.252 %

Controller 90
SP Constrained Time Percentage
Average: 75.174 %

21.1.2017 00:00 CET 7 days 28.1.2017 00:00 CET

Figure 61: Treemap for setpoint analysis for constrained MPC

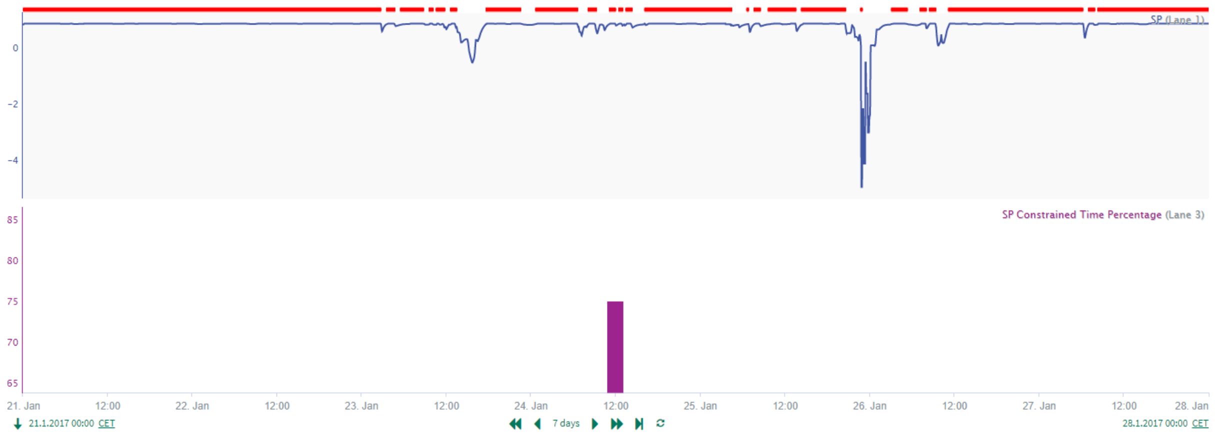


Figure 62: Setpoint of controller 90 with maximum constraint

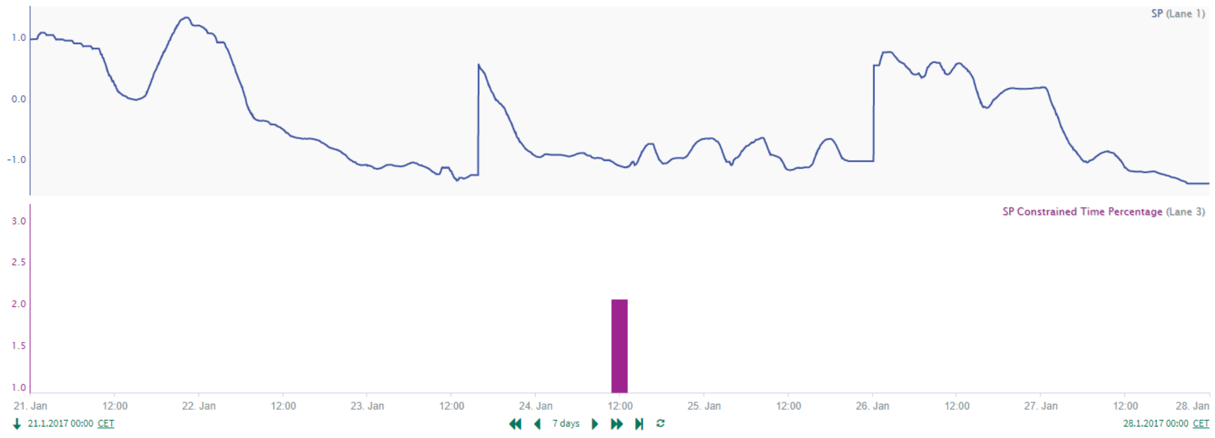


Figure 63: Setpoint of controller 60 without constraints

## 5.5 Add-on Evaluation

This Section will assess the extent to which the add-on meets the specifications and requirements of Section 5.1. For this purpose, Section 5.5.1 evaluates the fulfilment of the add-on input and output specifications. The Sections 5.5.2 to 5.5.4 discuss and evaluate the fulfilment of the user requirements, detection method requirements and requirements from Seeq Corporation. The final evaluation is given in Section 5.5.5.

### 5.5.1 Evaluation of Add-on Input and Output Specifications

First, the add-on inputs and outputs are evaluated using Table 11 which summarises the contents of Table 8 and Table 10.

*Table 11: Comparison of add-on input and output specifications and actual add-on inputs and outputs*

| Add-on input specifications from Table 7 | Add-on inputs from Table 9 |
|---|---|
| **Input asset tree** | **Input asset tree** |
| **Start and end time of analysis period** | **Start and end time of analysis period** |
| **Thresholds for visualisation in treemap** | **Thresholds for visualisation in treemap** |
| **Other input options for higher user-friendliness** | **Other input options for higher user-friendliness:**<br>▪ Selection of control signals to be analysed<br>▪ Short capsules and gaps (optional)<br>▪ Name for the output asset tree (optional) |
| **Add-on output specifications from Table 7** | **Add-on outputs from Table 9** |
| **Output asset tree with:**<br>▪ Input control signals | **Output asset tree with:**<br>▪ Input control signals |
| ▪ Constraint/saturation signal | ▪ Constraint/saturation signal |
| ▪ Conditions with time capsules for saturated/constrained periods | ▪ Conditions with time capsules for saturated/constrained periods:<br>“High Constraint/Saturation” condition<br>“Medium Constraint/Saturation” condition |
| ▪ Measure for the extent of saturation and constraints | ▪ Measure for the extent of saturation and constraints:<br>“Constrained Time %” or “Saturated Time %” |
| **Treemap coloured according to the extent of saturation/constraints** | **Treemap coloured according to the “Constrained/Saturated Time %”** |
| **Other outputs for higher user-friendliness** | **Other outputs for higher user-friendliness:**<br>▪ Table in the UI with a top 30 of the most constrained/saturated periods<br>▪ Option to export the table to a CSV file |

Regarding the add-on inputs, all specifications were fulfilled and three inputs for a higher user-friendliness were implemented which are the selection of control signals for the analysis, the specification of short gaps and capsules and the specification of a name for the output asset tree. Regarding the add-on outputs, the add-on creates an output asset tree with the input control signals, constraint/saturation signals, "High Constraint/Saturation" conditions, "Medium Constraint/Saturation" conditions and "Constrained/Saturated Time %". The "Constrained/Saturated Time %" is an easy-to-understand statistic for the extent of saturation and constraints. Based on the statistic, the "High Constraint/Saturation" condition and "Medium Constraint/Saturation" condition are generated which are responsible for the treemap colours and the time capsules in the time trend. The treemap in combination with the top 30 table in the UI gives the user the ability to get a plant-wide overview of constraints and saturation in control loops. For higher user-friendliness, the table can be exported to a CSV file. Furthermore, the add-on was tested successfully on all four asset trees from Section 5.2 to guarantee correct functionality with different asset tree structures. In summary, the add-on fulfils all input and output specifications, and additional inputs and outputs were implemented to enhance the user experience.

### 5.5.2 Evaluation of User Requirements

Correct, easy-to-interpret, meaningful results: The treemap along with the table in the UI, the "Constrained/Saturated Time %" and the time capsules in the time trend give the user the ability to conduct a plant-wide control loop performance assessment in terms of saturation and constraints. Especially the automatically generated treemap is an easy-to-use tool as it visualises the extent of constraints and saturation using different colours. The "Constrained/Saturated Time %" is easy to interpret because it measures the extent of constraints and saturation on a scale from 0 % to 100 %. To provide correct results the detection method with the highest robustness was selected in Section 3.3. However, as the false results could not be reduced to 0, 100 % correct results cannot be guaranteed. The treemap visualisation partially compensates for the false results, as they are likely to have a low "Constrained/Saturated Time %" and are therefore shown as green panels in the treemap which are of lower interest for the user as red and yellow panels. In summary, the constraint detection add-on provides the highest possible proportion of correct results and generates easy-to-interpret and meaningful outputs so that the user requirement is met to a good extent.

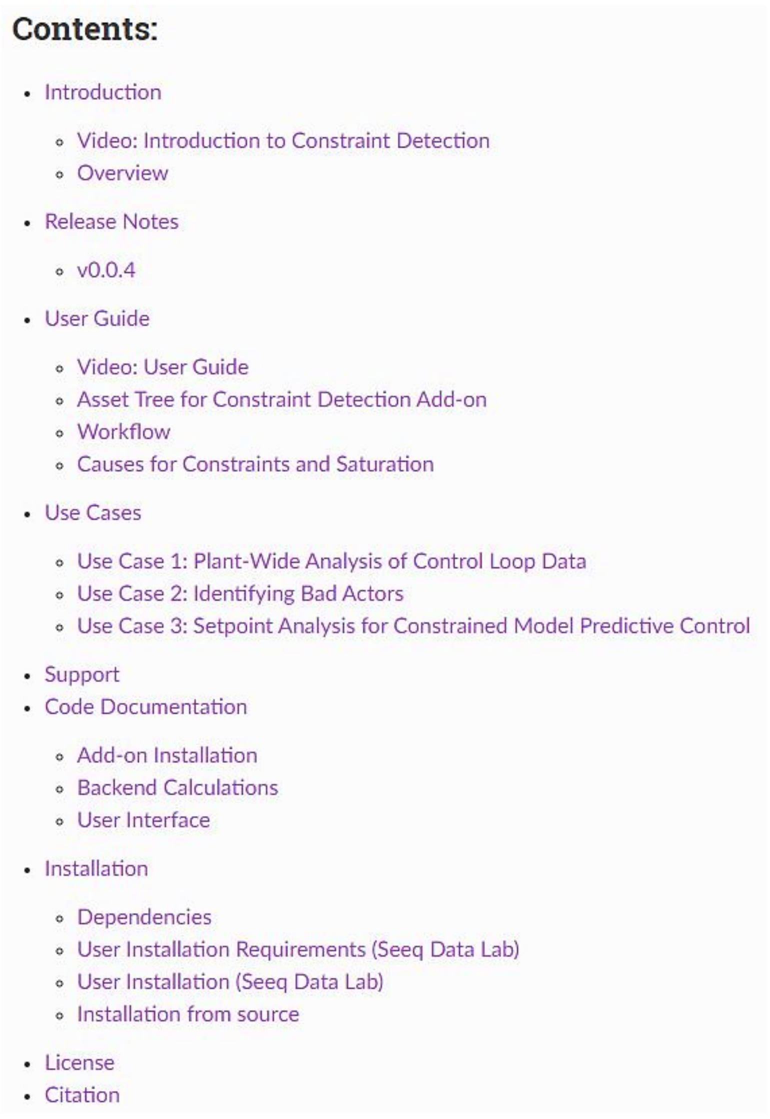
Easy workflow and intuitive UI: In the UI, the user can start from the top and work down to the bottom of the UI to specify all inputs. In every section of the UI, a short statement tells the user what needs to be done (e.g. "Select Signals for Analysis", "Select Asset Tree"). In addition, some input fields provide a hint which helps the user to fill in the UI correctly. Furthermore, the question mark icons with the dialog windows provide guidance when filling in the UI. The use of the Python library ipyvuetify makes the UI also visually appealing. The workflow is improved by disabling and enabling certain parts of the UI during the add-on execution, so that the user can only use the add-on sections that are intended to be used in the respective step in the workflow.

Nevertheless, the add-on has two drawbacks in its workflow. The first one is the required asset tree structure which forces the user to generate an asset tree with Seeq Data Lab before the add-on can be used. Since this is impractical for the user, template code for the use in Seeq Data Lab is made available on GitHub. The second drawback is related to a bug in the Seeq Python library SPy. The bug means that the add-on can only be executed by the owner of the respective workbook. The add-on cannot be used by users who access a shared workbook. Seeq Corporation is currently working on this bug, so that it can be fixed in the future.

In summary, the UI is intuitive and the workflow is easy once the asset tree in the required structure has been generated with the template code. Nevertheless, it is to be expected that the asset tree is the critical point of the add-on that determines whether the add-on is used. The bug in SPy is considered less important as it will be solved in the future. As a result, this user requirement is fulfilled to a good extent.

<u>Short processing time:</u> The processing time depends on the number of signals in the input asset tree and on the length of the analysis period. The processing time for the example workflow in Section 5.3.1 is less than a minute whereas the processing time for all three use cases in Section 5.4 is around 15 minutes. The processing steps that take the most time are importing and exporting all signals and conditions. These steps are executed by functions from the Seeq Python library SPy called spy.pull and spy.push. Since the speed of these functions cannot be influenced, the long processing time of the add-on had to be accepted in order to guarantee all other functions. The progress bar was implemented to compensate for the long processing time and to communicate to the user that the add-on is still in progress. In summary, the processing time depends on the use case and no final statement about the fulfilment of the requirement can be made.

<u>Documentation with comprehensible user guide, installation instructions and use cases:</u> The contents of the publicly available documentation are shown in Figure 64.


**Contents:**

- Introduction
  - Video: Introduction to Constraint Detection
  - Overview
- Release Notes
  - v0.0.4
- User Guide
  - Video: User Guide
  - Asset Tree for Constraint Detection Add-on
  - Workflow
  - Causes for Constraints and Saturation
- Use Cases
  - Use Case 1: Plant-Wide Analysis of Control Loop Data
  - Use Case 2: Identifying Bad Actors
  - Use Case 3: Setpoint Analysis for Constrained Model Predictive Control
- Support
- Code Documentation
  - Add-on Installation
  - Backend Calculations
  - User Interface
- Installation
  - Dependencies
  - User Installation Requirements (Seeq Data Lab)
  - User Installation (Seeq Data Lab)
  - Installation from source
- License
- Citation

*Figure 64: Contents of the add-on documentation*

The documentation provides a user guide which explains the required asset tree structure, the workflow and possible causes for constraints and saturation in control loops. The user guide also includes a video with a step-by-step explanation of the add-on use. The use cases from Section 5.4 are available in the documentation. Installation instructions to install the add-on via Seeq Data Lab are given as well. Additionally, the documentation provides an introduction with an introductory video to the constraint detection add-on, release notes for the latest add-on version and a support section. The documentation contains all important components and is detailed and well structured so that it meets all user requirements.

Simple way to get support: As already described in Section 0, the UI provides multiple links to GitHub and to the documentation where the user can get further information or create a GitHub issue to report an error or to request support. As a result, this user requirement can be seen as completely fulfilled.

### 5.5.3 Evaluation of Detection Method Requirements

In Section 3.3, the modified method by Bauer et al. was selected as the detection method to be implemented in the constraint detection add-on. The method is easy to understand, has a low computation time and uses constant hard-coded parameters. In terms of robustness, the implemented method had the highest number of correct results where 47 out of 54 signals were analysed correctly. Consequently, the detection method meets the requirements to a good extent.

### 5.5.4 Evaluation of Requirements from Seeq Corporation

Maintainable code that follows coding standards: For better maintainability, the code was divided into multiple Python modules which are listed below.

- _seeq_add_on.py: contains the UI
- _SPy_functions.py: contains all functions that use the Seeq Python library SPy
- _saturation_detection.py: contains the detection method and related functions

Furthermore, multiple Python modules are responsible for the correct installation in Seeq. The modular structure simplifies working with the code, as only individual modules need to be revised when making changes to the add-on. In addition, all functions and classes provide a docstring. Docstrings are used to give short explanations of the code and to specify the input arguments and return objects in functions and classes. The docstrings were also used to automatically generate the code documentation. Furthermore, understandable names were chosen for all functions, classes, and objects. In the Python code, comments are used to explain separate work steps and to increase the code comprehensibility. With the PyCharm development environment, the standard code style for Python was followed, which increases the readability of the code. In summary, the code is structured clearly and is explained using docstrings, comments and comprehensible naming. As a result, the requirement is completely fulfilled.

Code Documentation: In Figure 64, the code documentation is listed as a section in the add-on documentation. The code documentation is divided into three parts which are the add-on installation, backend calculations and user interface. These subsections provide explanations, input arguments, and return objects for all functions and classes in the add-on. Links to the source code are given as well so that the reader can switch easily between the code documentation and the source code. The code documentation is clearly structured and comprehensive so that the requirement is completely fulfilled.

Publication of the add-on on GitHub: The Python code is published on GitHub along with the template code for the asset tree and a README file with an overview of the add-on. The “Constraint-Detection” repository is listed in the repositories of the GitHub organisation “HAW Process Automation”. Consequently, the requirement can be considered as completely fulfilled.

Upload of the add-on as a Python package on PyPI: The latest version of the constraint detection add-on was uploaded as a Python package named “seeq-constraintdetection” on PyPI and can be installed using the terminal in Seeq Data Lab (Tiedemann 2022b). The requirement is completely fulfilled.

### 5.5.5 Summary of Add-on Evaluation

The evaluation of the add-on requirements which was discussed in Section 5.5.2 to 5.5.4 is summarised in Table 12. The table shows that 9 out of 13 requirements were completely fulfilled. The “correct, easy-to-interpret, and meaningful results” and “robustness” requirements were only fulfilled to a good extent because the detection method does not guarantee 100 % correct results. The “easy workflow and intuitive UI” requirement was only fulfilled to a good extent due to the required asset tree structure. The add-on processing time was not evaluated since it strongly depends on the use case.

From Section 5.5.1, it was determined that the add-on is fully functional as it meets all input and output specifications and provides additional features which increase the user-friendliness. In summary, the constraint detection add-on was successfully developed and implemented and can be described as a helpful tool for CPM which fulfils most requirements to a complete extent.

In the future, the add-on could be improved by providing an additional tool that actively helps the user with the generation of the required asset tree structure. Furthermore, additional features for a more flexible use could be implemented. For example, future features could include an option to switch between “single signal analysis” and “asset tree analysis” or a feature to filter out signals of low interest.

*Table 12: Evaluation of add-on requirements*

| Requirement group | Requirement | Evaluation |
|---|---|---|
| **User requirements** | Correct, easy-to-interpret, meaningful results | Fulfilled to a good extent |
| | Easy workflow and intuitive UI | Fulfilled to a good extent |
| | Short processing time | - |
| | Documentation with comprehensible user guide, installation instructions and use cases | Completely fulfilled |
| | Simple way to get support | Completely fulfilled |
| **Detection method requirements** | Robustness | Fulfilled to a good extent |
| | Understandability | Completely fulfilled |
| | Constant hard-coded parameters | Completely fulfilled |
| | Low computation time | Completely fulfilled |
| **Requirements from Seeq Corporation** | Maintainable code that follows coding standards | Completely fulfilled |
| | Code documentation | Completely fulfilled |
| | Publication of the add-on on GitHub | Completely fulfilled |
| | Upload of the add-on as a Python package on PyPI | Completely fulfilled |

# 6. Summary

The main objective of this thesis was the development of an add-on for the detection of constraints and saturation in control loop data and the implementation of the add-on in the data analytics software Seeq. For this purpose, a detection method had to be selected by means of a literature research and comparison of different detection methods. In addition, the thesis aimed to establish a clear distinction between constraints and saturation in control loops. In this Chapter, the thesis results will be summarised, and it will be reviewed whether the aim of the thesis could be achieved.

Constraints are defined as limits which are inherent to a control loop component or signal. The sensor and the actuator of a control loop are physically limited due to the sensor's measuring range and the actuator range. Therefore, the measured process variable and the manipulated variable have a minimum and a maximum value and can become constrained. The setpoint can be limited if certain control strategies are applied such as constrained MPC or constrained cascade control. In this case, the setpoint has a minimum and maximum, and can reach a constraint. The controller itself has no inherent limits but the controller output is still limited because the actuator has constraints. For this reason, saturation was defined as controller saturation and occurs when the actuator reaches a constraint and consequently causes the controller to become saturated. In the context of the thesis, it is important to emphasise that saturation and constraints are not necessarily a process fault but can be a deliberate way of plant operation. In process data, constraints and saturation can be recognized by the fact that the signal has a flat time trend and sticks to its minimum or maximum and only deviates from there for short time periods.

Three detection methods for controller saturation were investigated. The method by Matsuo et al. uses the Kolmogorov-Smirnov Two-Sample test to measure the statistical similarity between a saturated test signal and the actual controller output. The method by Bauer et al. detects saturation by analysing if multiple consecutive samples lie within a saturation band close to the minimum or maximum of a signal. A modified version of Bauer's method was developed which adds the condition that the numerical derivative of the signal must be zero for saturation to be detected. The method by Matsuo et al., the method by Bauer et al. and the modified method by Bauer et al. were compared in terms of robustness, understandability, computation time and hard-coded parameters using publicly available industrial data. The modified version of Bauer et al. was selected for the implementation in the add-on due to its high robustness, easy algorithm, and low computation time.

For the development of the constraint detection add-on, a set of specifications and requirements was defined that the add-on should fulfil. Based on the specifications and requirements, an add-on was developed that imports an asset tree and analyses the selected control signals in a user-specified time period. As a result, the add-on generates treemaps for the selected control signals, a top 30 table of the signals with the highest "Constrained/Saturated Time %" and an output asset tree. To test the add-on, four asset trees with different structures were generated from industrial data and the add-on successfully analysed all of them. The add-on is fully functional and user-friendly except for the significant limitation of the required asset tree structure. Furthermore, the add-on is fully and extensively documented, and a GitHub repository has been created. The constraint detection add-on is freely available on PyPI as a Python package which is named "seeq-constraintdetection" and can be installed in Seeq. In summary, the add-on meets all input and output specifications and fulfils most of the requirements to full extent, so that the development and implementation of the constraint detection add-on can be assessed as successful.

## References

Ackerman, Pascal (2017): Industrial cybersecurity. Efficiently secure critical infrastructure systems. Birmingham, UK: Packt Publishing.

Akavalappil, Vijoy; Radhakrishnan, T. K. (2022): Comparison of current state of control valve stiction detection and quantification techniques. In *Transactions of the Institute of Measurement and Control* 44 (3), pp. 562–579. DOI: 10.1177/01423312211038288.

Bacci D. Capaci, Riccardo; Scali, Claudio (2018): Review and comparison of techniques of analysis of valve stiction: From modeling to smart diagnosis. In *Chemical Engineering Research and Design* 130, pp. 230–265. DOI: 10.1016/j.cherd.2017.12.038.

Bauer, Margret; Auret, Lidia; Di Bacci Capaci, Riccardo; Horch, Alexander; Thornhill, Nina F. (2019): Industrial PID Control Loop Data Repository and Comparison of Fault Detection Methods. In *Ind. Eng. Chem. Res.* 58 (26), pp. 11430–11439. DOI: 10.1021/acs.iecr.8b06354.

Bauer, Margret; Horch, Alexander; Xie, Lei; Jelali, Mohieddine; Thornhill, Nina F. (2016): The current state of control loop performance monitoring – A survey of application in industry. In *Journal of Process Control* 38 (3), pp. 1–10. DOI: 10.1016/j.jprocont.2015.11.002.

Bauer, Margret; Madolo, Sethabile (2007): Detection and Effect of Quantisation in Data-Driven Process Analysis. In *IFAC Proceedings Volumes* 40 (5), pp. 75–80. DOI: 10.3182/20070606-3-MX-2915.00011.

Bernstein, Dennis S.; Michel, Anthony N. (1995): A Chronological Bibliography on Saturating Actuators. In *International Journal of Robust and Nonlinear Control* 5, pp. 375–380.

Biswas, Pinak Pani; Ray, Subhabrata; Samanta, Amar Nath (2009): Nonlinear control of high purity distillation column under input saturation and parametric uncertainty. In *Journal of Process Control* 19 (1), pp. 75–84. DOI: 10.1016/j.jprocont.2008.01.007.

Choudhury, M.A.A.S.; Shah, Sirish L.; Thornhill, Nina F. (2004): Diagnosis of poor control-loop performance using higher-order statistics. In *Automatica* 40 (10), pp. 1719–1728. DOI: 10.1016/j.automatica.2004.03.022.

Choudhury, M.A.A.S.; Thornhill, Nina F.; Shah, S. L. (2005): Modelling valve stiction. In *Control Engineering Practice* 13 (5), pp. 641–658. DOI: 10.1016/j.conengprac.2004.05.005.

Cox, John (2021): Advanced Analytics for Process Control Engineers. Available online at https://seeq.com/resources/blog/advanced-analytics-for-process-control-engineers, updated on 13.10.2021, checked on 13.07.2022.

Dambros, Jônathan W.V.; Farenzena, Marcelo; Trierweiler, Jorge O. (2019): Oscillation Detection and Diagnosis in Process Industries by Pattern Recognition Technique. In *IFAC-PapersOnLine* 52 (1), pp. 299–304. DOI: 10.1016/j.ifacol.2019.06.078.

Duller, Christine (2008): Einführung in die nichtparametrische Statistik mit SAS und R. Ein anwendungsorientiertes Lehr- und Arbeitsbuch. Heidelberg: Physica-Verlag HD.

Essinger, Timothy: Stiction Analyser User Guide. Available online at https://haw-process-automation.github.io/Stiction-Analyser/user-guide, checked on 05.07.2022.

Forsman, Krister; Stattin, Andreas (1999): A New Criterion for Detecting Oscillations in Control Loops. In *European Control Conference*, pp. 2313–2316.

Galloway, Brendan; Hancke, Gerhard P. (2013): Introduction to Industrial Control Networks. In *IEEE Commun. Surv. Tutorials* 15 (2), pp. 860–880. DOI: 10.1109/SURV.2012.071812.00124.

Hägglund, Tore (1995): A Control-Loop Performance Monitor. In *Control Engineering Practice* 3 (11), pp. 1543–1551.

Hägglund, Tore (1999): Automatic detection of sluggish control loops. In *Control Engineering Practice* 7, pp. 1505–1511.

Hägglund, Tore (2019): Praktisk processreglering. 4. Edition: Studentlitteratur AB.

Harris, Thomas J. (1989): Assessment of control loop performance. In *The Canadian Journal of Chemical Engineering* 67 (5), pp. 856–861. DOI: 10.1002/cjce.5450670519.

HAW Process Automation (Ed.): Constraint Detection GitHub Issues. Available online at https://github.com/HAW-Process-Automation/Constraint-Detection/issues, checked on 12.10.2022.

He, Q. Peter; Wang, Jin; Pottmann, Martin; Qin, S. Joe (2007): A Curve Fitting Method for Detecting Valve Stiction in Oscillating Control Loops. In *Industrial and Engineering Chemistry Research* 46 (13), pp. 4549–4560. DOI: 10.1021/ie061219a.

Heister, Timo; Rebholz, Leo G.; Xue, Fei (2019): Numerical analysis. An introduction. Berlin, Boston: De Gruyter (De Gruyter Textbook).

Horch, Alexander (1999): A simple method for detection of stiction in control valves. In *Control Engineering Practice* 7 (10), pp. 1221–1231. DOI: 10.1016/S0967-0661(99)00100-8.

Horch, Alexander; Kühl, Peter (2003): Detection of sluggish control loops - Experiences and improvements. In *IFAC Fault Detection, Supervision and Safety of Technical Processes*, pp. 99–104.

Hu, Yan; Yang, An; Li, Hong; Sun, Yuyan; Sun, Limin (2018): A survey of intrusion detection on industrial control systems. In *International Journal of Distributed Sensor Networks* 14 (8), 1-14. DOI: 10.1177/1550147718794615.

Jämsä-Jounela, S-L.; Poikonen, R.; Vatanski, N.; Rantala, A. (2003): Evaluation of Control Performance in Remote Maintenance Centers.

Jelali, Mohieddine (2006): An overview of control performance assessment technology and industrial applications. In *Control Engineering Practice* 14 (5), pp. 441–466. DOI: 10.1016/j.conengprac.2005.11.005.

Jelali, Mohieddine (2013): Control Performance Management in Industrial Automation. Assessment, Diagnosis and Improvement of Control Loop Performance. London: Springer-Verlag.

Kano, Manabu; Maruta, Hiroshi; Kugemoto, Hidekazu; Shimizu, Keiko (2004): Practical Model and Detection Algorithm for Valve Stiction. In *IFAC Proceedings Volumes* 37 (9), pp. 859–864. DOI: 10.1016/S1474-6670(17)31917-1.

Kreisselmeier, Gerhard (1996): Stabilization of Linear Systems in the Presence of Output Measurement Saturation. In *Systems & Control Letters* 29, pp. 27–30.

Massey, Frank J. (1951): The Kolmogorov-Smirnov Test for Goodness of Fit. In *Journal of the American Statistical Association* 46 (253), pp. 68–78.

Matsuo, Toru; Tadakuma, Isao; Thornhill, Nina F. (2004): Diagnosis of a Unit-Wide Disturbance Caused by Saturation in Manipulated Variable. In *IEEE Advanced Process Control Applications for Industry Workshop*, pp. 1–9.

Miao, Tina; Seborg, Dale E. (1999): Automatic Detection of Excessively Oscillatory Feedback Control Loops. In *Proceedings of the 1999 IEEE International Conference on Control Applications*, pp. 359–364.

Risse, Michael (2019): The data historian’s history told. Edited by Control Engineering. Available online at https://www.controleng.com/articles/the-data-historians-history-told/, updated on 27.04.2019, checked on 15.06.2022.

Rossi, M.; Scali, Claudio (2005): A comparison of techniques for automatic detection of stiction: simulation and application to industrial data. In *Journal of Process Control* 15 (5), pp. 505–514. DOI: 10.1016/j.jprocont.2004.11.003.

Saberi, Ali; Stoorvogel, Anton; Sannuti, Peddapullaiah (2000): Control of Linear Systems with Regulation and Input Constraints. London: Springer London.

Scali, Claudio; Ghelardoni, Claudio (2008): An improved qualitative shape analysis technique for automatic detection of valve stiction in flow control loops. In *Control Engineering Practice* 16 (12), pp. 1501–1508. DOI: 10.1016/j.conengprac.2008.04.009.

Seborg, D. E.; Edgar, T. F.; Mellichamp, D. A. (2004): Process dynamics and control. 2. Edition. Hoboken, NJ: Wiley.

Seborg, Dale E.; Edgar, Thomas F.; Mellichamp, Duncan A. (1989): Process dynamics and control. New York: Wiley (Wiley series in chemical engineering).

Seeq Corporation (Ed.): Control Loop Performance Monitoring (CLPM), checked on 13.07.2022.

Seeq Corporation (Ed.): Seeq Architecture. Available online at https://seeq.com/product/seeq-architecture, checked on 11.07.2022.

Seeq Corporation (Ed.): Seeq Data Lab. Available online at https://seeq.com/product/seeq-data-lab, checked on 11.07.2022.

Seeq Corporation (Ed.): Seeq Organizer. Available online at https://seeq.com/product/organizer, checked on 11.07.2022.

Seeq Corporation (Ed.): Seeq Workbench. Available online at https://seeq.com/product/workbench, checked on 11.07.2022.

Seeq Corporation (Ed.): Welcome to Seeq Add-on Gallery. Available online at https://seeq12.github.io/gallery/, checked on 11.07.2022.

South African Council for Automation and Control (Ed.): Resources. Available online at https://sacac.org.za/resources/, checked on 13.06.2022.

Srinivasan, Ranganathan; Rengaswamy, Raghunathan; Miller, Randy (2005): Control Loop Performance Assessment. 1. A Qualitative Approach for Stiction Diagnosis. In *Industrial and Engineering Chemistry Research* 44 (17), pp. 6708–6718. DOI: 10.1021/ie0490280.

Srinivasan, Ranganathan; Rengaswamy, Raghunathan; Miller, Randy (2007): A modified empirical mode decomposition (EMD) process for oscillation characterization in control loops. In *Control Engineering Practice* 15 (9), pp. 1135–1148. DOI: 10.1016/j.conengprac.2007.01.014.

Statext (Ed.): Kolmogorov-Smirnov Test. Available online at https://www.statext.com/practice/KolmogorovSmirnovT02.php, checked on 17.05.2022.

Tarbouriech, Sophie; Turner, Matthew (2009): Anti-windup Design: An Overview of Some Recent Advances and Open Problems. In *IET Control Theory & Applications* 3 (1), pp. 1–19. DOI: 10.1049/iet-cta:20070435.

Thornhill, Nina F. (2005): Finding the source of nonlinearity in a process with plant-wide oscillation. In *IEEE Transactions on Control Systems Technology* 13 (3), pp. 434–443. DOI: 10.1109/TCST.2004.839570.

Thornhill, Nina F.; Hägglund, Tore (1997): Detection and Diagnosis of Oscillation in Control Loops. In *Control Engineering Practice* 5 (10), pp. 1343–1354.

Thornhill, Nina F.; Huang, B.; Zhang, H. (2003): Detection of Multiple Oscillations in Control Loops. In *Journal of Process Control* 13, pp. 91–100.

Tiedemann, Lea (2022a): Asset Tree for Constraint Detection Add-on. Available online at https://constraint-detection.readthedocs.io/en/latest/userguide.html#asset-tree-for-constraint-detection-add-on, updated on 05.10.2022, checked on 12.10.2022.

Tiedemann, Lea (2022b): seeq-constraintdetection. Available online at https://pypi.org/project/seeq-constraintdetection/, checked on 07.11.2022.

Tiedemann, Lea (2022c): Template Code for Asset Tree. Edited by HAW Process Automation. Available online at https://github.com/HAW-Process-Automation/Constraint-Detection/tree/main/Template%20Code%20for%20Asset%20Tree, updated on 28.09.2022, checked on 12.10.2022.

Tiedemann, Lea (2022d): Use Cases. Available online at https://constraint-detection.readthedocs.io/en/latest/usecases.html, updated on 05.10.2022, checked on 12.10.2022.

Tiedemann, Lea (2022e): Video: User Guide. Available online at https://constraint-detection.readthedocs.io/en/latest/userguide.html#video-user-guide, updated on 05.10.2022, checked on 12.10.2022.

Tiedemann, Lea (2022f): Welcome to seeq-constraintdetection's documentation! Available online at https://constraint-detection.readthedocs.io/en/latest/index.html, updated on 05.10.2022, checked on 12.10.2022.

Visioli, Antonio (2006): Practical PID Control. London: Springer London.

Wade, Harold L. (2004): Basic and advanced regulatory control. System design and application. 2. Edition. Research Triangle Park, NC: ISA-The Instrumentation Systems and Automation Society.

Yamashita, Yoshiyuki (2006): An automatic method for detection of valve stiction in process control loops. In *Control Engineering Practice* 14 (5), pp. 503–510. DOI: 10.1016/j.conengprac.2005.03.004.

Zaiontz, Charles: Two-Sample Kolmogorov-Smirnov Test. Edited by Real Statistics Using Excel. Available online at https://www.real-statistics.com/non-parametric-tests/goodness-of-fit-tests/two-sample-kolmogorov-smirnov-test/, checked on 03.06.2022.

Zakharov, Alexey; Jämsä-Jounela, Sirkka-Liisa (2014): Robust Oscillation Detection Index and Characterization of Oscillating Signals for Valve Stiction Detection. In *Industrial and Engineering Chemical Ressearch* 53 (14), pp. 5973–5981. DOI: 10.1021/ie402636m.