

Marius Lauvland

# Condition monitoring of a generator cooling system in a hydropower plant

Bachelor's project in Renewable Energy Engineering

Supervisor: Felix Kelberlau

July 2019

**NTNU**  
Norwegian University of Science and Technology  
Faculty of Engineering  
Department of Energy and Process Engineering





## Bachelor's thesis

<b>Thesis title</b> Condition monitoring of a generator cooling system in a hydropower plant <b>Norwegian title</b> Tilstandsovervåking av eit generatorkjølesystem i eit vasskraftverk	<b>Start date</b> 01.04.2019
	<b>Date of submission</b> 10.07.2019
	<b>Number of pages report / appendix</b> 65 / 28
<b>Student</b> Marius Lauvland	<b>Supervisor</b> Felix Kelberlau PhD Candidate, NTNU felix.kelberlau@ntnu.no
<b>Course of study</b> Renewable Energy Engineering	<b>Project number</b> FEN1914
<b>Assignment owner</b> Voith Hydro AS	<b>Representative and external supervisor</b> Øyvind Holm oeyvind.holm@voith.com

Free publication Delayed publication Free publication after



## Preface

This thesis marks the conclusion of a bachelor's degree in Renewable Energy Engineering at the Department of Energy and Process Engineering at the Norwegian University of Science and Technology(NTNU) in Trondheim.

The scope of the assignment has been compiled in a cooperation between Øyvind Holm at Voith Hydro, Felix Kelberlau at NTNU and me.

The help I have received throughout the project has been indispensable. There are many to be thanked for their help and support, but I would like to particularly mention Øyvind Holm at Voith Hydro, who has willingly shared his hydropower and engineering insight. I have gained much knowledge thanks to his outstanding pedagogical skills.

Next, I would like to thank Felix Kelberlau who has been available ever since September, and who even spent his holiday supervising me. His advice is always analytic and well-founded.

This project would not have been possible to complete without help from others, thanks are extended to Geir Småøien, Martin Fald Gaarden, Trond Sliper, Jostein Kjærnes Fossum and Felix Lippold at Voith for helping with everything, big and small, and for conducting the visit to Brattset hydropower plant. Furthermore, NTNU has shown great flexibility by allowing the work to continue throughout the summer holiday, so that it was possible to go on exchange and finish the studies on time. This was more than one could expect, and Håvard Karoliussen, Tor Hennem, and, once again, Felix Kelberlau, must be thanked for making this process easy. Oddvar Bjerkås at TrønderEnergi is thanked for his great tour of Brattset hydropower plant and for providing insight into the maintenance routines of the hydropower plant. Greatly appreciated is also advice and guidance from Thomas Welte at SINTEF and Viggo Pedersen at NTNU. Gratitude is extended to a special someone, who is thanked for her proofreading and valuable support.

This work has truly been rewarding and educational, and I have been looking forward to every single day.

Trondheim, July 10<sup>th</sup> 2019



Marius Lauvland



## Abstract

This project thesis has investigated the potential for condition monitoring of a generator cooling system in a hydropower plant, using physical models on data from a SCADA system. The work has been supervised by Voith Hydro AS.

SCADA data from Brattset hydropower plant, made available by Voith Hydro, was used throughout the work with the project. The scope of the project focused in the generator cooling system, with a particular focus on the generator air coolers. Several approaches to condition monitoring have been investigated during the project. The data that was used were mostly analogue air temperature measurements, but measurement of power output were also used. The temperatures for the cold air of the coolers and the measurements of the warm air from the generator, were used to determine the condition of the coolers. The difference in the air temperature over the air coolers were believed to be influenced by the condition of the coolers. The coolers of both generators were investigated individually and collectively.

Several analyses were conducted, using Python and associated libraries, to develop three condition monitoring concepts. One concept monitors the cold air of the air coolers individually, and may function as an early warning system following a leakage of the water pipes in the cooler. The second concept may monitor the cooling effect of each cooler individually or all the coolers belonging to one generator at the same time by monitoring the difference in air temperature over time. The last concept predicts the temperature given a specific power output of the generator.

The developed concepts show that SCADA data has the necessary sampling frequency and accuracy to be used for condition monitoring. The concepts also show that the air coolers in a hydropower plant may be suited for condition monitoring provided that you have enough data.





# Contents

<b>Preface</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Scope . . . . .	2
1.3 Limitations . . . . .	2
1.4 Structure of the report . . . . .	2
<b>2 Background theory</b>	<b>5</b>
2.1 Hydropower plants . . . . .	5
2.1.1 Turbines . . . . .	6
2.1.2 Generators . . . . .	6
2.1.3 Cooling systems . . . . .	8
2.1.4 SCADA systems . . . . .	10
2.1.5 Brattset hydropower plant . . . . .	12
2.2 Maintenance . . . . .	15
2.2.1 Preventive maintenance . . . . .	15
2.2.2 Condition Monitoring . . . . .	15
2.2.3 Hydropower plant maintenance . . . . .	16
2.2.4 Maintenance of Brattset hydropower plant . . . . .	16
<b>3 Method and approach</b>	<b>19</b>
3.1 Project progress . . . . .	19
3.2 Data export . . . . .	19
3.3 Pre-processing . . . . .	20
3.3.1 Proof of Concept . . . . .	22
3.3.2 Change in air temperature over the coolers . . . . .	23
3.3.3 Quantitative vs qualitative approach . . . . .	23
3.4 Time span . . . . .	25
3.5 Analysis . . . . .	26
3.5.1 Scenario 1 . . . . .	27
3.5.2 Scenario 2 . . . . .	29
<b>4 Results</b>	<b>31</b>
4.1 Proof of Concept . . . . .	31
4.2 Scenario 1 . . . . .	32
4.3 Scenario 2 . . . . .	40
<b>5 Discussion</b>	<b>43</b>
5.1 Pre-processing . . . . .	43
5.2 Data . . . . .	43
5.3 Uncertainties . . . . .	44
5.4 Scripting . . . . .	44
5.5 Methods and tools . . . . .	44
5.6 Proof of concept . . . . .	44
5.7 Scenario 1 . . . . .	45

5.8 Scenario 2 . . . . .	46
5.9 Is it worth it? . . . . .	46
5.10 Why condition monitoring . . . . .	46
<b>6 Conclusion and further work</b>	<b>49</b>
6.1 Conclusion . . . . .	49
6.2 Further work . . . . .	49
<b>References</b>	<b>51</b>
<b>Appendix A Piping and instrumentation diagrams</b>	<b>I</b>
<b>Appendix B Script for the Proof of Concept plot</b>	<b>II</b>
<b>Appendix C Scenario 1 script</b>	<b>XII</b>
<b>Appendix D Scenario 2 script</b>	<b>XXI</b>

## List of Figures

2.1	Cross section of Brattset hydropower plant. Received from TrønderEnergi .	5
2.2	Spiral casing with stay vanes, wicket gates and runner blades, modified from [16, p. 838] . . . . .	6
2.3	Toothed stator segment and salient rotor pole [16, p. 356] . . . . .	7
2.4	Generator air cooling loop, received from Voith Hydro . . . . .	8
2.5	Generator air cooler at Brattset hydropower plant, received from TrønderEnergi	9
2.6	Operation, warning and alarm zones for a measurement in a SCADA system, inspired by [37] . . . . .	12
2.7	Piping and instrumentation diagram for Brattset's cooling system, received from TrønderEnergi . . . . .	13
2.8	Maintenance strategy hierarchy, inspired by [46] . . . . .	16
3.1	The progress towards Condition Monitoring, inspired by [34, p. 6] . . . . .	20
3.2	Screenshot from Voith's Analyzer . . . . .	21
3.3	Screenshot from the analyzer's export function . . . . .	21
3.4	Minute and second sampling over 30 minute periode for power output . . .	22
3.5	Comparison of different sampling techniques . . . . .	23
3.6	Proof of Concept using cold air measurements from turbine 1, legend in plot 1 and 2 apply to plot 3 . . . . .	24
3.7	Effect of cooler illustrated as the change in temperature . . . . .	24
3.8	Sample frequencies with plots for different time spans over a 30 minute time period for power measurements on generator 1 on the 22 <sup>nd</sup> of June 2019 . .	26
3.9	Difference in temperature over cooler 1 on the 25 <sup>th</sup> of March 2017 . . . . .	28
3.10	Difference in temperature over cooler 1 on the 11 <sup>th</sup> of December 2017 . . .	28
3.11	Difference in temperature over cooler 1 on the 8 <sup>th</sup> of June 2019 . . . . .	29
4.1	Proof of Concept for each individual cooler for generator 1 . . . . .	31
4.2	Difference in temperature over air cooler 1 for generator 1, linear regression, and color according to the output power . . . . .	33
4.3	Difference in temperature over air cooler 2 for generator 1 and linear regression, and color according to the output power . . . . .	34
4.4	Difference in temperature over air cooler 3 for generator 1 and linear regression, and color according to the output power . . . . .	34
4.5	Difference in temperature over air cooler 4 for generator 1 and linear regression, and color according to the output power . . . . .	35
4.6	Difference in temperature over the air coolers for generator 1, with the mean temperature of the coolers, linear regression and setpoint of the power . . .	35
4.7	Difference in temperature over the air coolers for generator 1, without the two extreme values, with the mean temperature of the coolers, linear regression and setpoint of the power . . . . .	36
4.8	Difference in temperature over air cooler 1 for generator 2, linear regression, and color according to the output power . . . . .	37
4.9	Difference in temperature over air cooler 2 for generator 2, linear regression, and color according to the output power . . . . .	38

4.10	Difference in temperature over air cooler 3 for generator 2, linear regression, and color according to the output power . . . . .	38
4.11	Difference in temperature over air cooler 4 for generator 2, linear regression, and color according to the output power . . . . .	39
4.12	Difference in temperature over the air coolers for generator 2, with the mean temperature of the coolers, linear regression and setpoint of the power . . .	39
4.13	Difference in temperature over the air coolers for generator 2, without the two extreme values, with the mean temperature of the coolers, linear regression and setpoint of the power . . . . .	40
4.14	Temperature differences at various power outputs for generator 1 . . . . .	41
4.15	Temperature differences at various power outputs for generator 2 . . . . .	41

## List of Tables

3.1	Inexplicable power output values . . . . .	25
3.2	Returned values for different time spans over a 30 minute time period for power output on generator 1 on the 22 <sup>nd</sup> of June 2019 . . . . .	26
3.3	Time series exported from Analyzer that fulfill the requirements set for analysis . . . . .	27

# 1 Introduction

This chapter will introduce the background and motivation of the thesis. The scope and the problem formulation are given, and the limitations will be accounted for, before the structure of the report is described.

## 1.1 Background

Data and their applications are becoming increasingly accessible and profitable. The world's most valuable companies either have data as their business concept or actively use it in the development of their business[1]. We have entered the data age, or the Fourth Industrial Revolution, as some call it[2, 3].

Surrounded by services generated by data or data-generating services, concepts and expressions like Cyber Physical Systems, Digital Twins, Internet of Things and Internet of Services have become customary. The application of new technology lead to advanced services and systems, where some are almost as advanced as a human beings. In some cases human-like behavior is even being targeted. Just imagine the benefits of an automated system that can think for itself, or the potential savings related to a system that never breaks down, because it notifies the operator if something is on the verge of going wrong.[4]

In 2017 Norway's hydropower plants accounted for 95,8% of the country's produced electricity[5]. The hydropower plants are undoubtedly the country's most important source of electricity, which at times even provide an electricity surplus. As Norway is gradually becoming more integrated in the European power market – through the construction of several oversea power cables – the access to a market that largely consists of power from fossil fuels or inflexible, intermittent sources, can bring social-economic growth by offering flexible, renewable hydropower.[6, 7]

The downside of exploiting this opportunity of income, is that most hydropower plants are not designed for this modern usage pattern with rapid load regulation, called hydro peaking. Although they may handle it in the short term, it could incur additional maintenance and repair costs in the long run. Only some of the effects and the additional wear it causes are known, like increased sediments in the turbine, accelerated aging and problems with the cooling, others are yet to be discovered[8, 9].[7, 10]

To get a better understanding of the strain a modern usage pattern causes, and possibly avoid costly downtime, one can use the tools and analysis methods made available through the Fourth Industrial Revolution. These tools make it possible to continuously monitor the health of the hydropower plant by gathering data from key components. Of course, this does not only apply to hydropower plants used for hydro peaking, but all hydropower plants. Thus, money can be saved by increasing the lifetime of the production equipment, scheduling its maintenance and avoiding downtime. It can also supply value which is not as easy to quantify, like information on the condition and health of a specific component or the entire hydropower plant.

### 1.2 Scope

The scope and specification of the assignment was developed in a cooperation between Voith Hydro, NTNU, and the student. Voith Hydro, who is the assignment owner, wanted some sort of analysis of the cooling system in a hydropower plant using physical models. They wanted the products or results to be easy to reuse in another setting, which implies that no – or at least as little as possible – power plant specific details should be included. The products or results should also be robust enough to ensure that random errors do not have significant effect. Beyond those points, there were no strict guidelines.

The work in the initial phase of the project was therefore concentrated on finding a good problem formulation that potentially could bring some benefit to Voith Hydro. In order to achieve this, the behaviour of the cooling system and all its components was studied in detail and relevant literature was reviewed. Subsequently, the list of signals, containing almost 1200 signals, was analyzed carefully to identify possible objectives.

This was a strenuous process, that kept on for a while. At the time the preliminary project for the bachelor's thesis was delivered, it had been narrowed down to be a topic related to Condition Monitoring and the working title at that time was: "Condition Monitoring of a hydropower plant cooling system".

Further, various theories and concepts were examined, and based on the signals that were available, it was determined that the project's focus would be the generator cooling. The project will investigate the potential for Condition monitoring of a generator cooling system in a hydropower plant using physical models on data from the SCADA system. The definition of the problem for this project is:

#### **Condition monitoring of a hydropower plant cooling system**

### 1.3 Limitations

The project will only have access to data from one hydropower plant. Available data stretches from the end of March 2017 until today. The data is historical SCADA data, which means that the resolution and sampling frequency are given and there can be no real-time analysis. The format of the data is set, and the thesis will mainly focus on the cooling system of the generators. The data, its results and analyses will not be verified in any way. Errors or systematic faults may be present.

### 1.4 Structure of the report

Section 1, the introduction of the report, highlights the background of the project, its scope and limitations. The background theory in section 2, will focus on the theory and background of hydropower plants and their SCADA systems, with particular emphasis on the generator cooling system. Maintenance strategies, Condition Monitoring and hydropower maintenance will also be evaluated. Section 3, method and approach, will present the methods and tools that have been used. It will also explain the data that was analyzed

and how the work progressed. Then, in section 4, the final results are presented. They are discussed in the following section, section 5. Next, the conclusions for the project are drawn and the outlook is presented in section 6. The appendices include piping and instrumentation diagrams in a larger format than what is present in the report, in appendix A, and some selected Python scripts are presented in appendices B, C and D.

Many of the references for the report are online, and an IEEE-style reference system is therefore applied. The reference method that is used separates between a reference used for a section or a single sentence. If the reference is given in the sentence, before the period, it is applicable for the current sentence only. This is often applied where specific numbers from a reference are reproduced. Direct quotes and figures are marked with their exact page reference. If the reference is given after the sentence, often at the end of a paragraph, it applies to all foregoing sentences of the paragraph.

SI-units are used in most cases, but there are some exceptions from the base SI-units. These exceptions are applied when the author considers another unit to be more suitable, or more commonly used, than the corresponding base SI-unit. The exceptions are where minutes, hours and days are used in place of seconds.[11] Numbers, dates, and expressions of time are formatted according to the guidelines of The Language Council of Norway[12].





## 2 Background theory

This chapter will introduce the theoretical background of the thesis. The operation and behaviour of the most common type of hydropower turbine and generator will be examined. Hydropower SCADA systems and other auxiliary systems will be reviewed. Cooling systems, with a particular focus on hydropower generator cooling are detailed, before different types of maintenance and strategies, as well as hydropower maintenance, are evaluated.

### 2.1 Hydropower plants

Hydropower plants are complex constructions that consist of several subsystems. Figure 2.1 shows an edited cross section illustration of Brattset hydropower plant. On the left, in the red box, one can see the penstock. The penstock is the waterway that leads the water from a reservoir into the power plant and the turbine's spiral casing. The turbine's spiral casing, green box, is visible close to the center of the drawing. High pressured water from the penstock enters the turbine's spiral casing tangentially and moves along the casing, before it is turned towards the runner at the center of the casing, by the wicket gates and stay vanes. The water's momentum is transferred to the runner, which will rotate, and as the water passes through it there is a large drop in pressure, before the water axially exits the runner and is led to the draft tube, yellow box, upon which it will exit the power station. The shaft transfers the rotation of the runner to the generator. The generator, orange box, is located on the same axis as the runner and the two are connected through a shaft, blue box.[13, 14]

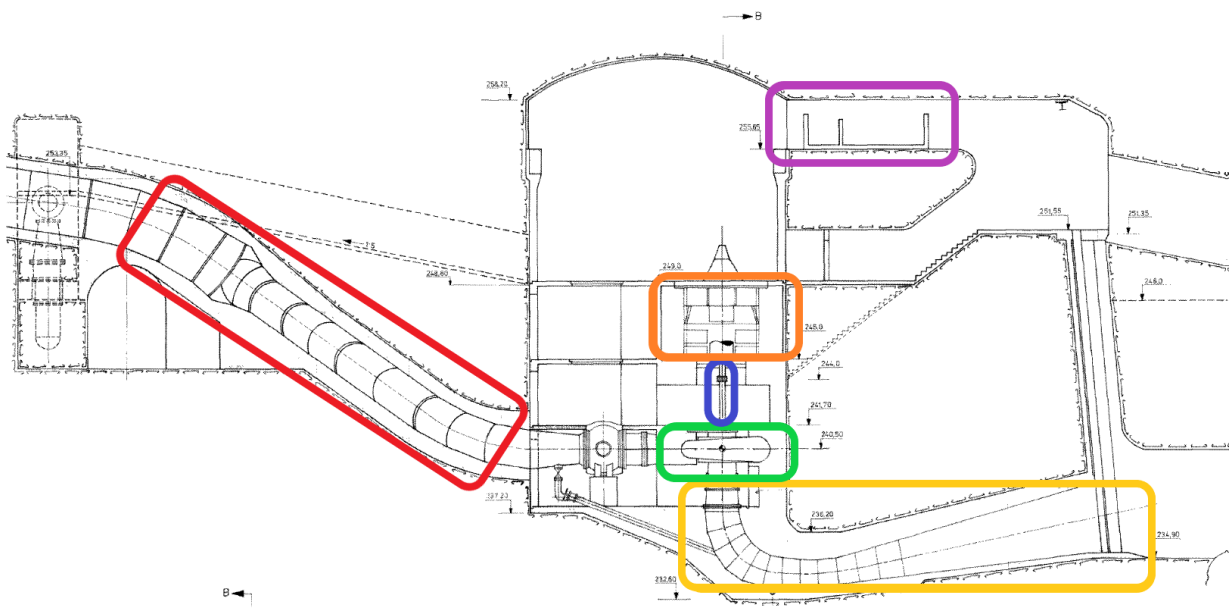


Figure 2.1: Cross section of Brattset hydropower plant. Received from TrønderEnergi

### 2.1.1 Turbines

In order to achieve a high efficiency, the choice of turbine plays an important role. In hydropower there are mainly three types of turbines to choose from, Kaplan, Francis and Pelton. The turbines come in several variations and have many customization options. The type of turbine one ends up installing is often given, due to local conditions like the amount of water available, volume flow rate, and the height difference, head. But the chosen turbine will most times be adapted to the local conditions and is specifically designed for that exact power plant, to achieve the highest efficiency possible. The Kaplan turbine is best suited for river power plants, where there is a lot of water available, whereas the Pelton turbine is better suited for locations that offer high heads, but less water. The Francis turbine, which is an intermediary of the two, is the turbine which is most commonly used.[13, 14]

**Francis turbines** The Francis turbine is the oldest design still used today. The turbine's casing is completely filled with water as the Francis turbine extracts the kinetic and pressure energy of the water. Although the turbine's stay vanes are stationary, the flow rate through the runner can be controlled by adjusting the angle of the wicket gates, shown in figure 2.2.[13, 15]

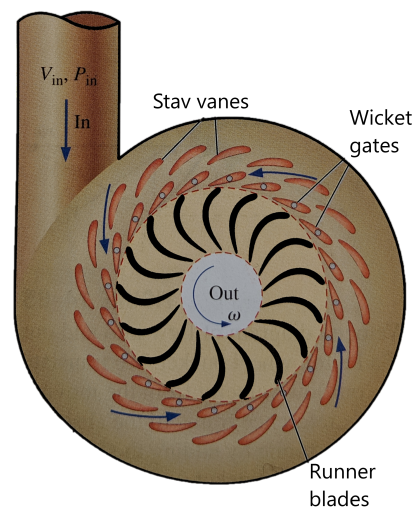


Figure 2.2: Spiral casing with stay vanes, wicket gates and runner blades, modified from[16, p. 838]

### 2.1.2 Generators

The generator is a reversed motor that transforms the mechanical energy applied through the rotating shaft into electrical energy. The generator has a rotating part, the rotor and

a stationary part, the stator, and the energy transformation is achieved through electromagnetic induction. There are a variety of generators, each with its application, but for hydropower plants, synchronous generators are the predominant choice.[17]

**Synchronous generators** A synchronous generator is usually constructed with three phases, a revolving magnetic field in salient poles and a stationary armature. The electromagnetic field on the rotor has a positive and a negative pole. When this magnetic field rotates its flux will vary which will induce a current in the armature windings of the stator. Since the magnetic field has positive and negative poles, the induced current will be alternating.[16, 17]

The stator is composed of silicon-iron-steel laminations that are from 0,35-0,5 mm thick and which are covered by an insulating varnish. The laminations have slots that carry the armature windings, shown in figure 2.3, and the laminations are tightly stacked in a frame that clamps them together to avoid vibrations. The windings are insulated according to specifications of temperature classes. The salient poles of the rotor are constructed of much thicker iron laminations. The holes in the salient pole shown in figure 2.3 are for the windings that set up the electromagnetic field. The field on the rotor is excited by a DC generator, which is usually mounted on the same shaft. The air-gap between the rotor and stator pole is usually about 30 mm.[16, 17]

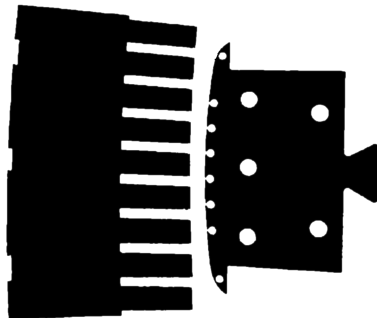


Figure 2.3: Toothed stator segment and salient rotor pole[16, p. 356]

The process of transforming the water's kinetic and pressure energy to electricity is very efficient, but as Wildi states: "Whenever a machine transforms energy from one form to another, there is always a certain loss"[16, p. 122]. The loss Wildi mentions, can not simply disappear, so what happens in reality, is that most of this energy is transformed into thermal energy, instead of electric energy. The losses in a generator depend on its load and can be separated into three main groups, mechanical, due to friction and windage, iron losses, due to induced currents in the conducting parts of the generator, and copper losses, caused by heat-loss in the conductors. The "lost" energy is dissipated as heat, and that heat must be transported away to avoid a drop in generator performance or life expectancy. The temperature rise, which is the temperature difference between a machine's warmest accessible part and its ambient temperature, have a great impact on its service life, and according to Wildi[16, p. 128]: "Tests made on many insulating materials have shown that the service life of electrical apparatus diminishes approximately by half every time the

temperature increases by 10°C.” This means that a generator in hydropower plant with a life expectancy of 40 years at a rated temperature of 30°C, will have a service life of only 20 years at 40°C. [16, 17]

### 2.1.3 Cooling systems

Every hydropower plant must to a large extent be adapted to the local conditions and there is an almost infinite number of compositions of turbines, generators and methods of construction. This also applies to the cooling systems of the hydropower plants, which are uniquely adapted to each plant. There may therefore be large variations in how the cooling systems are designed from one plant to another. There are, however, some common features.

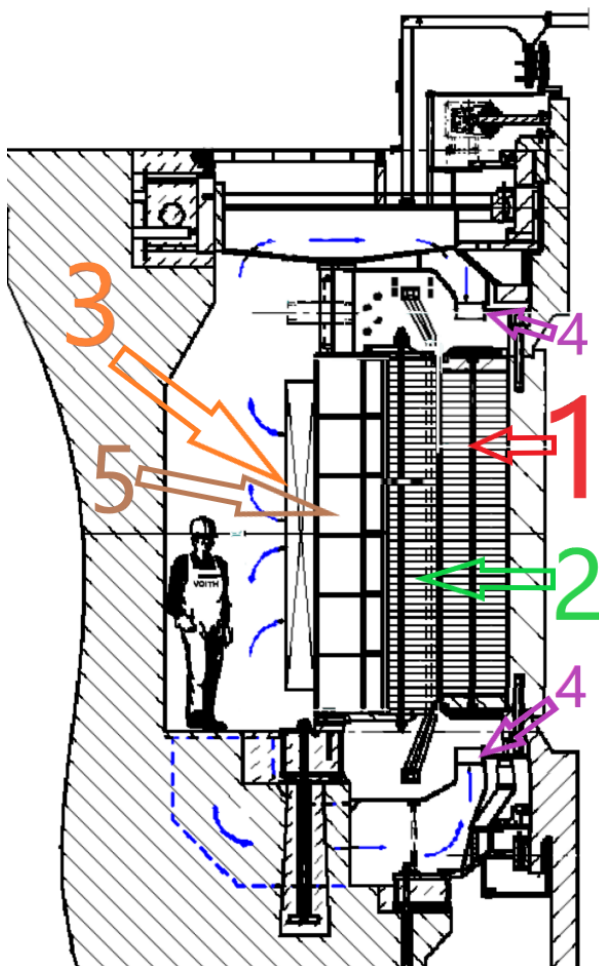


Figure 2.4: Generator air cooling loop, received from Voith Hydro

Machines in most hydropower plants are water cooled, given the fact that water is an effective coolant and an abundant resource in a hydropower plant. This will, however, require piping and other types of infrastructure, in addition to maintenance of the associated equipment. The machine’s losses, and thereby the heat it dissipates, are directly linked to its size. Smaller hydropower plants may therefore use simpler, less effective, types of air cooling, or a combination of air and water cooling. On the other side, direct water cooling may not be sufficient for very large machines. In those situations it may become necessary to use more effective coolants, such as pressurized hydrogen, to further improve the cooling.[16]

The cooling system plays a crucial part in any hydropower plant, and if it ceases to work, production will quickly stop. The system will ordinarily supply several of the hydropower plants subsystems. Even though there are more than one turbine in the power plant, there is usually just one, joint, cooling system. The subsystems it supplies may include the turbine bearings and the transformer(s), the turbine regulator, and of course the generator(s).

**Generator cooling systems** The generator accounts for the biggest losses of the hydropower plant's subsystems and for that reason it is the cooling system's largest consumer. For a water cooling system this becomes apparent by inspecting the dimensions of the pipes that supply the generator cooling system.

The generator itself is normally cooled down by air, but the air it uses as coolant, is refrigerated by a water fed heat exchanger. The blue markers in figure 2.4 illustrate the loop of the cooling air for a generator cooling system. The space in which the air circulates is closed and if one looks closely at the figure, one can recognize the rotor, marked by the arrow with a red 1, and the stator, marked by a green 2, and the very narrow air gap between them. The heat exchanger is marked by an orange number 3. The heated air is transported through the stator laminations and the heat exchanger radially by forced convection. The pressure needed for forced convection is created by the rotation of the rotor and the small fan blades that are attached to the top and bottom of the rotor, marked by two purple number 4's in figure 2.4.[16, 17, 18, 19]

As stated, the air is cooled when it moves through the heat exchanger, and the heat exchanger is refrigerated by circulating water from the hydropower plant cooling system. The heat exchanger is equipped with a water inlet and outlet for circulation, and plugs for venting and draining the cooler. Its frame is usually made of stainless steel, and the body is made out of a metal with high thermal conductivity – usually copper – that is finned to create an area that is as large as possible, for efficient heat transfer. A generator air cooler is shown in figure 2.5.[19, 20, 21]



Figure 2.5: Generator air cooler at Brattset hydropower plant, received from TrønderEnergi

The most common types of damage for generator air coolers are leakage, fouling – which is algae and other organisms that start to grow on the water side of pipes – and smudging of the coolers on the air side. Leakage of cooling water into the generator can cause it to break down. Small leakages and the other types of damages share the same symptom, where at

first the difference in temperature between the cold water and the cooled air rises a little, up to 10°C. Untreated, this temperature increase will continue as the damage worsens, until the temperature increases so much that the generator shuts down automatically. An increase in the temperature of the cooling air can also damage other components in the generator. Where the most prominent and serious are degradation of the insulation in both the stator and rotor. The elevated temperature degrades the epoxy in the insulation which makes the materials brittle and may lead to cracks in the material which reduce its insulating abilities.[21, 19]

Some hydropower plants may require additional cooling, which is possible to achieve by direct water cooling. This is somewhat more advanced, because pipes for the cooling water must be added to the rotor and stator. The rotor must also have a contraption that can circulate its cooling water while it rotates, which is fairly advanced and not as reliable, and therefore rarely chosen.[17, 19]

### 2.1.4 SCADA systems

All modern industrial systems, like wind turbines, hydropower plants and oil rigs, are controlled by PLCs. PLC is short for programmable logic controller, and PLC equipment is electronic equipment specialized for control, measuring, communication and governing of a system. Any industrial system constructed today is too complex to be handled by one PLC, so there are usually a whole group of different PLCs, controllers and displays with graphical user interfaces connected in a system, called a control system. The system communicates and interacts on different levels to control the industrial system from a control room or a remote location. The term SCADA is an acronym for Supervisory Control And Data Acquisition, and it is the name of the architecture most control systems use. It is common to refer to control systems as SCADA systems.[22, 23, 24]

Most SCADA systems are modular and will usually have a large number of sensors to measure and monitor process values and outputs in a system. It can measure everything from temperature and pressure to current and water levels, depending on what you need and want. The measured values can be used for automatic control, which will give the system high reliability. In some cases there are also integrated control circuits that control a process without communicating with the SCADA system. Besides the electronic SCADA system, there are usually other protective measures like mechanical or electromechanical switches and governors that protect the often highly specialized and very expensive industrial equipment.[16, 25, 26]

As stated, the SCADA system collects data from various sensors, and this data is either collected as digital signals or measurements. We know that an electronic system, like a computer or a PLC, only understand binary data; bits. This means that a PLC easily understands digital signals, – which are on or off – the signal is simply translated to a bit. A real process value, like pressure, temperature or flow, on the other hand, is continuous, and a continuous value cannot be represented as a discrete value of 0's and 1's directly. The continuous process value is therefore approximated by taking a sample of the process value, and transforming this sample into a discrete value. This process is called discretization,

or more easily understandable, Analog to Digital Conversion. Once a value has been discretized it can be represented in binary digits, making it readable by the PLC and any other computer.[27, 28]

**Information loss, priority and resolution** Sampling is a widely known technique, that is frequently used in sound and image processing and compressing. Any sampled value will only represent the value of the process it was sampled from, at the very instant the sample was taken. The value the process has between two samples, will not be measured, and this information will therefore be lost. For most processes, however, the sample frequency is adapted to the rate of change of the process value, to ensure that as little information as possible is lost. Completely loss-less sampling would make it possible to perfectly represent an analogue process value based on its discretized, digital values.[29, 30, 31]

In a SCADA system there is an internal hierarchy. A signal or a measurement will be given priority according to their rank in this hierarchy. A high priority measurement may therefore be sampled at a higher rate than other measurements.[32]

Another factor that will affect the accuracy of the digitized value is the resolution of the sample, sometimes called the measurements granularity. The smallest change in the process value that is perceivable by a sensor, is referred to as the sensor's resolution. Although this is an important feature of any sensor, it may be other features that limit the resolution of a measurement in a SCADA system. There may be limitations in the system's properties, such as its processor or system architecture.

We assume that a system communicates using data words with a size of two bytes, 16 bits. Some bits are reserved for transfer and one bit is reserved for the sign, so there are eleven bits available for any measurement. These eleven bits provide a data range of  $2^{11} = 2048$  possibilities for number representation, or an accuracy of  $\frac{1}{2048} \cdot 100 \approx 0,05\%$ . For a temperature measurement that has a measuring range of  $\pm 261^\circ\text{C}$ , its resolution will be:

$$\frac{\text{Measuring range}}{\text{Data range}} = \frac{\pm 261^\circ\text{C}}{2048\text{bits}} \approx 0,13^\circ\text{C/bit} \quad (2.1)$$

**SCADA system application** In reality a SCADA system will sample data at irregular intervals. The sampling algorithm, or the rules that determine when a sample of a measurement or a signal is to be taken, can be influenced by a variety of factors. To keep the rate of transfers low, most measurements will be updated upon change. A measurement may be transferred only when the change since last transfer is, as an example, 1%, but this will vary from one measurement to another. The importance of the measurement or signal is a key factor. Other influences may be the rate of change in the measured value and the time since the last sample was taken. Each signal or measurement may have a different set of rules and underlying factors that affect its interval. The irregular sampling interval of each signal, and the possible variations in the factors that determine the sampling, contribute to the large variation of sampling frequencies within the system.[33, 34,



35, 36]

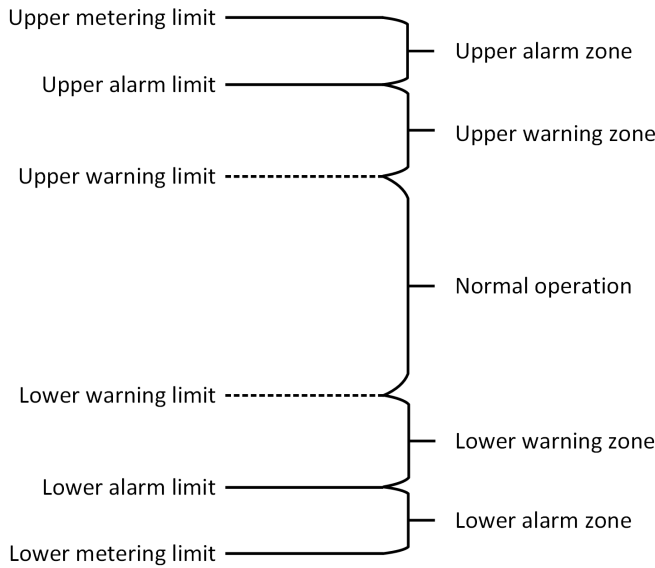


Figure 2.6: Operation, warning and alarm zones for a measurement in a SCADA system, inspired by [37]

temperature rises above this threshold, a warning may be triggered. Should the temperature go  $20^{\circ}\text{C}$  above normal operating temperature, this could damage its material and shorten the expected lifetime, as stated in subsection 2.1.2. The SCADA system might in that situation be configured to go into stop mode to prevent further damage. The concept of operating, warning and alarm zones of a signal is shown in figure 2.6.[34, 38, 39]

The SCADA system is meant to control and protect the plant, automatically or semi-automatically. That means that it will either notify the operator or act on itself – according to a pre-defined pattern – should an anomalous situation arise. An anomalous situation is a transient event, where something is not going according to plan, and may come as result of a whole range of issues. An anomalous situation will most likely carry some symptoms, such as an increase in temperature or pressure, and a well configured SCADA system will act or react on this symptom, to prevent further damage. To trigger a reaction in the SCADA system it may be configured with predetermined thresholds, like temperature thresholds. For a generator the first threshold may be set to a value  $10^{\circ}\text{C}$  above the normal operating temperature. If the tem-

### 2.1.5 Brattset hydropower plant

The Brattset hydropower plant is located in Rennebu kommune in Trøndelag fylke, Norway, and was commissioned in 1982. The plant is owned by Kraftverkene i Orkla, and operated by TrønderEnergi Kraft. The hydropower plant has two identical Francis turbines rated to 40 Mega Watt(MW). The gross head is 273 m and it produces 400 Giga Watt hours(GWh) annually. The generators have a rating of 46 Mega Volt Ampere(MVA). Like in most hydropower plants the generators are synchronous, and the generator is designed for temperature class F, but it is operated as class B, according to the NEK IEC 60034-1:2017 standard[40]. [41, 42, 43]

Brattset hydropower plant's current SCADA system is engineered by Voith Hydro and was commissioned in 2014. The power plant is a part of a Voith Hydro pilot program and it is equipped with a data diode that extracts messages, temperatures, alarms and measurement values from the SCADA system. This data is stored in a database and made available for off-site access.[32]

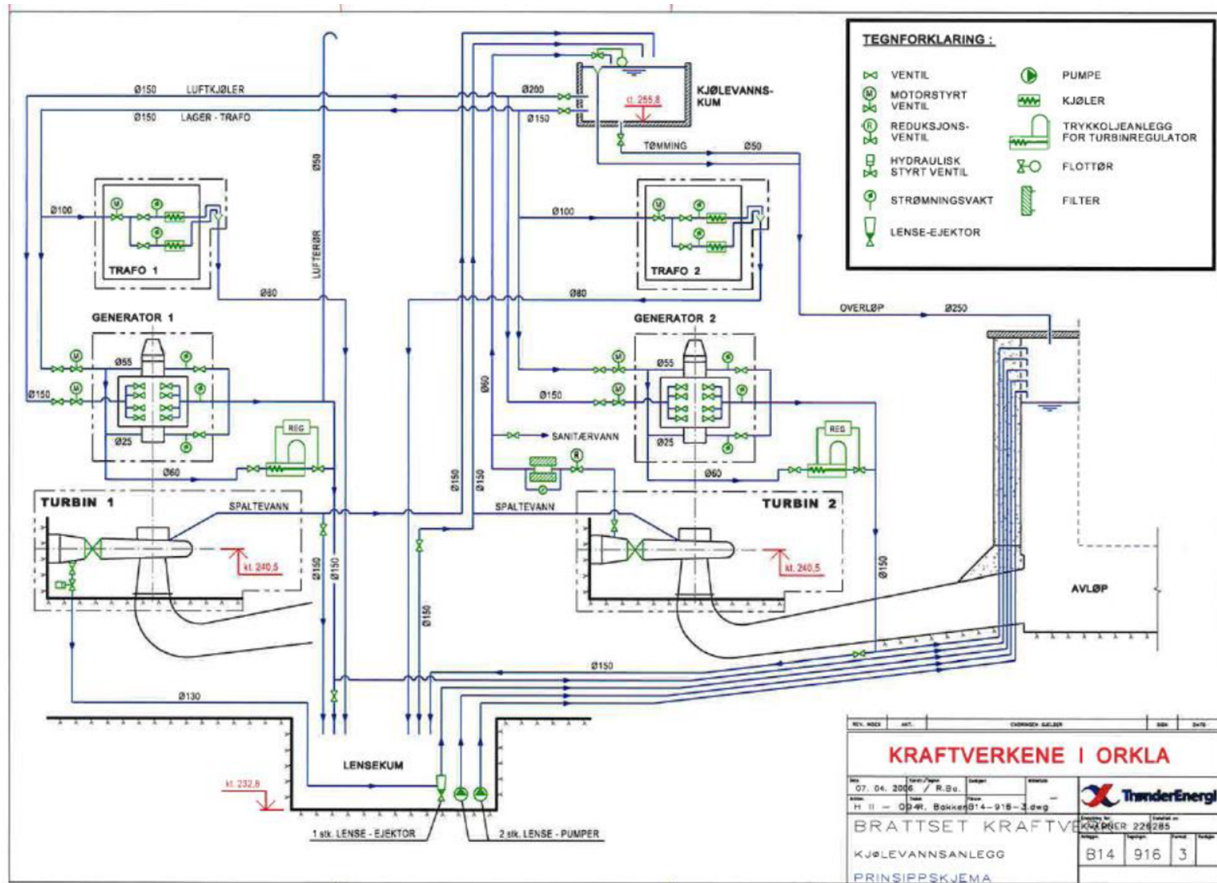


Figure 2.7: Piping and instrumentation diagram for Brattset's cooling system, received from TrønderEnergi

**Brattset's cooling system** Like most hydropower plants Brattset relies on water as coolant for the cooling system. The cooling water is stored in the cooling water sump at the highest point of the hydropower plant, and the water in the sump is supplied by intakes close to the turbines, with a reserve intake in the penstock of turbine 2. The cooling water sump is marked with a purple box in figure 2.1. From the sump the cooling water is fed through a system of insulated stainless steel pipes to its consumers. The flow of cooling water, from the sump through the system, is forced by the potential from the height difference between the cooling water sump and the outlet of the cooling water.

The piping and instrumentation diagram (P&ID) in figure 2.7 illustrates the cooling system. This diagram is also given in full size in appendix A. The cooling system's consumers are transformer 1 and 2, the upper and lower bearings of turbine 1 and 2 and the cooling for generator 1 and 2, illustrated by the four air coolers at the center of each generator. The largest consumers are by far the generators. In recent years there have been some updates to the cooling system, which have not been included in the P&IDs. The cooling of turbine regulators are no longer a part of the cooling system.

**Brattset's generator cooling system** The flow of cooling water to the generator air coolers is monitored by electronic flow meters. The flow meter will notify if the flow deviates from set limits. The amount of water that enters the air coolers is governed by a directly controlled mechanical valve with temperature feedback. This means that this valve is controlled by an additional controller, separate from the SCADA system. One of the reasons there is a separate controller is that generator air coolers are considered so important that they will have to uphold their function, should the SCADA system break down. How this controller is parameterized is not known. It may have been set when the system was commissioned, if it has not been replaced or changed since then. In addition, there are two mechanical contact thermometers for the cold air, and two for the hot air. They provide additional security, and will if they are triggered give warning in the SCADA system, and if necessary they will initiate a shutdown to prevent further damage.

The cold air temperature is measured using four PT100 elements, one for each air cooler. The element that measures the temperature of the cold air is shown in center of the air cooler in figure 2.5. PT100 elements, PT is an acronym for platinum, can measure temperature by utilizing that one of the properties of platinum is that its resistance changes with temperature, 100 means that the element has a resistance of 100  $\Omega$  at 0°C.[44] The increase in resistance with temperature is almost perfectly linear and the element is considered very reliable, according to IEC 60751:2008[45].

The hot air in the generator is considered to be uniform and its temperature is measured by two PT100 elements close to the backside of the air coolers, indicated by the brown arrow in figure 2.4. The velocity of the air is not known, and is considered difficult to calculate due to thermal time constants and the fact that Brattset was designed and commissioned at a time when the progress and possibilities in computational fluid dynamics was not as good as today.[17]

The air coolers on one of the generators have recently been replaced, the replacement for the other generator is scheduled this fall. The only information on the oldest air coolers are that they have a weight of 385 kg. The model that will replace them have a weight 170 kg, an inner volume of 32 l, a max operating temp 99°C. It operates at a working pressure of 0,6 MPa. The new coolers that have been replaced have a weight of 170 kg, an inner volume of 31 l, and a max operating temp 99°C. It also operates at a pressure of 0,6 MPa. All specified weights apply to an empty cooler.

## 2.2 Maintenance

All industrial processes require some kind of maintenance, and the maintenance may involve lubricating a valve or replacing a damaged part. It is no coincidence that these two examples are mentioned, as they represent the two main strategies of maintenance. The strategies are preventive and corrective maintenance. Preventive maintenance takes aim of preventing faults, and the maintenance activity is performed *before* a component breaks down. The goal is to extend the life expectancy of the component by scheduling maintenance according to certain time intervals or other criteria. Corrective maintenance is performed *after* the component has broken down. And the goal of a corrective maintenance activity is to bring the component to a condition where it can perform a required function.[3, 46]

Whether the preventive or corrective strategy is applied, depend on several factors, and the approach may vary from one business to another. As mentioned in section 2.1.4, will industrial processes often require highly specialized and very expensive equipment. If a key component breaks down, it can cause a long halt in production, due to low availability and long delivery times for specially crafted parts. Special parts or equipment is usually also very expensive. Preventive maintenance is without a doubt more expensive than corrective, but it may often be required and it will probably pay off in the long term.

### 2.2.1 Preventive maintenance

Preventive maintenance is traditionally conducted as predetermined maintenance, where the maintenance activity is carried out in accordance with established time intervals. When one follows this maintenance strategy it is common to schedule the maintenance according to recommendations given by the manufacturers of the components in the system. Another type of preventive maintenance is condition based maintenance. According to the European committee for standardization, condition based maintenance is "preventive maintenance which include assessment of physical conditions, analysis and the possible ensuing maintenance actions"[47, p. 35]. In other words, this maintenance strategy says that maintenance activities should be carried out when there is reason to believe that it is needed. To ensure high system reliability, it is fair to assume that an operator carries out more maintenance than what is needed, in line with the saying: "better safe than sorry". Maintenance carried out on the basis of the condition of a component, will therefore probably save the operator money and may also offer better reliability. But to carry out maintenance based on a component's condition, one must have a method for determining its condition. [47, 48]

### 2.2.2 Condition Monitoring

Condition Monitoring is an "activity, performed either manually or automatically, intended to measure at predetermined intervals the characteristics and parameters of the physical actual state of an item"[47, p. 41]. Condition monitoring is a broad term that can include

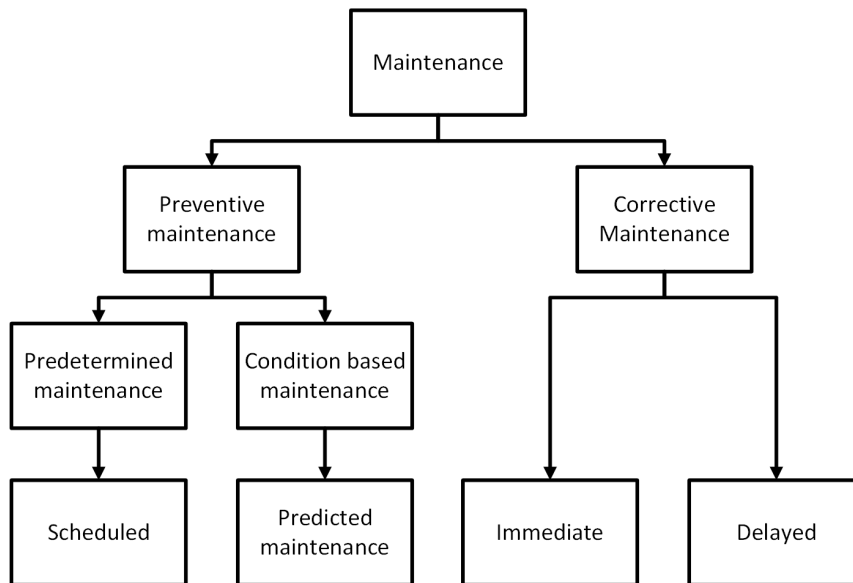


Figure 2.8: Maintenance strategy hierarchy, inspired by [46]

a variety of techniques and methods for monitoring the actual condition of a component. It is not the same as an inspection, it is a more thorough, continuous analysis that takes place over time to give an educated estimate of the components actual physical state. Condition Monitoring is usually carried out in the operating state.[3, 47]

### 2.2.3 Hydropower plant maintenance

Hydropower plant maintenance is usually carried out according to the preventive, predetermined maintenance strategy. The maintenance work is often planned months or even years ahead, as not to interfere with production. Larger maintenance operations for hydropower plants are often scheduled in what is called revisions. Regular revisions are conducted when they are needed, and a large revision may be conducted every thirtieth year. A revision is planned years ahead and it will often include maintenance of all the important components of a power plant, such as the transformer, generator and turbine. A revision is conducted at an optimal time – usually when production is low.

Traditionally, a lot of hydropower plants have been manned by specialists, and these specialists have carried out preventive maintenance activities on a daily basis. Today, this is not as common, and the operator usually has a group of people who are responsible for the maintenance of several hydropower plants.

### 2.2.4 Maintenance of Brattset hydropower plant

The maintenance of Brattset hydropower plant is conducted by its operator, TrønderEnergi. The information following is provided by them. TrønderEnergi use a maintenance program, IFS, to schedule maintenance. They conduct monthly inspections, and use a system where they give the most important components a score according to their condition. This score

is monitored over time. Large revisions of the hydropower are conducted very infrequent, once every 20-30 years.

**Maintenance of the generator cooling system** The generator air coolers are visually inspected once a month and they are cleaned once a year, both on the air and water side. They are cleaned with a specialized cleaning instrument and chemicals. TrønderEnergi expects generator air coolers to have a life expectancy of 40 years.

Older cooler models have parts that are constructed of steel. They are prone to corrosion that could cause leakages with severe water damage. This damage may cause generator breakdown. The downtime following such an event is unknown, and will rely on the type of fault that has occurred. Repairs will probably take a few days, whereas new coolers will take months to provide. New coolers cost around 700 000 NOK.



### 3 Method and approach

This chapter will describe how data were retrieved from Voith's Cloud solution, and how this data were pre-processed, analyzed and visualized using Python. Further, will it highlight the methods and approaches that were used. Partial results will be presented where they are needed, to clarify how and why the work progressed.

#### 3.1 Project progress

The progress towards condition monitoring follows the outline of figure 3.1. The generator air coolers are the chosen components, their failure modes which are lower cooling capacity, and their symptoms of failure modes which are increased temperature differences between the cold cooling water and the cold air from the cooler are known. There are no measurements of the temperature of the cold cooling water that could be used, but there are measurements of the warm air, checked close to the inlet of the air coolers, and the cold air, measured close to the outlet of the air coolers. The development of the difference in air temperature between those measurements may indicate the condition and performance of a cooler.

We know that the amount of cooling water to the generator air coolers at Brattset is regulated by a controller that has a feedback loop. Which temperature this controller uses for regulation is not known, but it is assumed that the controller will increase the amount of water to the coolers when there is a temperature rise.

There are two measurements for the temperature of the warm air, and both measurements were included. As stated in section 2.1.5 is the temperature of the air on the warm side assumed to be uniform, and therefore has the mean of the two measurements been used. There are four cold air measurements, one for each air cooler, which were all included. As mentioned in section 2.1.2, will the temperature of the generator increase with its load, and the power was therefore included. The actual output power is a regulated variable that will depend on other conditions and may fluctuate, for that reason has the setpoint of the power also been included[36].

#### 3.2 Data export

Brattset's SCADA data is accessible through the Analyzer on Voith's cloud solution. The Analyzer allows the user to select any of the around 1200 signals and displays them graphically. Around 15% are analogue measurements. The user can select a combination of signals and set the time span for which they are displayed. Figure 3.2 shows a screenshot of Voith's Analyzer with the aforementioned variables for generator 1. The two lines, green and blue, highest in the line chart, are the warm air measurements. Their y-axis is the leftmost, with Kelvin scale. The two lines below, in the middle of the chart belong to the MW-axis. The pink line indicates the actual output power in MW, and the brown just below is its setpoint. At the bottom of the chart one will see the four cold air tempera-



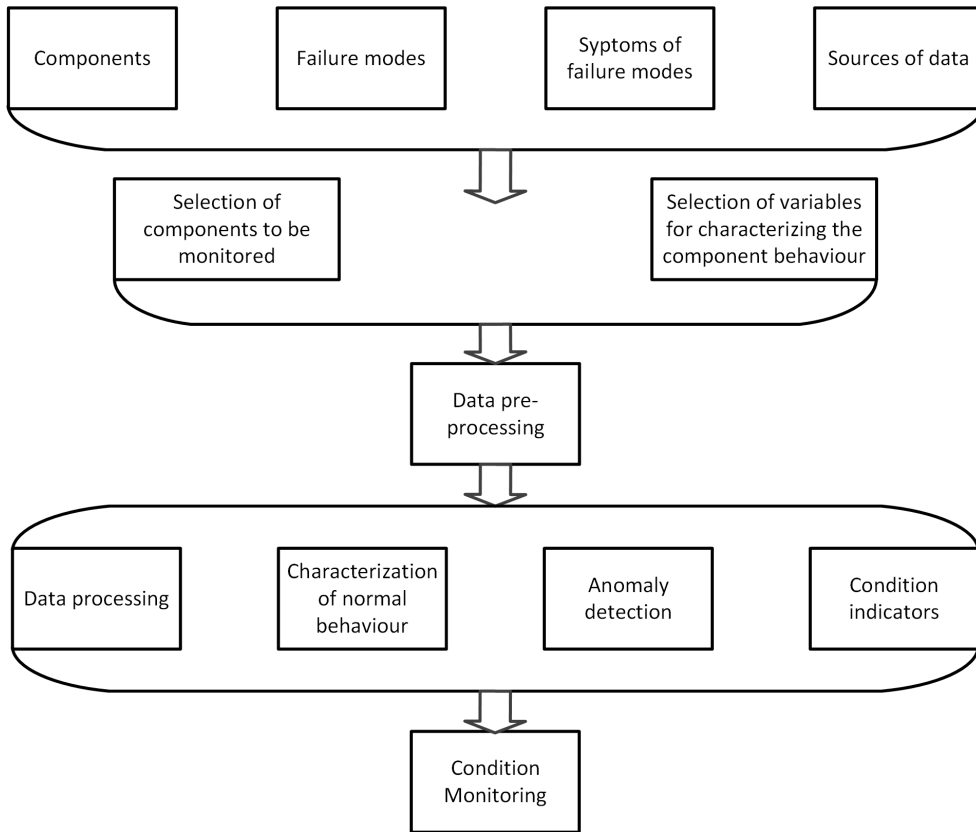


Figure 3.1: The progress towards Condition Monitoring, inspired by [34, p. 6]

ture measurements. The Analyzer does not offer any analysis functionality, but it offers possibilities of visual inspection of the data before it is exported for further analysis.

The data is exported as a file with comma separated values(CSV), illustrated by figure 3.3. The analyzer allows the user to export a maximum of 10 000 rows, and data is available from the end of March 2017 until today. The user is allowed to specify the time span for the values that are plotted in the analyzer, and at the same time this will specify the time span of the exported values. The specified time span has a large influence on the number of measurements in the exported files and their accuracy.

### 3.3 Pre-processing

The Python script that pre-processes the CSV-files was designed to be as general as possible, so that it would be capable of analyzing any file exported from the analyzer without any special precautions, to uphold the integrity of the data.

The script imports the data to a Pandas DataFrame. Pandas is a popular OpenSource framework for Python, and the DataFrame is a flexible data-type that can store large quantities of data. The data in a DataFrame is stored in columns and can be addressed and altered almost like data manipulation in a database. The data from the Analyzer was stored in reverse chronological order, so the first manipulation of the data was to reverse it, to chronological order.



Figure 3.2: Screenshot from Voith's Analyzer

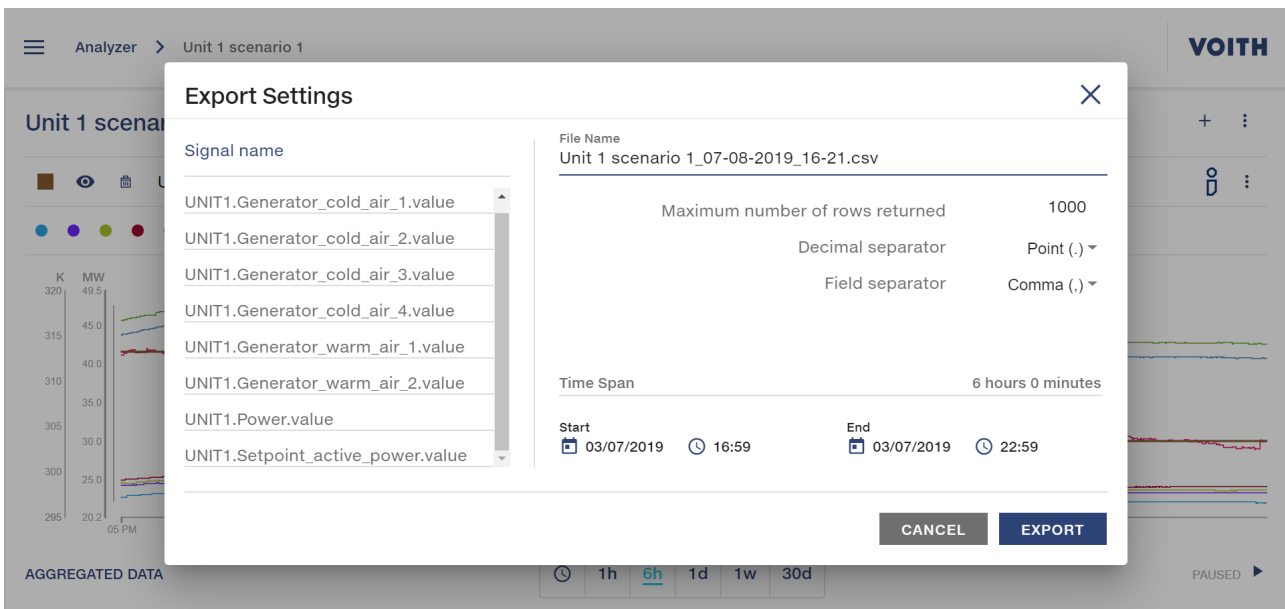


Figure 3.3: Screenshot from the analyzer's export function

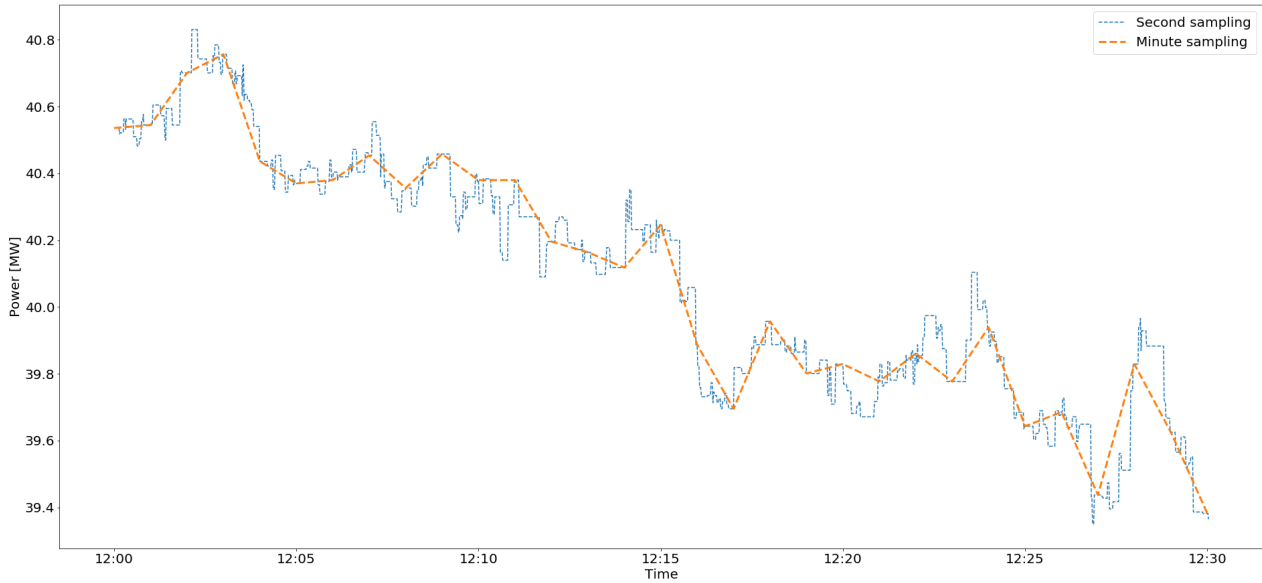


Figure 3.4: Minute and second sampling over 30 minute periode for power output

To do the work of analyzing the data as convenient as possible, all the signals belonging to one analysis were combined in one plot in the analyzer and was then exported as one file. This also means that one file will contain many measurements, and the number of measurements may vary from one file to another. The script takes the dynamic number of signals into account. The number of measurements for each signal may also vary, meaning that the same timespan may yield a different number of measurements. To overcome this challenge, each signal is processed, one by one, to set the sampling of each signal to the same frequency. The values are then forward filled, where the last value is filled forwards to the next value, according to the sampling rate. Several resampling techniques were compared, as shown in figure 3.5. The figure shows that the resampling techniques may yield approximately the same result, but the end result will depend heavily on the sampling frequency and the selected time span. Interpolation techniques will construct values for the resampling. It was decided that the forwardfill produced the most credible and real results. Different sampling frequencies were also considered, see figure 3.4.

Next, the script combines the signals in a DataFrame where the first common timestamp is found. This is set as the joint starting point for every signal. Subsequently, the last common timestamp is found, and set as the common end point. Given that the signals now have the same starting and end point, and sampling rate, they have the same number of values and the same dimension. Any duplicates present are dropped, and the data is ready for further analysis.

#### 3.3.1 Proof of Concept

First a proof of concept was developed. All cold air measurements for turbine 1 were combined in a plot over a wide time span, of approximately 16 days. Next, the average

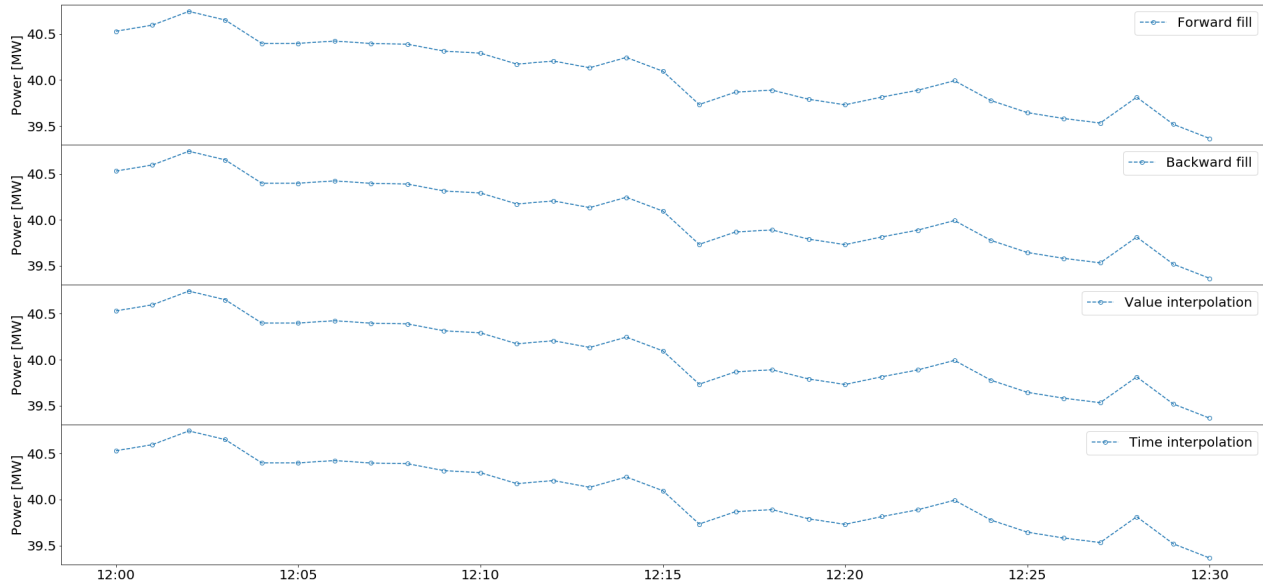


Figure 3.5: Comparison of different sampling techniques

temperature was calculated. Then, some imagined thresholds, at the average temperature plus and minus 1%, were calculated and plotted, as shown in figure 3.6. The next iteration used the two standard deviations of the calculated mean to establish the thresholds.

### 3.3.2 Change in air temperature over the coolers

Following, a plot was developed to uncover whether it was possible to determine the cooling effect of an air cooler based on the mean of the warm air and its cold air temperature. Data covering a large time span, of around 32 hours, that also included a change in output power of approximately 5 MW, from around 16:00 on the 28<sup>th</sup> of August to 00:00 on the 30<sup>th</sup> of August 2018 was used, as shown in figure 3.7. This analysis gave results that was used later in the project.

### 3.3.3 Quantitative vs qualitative approach

Initially the pre-processing of the data was done using data sets with large time spans. It was assumed that a quantitative approach set to wide time spans would have a greater chance of revealing the trends and tendencies one was looking for. When the script that pre-processes the quantitative data was finished it turned out that it was not directly reusable by another file set to an equivalent time span, but for another period in time. It was uncovered that some values outside the specified time span were included in the exported files. There were no consistencies in the number of values that were added, and values were added both before and after the specified time span. For this reason it would have been difficult to automate the process of pre-processing. The quantitative data also had some inexplicable values, that after thorough analysis were deemed unlikely

### 3. Method and approach

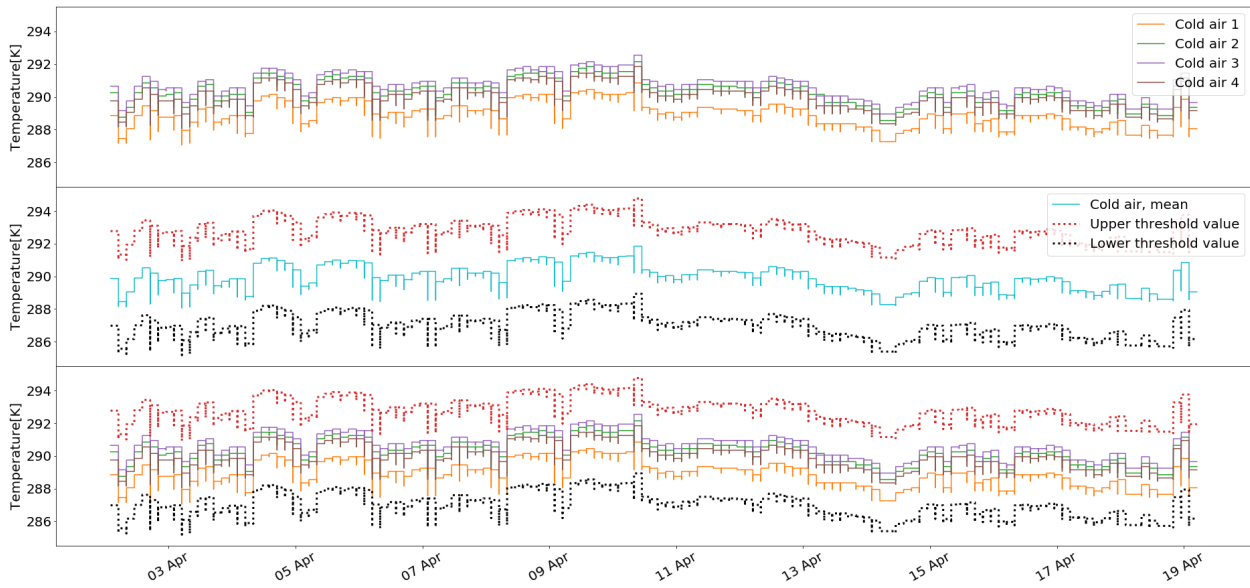


Figure 3.6: Proof of Concept using cold air measurements from turbine 1, legend in plot 1 and 2 apply to plot 3

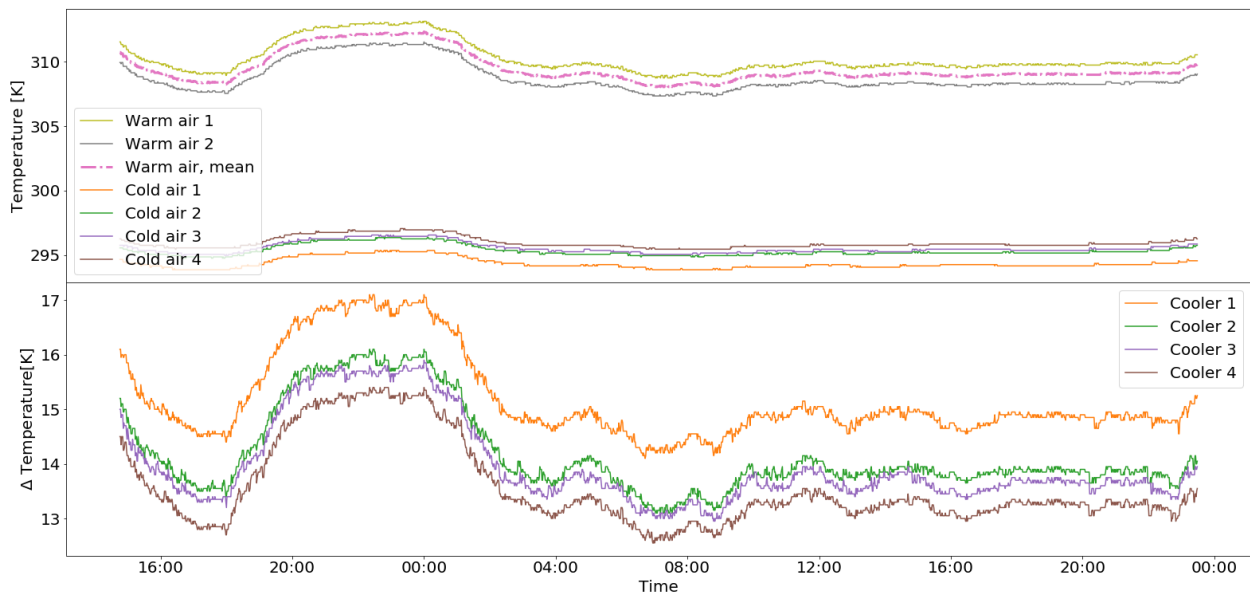


Figure 3.7: Effect of cooler illustrated as the change in temperature

and consequently false. The false values indicate a disproportionately large change in the measurement value over a short period of time. In table 3.1, on row four, one can see that there is a highly unlikely change in the power output of approximately 5 MW over the course of a second. The fault has the same failure mode every time, for all types of measurements. Even though the time span, its size, and the actual timestamp of the unlikely, system constructed, value differed every time, it always occurred close to what is perceived a real value. An example illustrating values from outside the specified span is also present in the table. The specified time span for export is on the 30<sup>th</sup> of May, but some values from the 21<sup>st</sup> of June are also included.

It was ruled that the inexplicable values were the result of some sort of aggregation done by the Analyzer to compress the data, for faster visualization, transfer and download. This is probably implemented for performance reasons, but ended up giving false data with bogus measurement values. Given that the system is in a trial phase and has not yet been commercialized, there is no available documentation supporting this feature.

Table 3.1: Inexplicable power output values

Timestamp	Value	Unit
6/21/2019, 11:47:26 AM	30,154	MW
6/21/2019, 11:47:26 AM	30,154	MW
5/30/2019, 11:58:00 PM	22,297	MW
5/30/2019, 11:57:59 PM	17,707	MW
5/30/2019, 8:58:00 PM	24,368	MW
5/30/2019, 7:57:59 PM	11,606	MW

Since it was not possible to avoid or remove the source of the false data, it was decided to use qualitative data, believed to not include the constructed values. Consequently, investigations to uncover a fitting time span for export were conducted.

By means of manual inspection it was uncovered that data set to a time span of about 12 hours, or less, did not contain data that involved large changes in measurement value over a very short time period. 12 hours was therefore considered the largest time span that could be used for export.

### 3.4 Time span

As specified in section 2.1.4, there is trade-off between the number of samples over a given time period and the accuracy of the samples. For a given amount of samples, will the time period it is spread over highly influence their accuracy.

It was not known how many values the Analyzer returned for given time spans, so to find an appropriate time span and its associated accuracy within the 12 hour limit, data with various time spans were exported, plotted in Python and analyzed in Excel. All the timeseries are from the 22<sup>nd</sup> of June, and the starting time is 00:00. The time spans that were examined were 30 minutes, 60 minutes, 360 minutes and 720 minutes. One can note the high number of returned values, compared to the number of returned values within the

interval, for the 30 minute time span in table 3.2. The values outside the interval, which refers to the issue mentioned section 3.3.3, make up a large part of the total returned values.

Table 3.2: Returned values for different time spans over a 30 minute time period for power output on generator 1 on the 22<sup>nd</sup> of June 2019

	No. of returned values	No. of returned values 00:00-00:30	Values per minute
<b>00:00-00:30</b>	783	255	8,50
<b>00:00-01:00</b>	420	41	1,37
<b>00:00-06:00</b>	1043	28	0,93
<b>00:00-12:00</b>	922	13	0,43

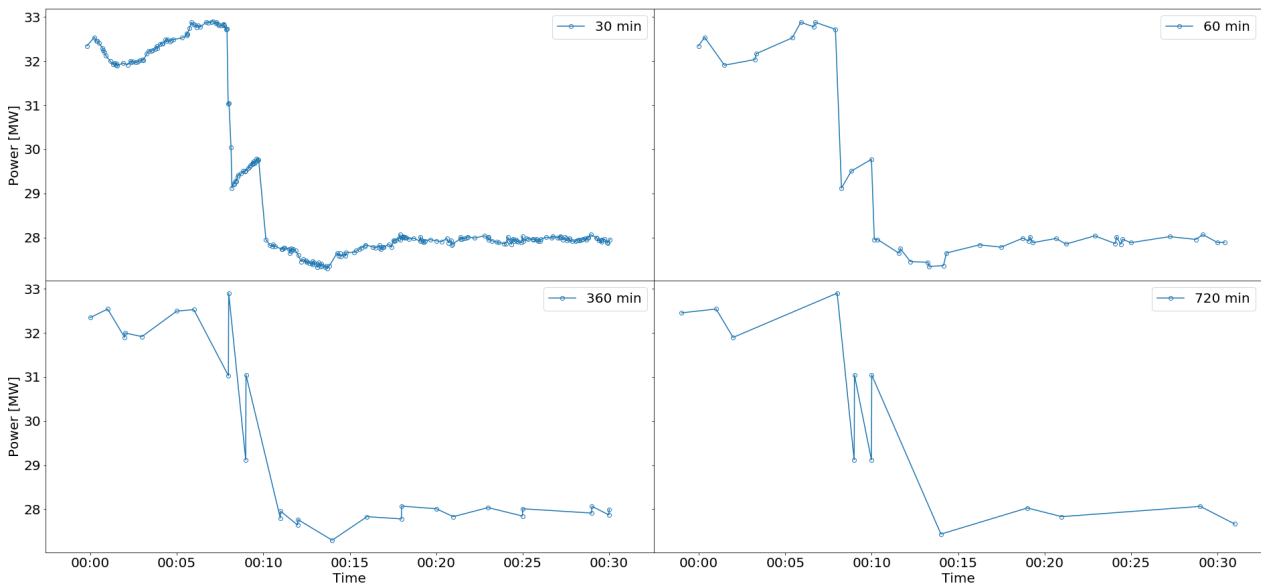


Figure 3.8: Sample frequencies with plots for different time spans over a 30 minute time period for power measurements on generator 1 on the 22<sup>nd</sup> of June 2019

### 3.5 Analysis

As stated in section 2.2.2, Condition Monitoring is carried out in operating state. The operating state for the generator air coolers is considered to be when the generator is running. In section 2.1.2 it is said that the temperature of the generator will vary with its load, so to be able to compare data on the condition of the air coolers over time, the data had to represent approximately the same generator load. We also know that the thermal time constant of the air, implies that it will take some time for the air temperature to reach steady state, following a change in generator load. It was therefore necessary to export qualitative data that represented a stable or nearly constant load over time.

A curve that represents the efficiency of a turbine is quite steep, and it is usually most

efficient at full load. This means that if a turbine is run outside its best efficiency point, its efficiency will be significantly lower. The hydropower plant operators will therefore usually run the hydropower turbine at full load, or close to full load, when it is running.

Taking all this into consideration, it was decided that the generator should run at an even load for three hours for the temperatures to be categorized as stationary. To increase the probability of finding representative data for the whole time period, it was decided that the generator load – or rather the setpoint of the load, since the actual delivered load will fluctuate – should be full load, 40 MW, or close to full load. The time span of the data was set to 30 minutes, to include as much information as possible.

Next, data covering the whole time period was examined for both generator 1 and 2, to find data that fulfilled the requirements. Effort was put into finding data that represented the whole time period.

For generator 1 nine timeseries were exported, and for generator 2 eight. Information regarding the timeseries are given in table 3.3.

Table 3.3: Time series exported from Analyzer that fulfill the requirements set for analysis

Generator 1			Generator 2		
Date	MW(setpoint)	Time	Date	MW(setpoint)	Time
25.03.2017	39,5	22:00-22:30	14.05.2017	40,0	11:00-11:30
31.03.2017	40,0	10:00-10:30	09.07.2017	40,0	12:00-12:30
28.06.2017	40,0	11:30-12:00	04.10.2017	40,0	15:00-15:30
13.10.2017	40,0	15:00-15:30	27.02.2018	39,0	22:30-23:00
11.12.2017	39,0	13:00-13:30	27.06.2018	40,0	03:00-03:30
28.04.2018	41,0	11:00-11:30	28.02.2019	39,7	17:30-18:00
25.08.2018	41,0	10:30-11:00	27.04.2019	40,0	11:30-12:00
27.02.2019	40,0	19:00-19:30	23.06.2019	40,0	03:30-04:00
08.06.2019	40,0	12:00-12:30			

### 3.5.1 Scenario 1

Scenario 1 concentrates on the difference in temperature between the warm and the cold air of the generator air coolers, for a given power output. Monitored over time, this value can be used for characterization of normal behaviour, anomaly detection and as a condition indicator, which are all steps towards Condition Monitoring, as outlined in figure 3.1.

**Pre-scenario 1** Based on the results shown in figure 3.7. It was decided to further improve this analysis by investigating the change in temperature over a 30 minute time span for each cooler. Since the power output may fluctuate, and probably will affect the temperature if it does so, it was included as a third variable, to explain any differences in the changing temperature over the cooler( $\Delta$  Temperature).

These plots are given for cooler 1 of generator 1 in figures 3.9, 3.10 and 3.11.



### 3. Method and approach

---

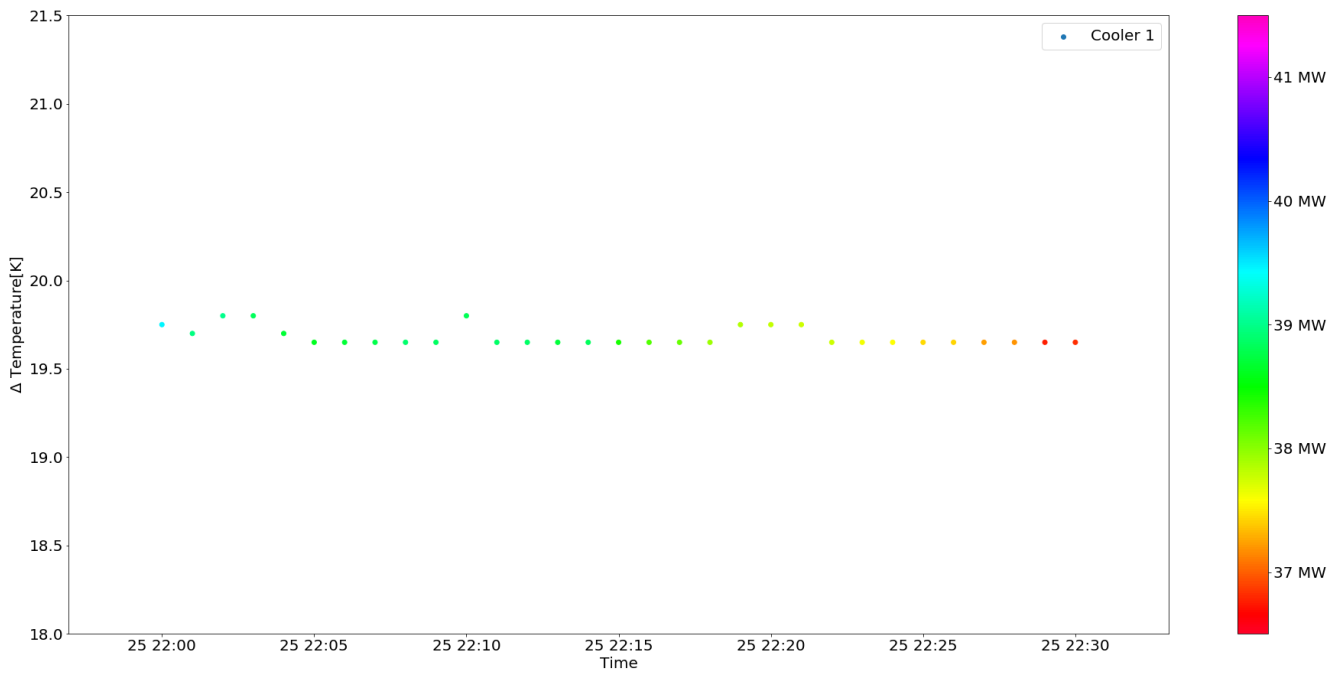


Figure 3.9: Difference in temperature over cooler 1 on the 25<sup>th</sup> of March 2017

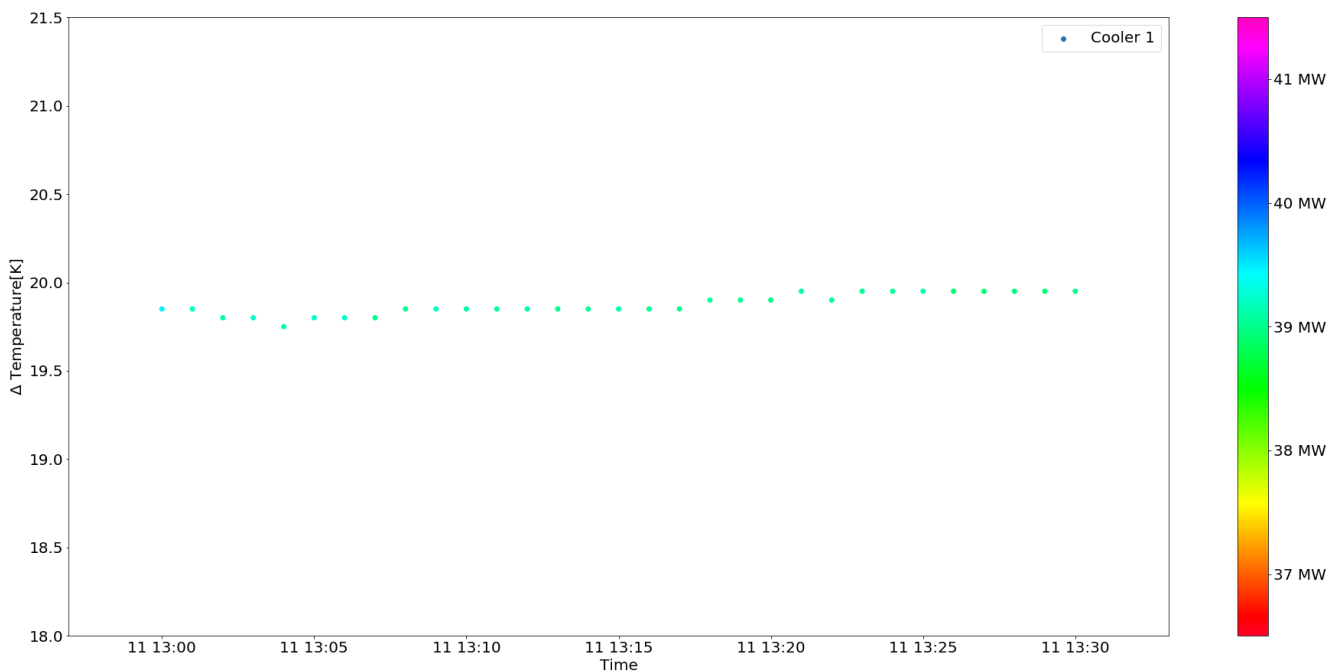


Figure 3.10: Difference in temperature over cooler 1 on the 11<sup>th</sup> of December 2017

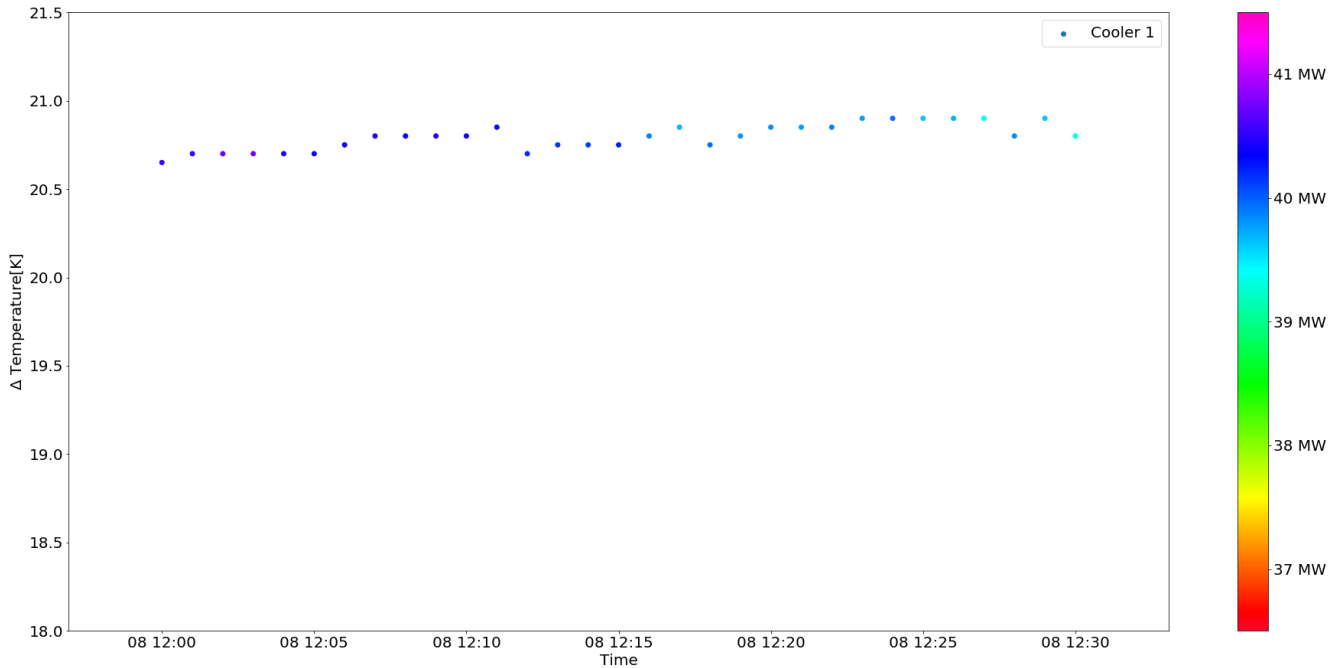


Figure 3.11: Difference in temperature over cooler 1 on the 8<sup>th</sup> of June 2019

**Scenario 1** To give a better understanding of the trend in the condition of all coolers, the mean of the temperature change for all the coolers were plotted. The data that was gathered, 9 time series for generator 1 and 8 for generator 2, was combined in plots for each generator to uncover the condition of the coolers for the whole time period. A linear regression was added, based on the plotted time series, in order to determine the rate of any possible changes in the temperature difference over the coolers.

### 3.5.2 Scenario 2

It is believed that the temperature of the machine will have an approximately linear relationship with the power output, with some wiggle room, of course. In theory, it should therefore be possible to predict the temperature of the air, and thereby the change in temperature given a certain power output. If the temperature is far from the expected value at a given output power, this could be characterized as an anomaly.

A plot illustrating this idea was realized. It illustrates the difference in temperature ( $\Delta$  Temperature) as a function of the power output. It could serve as a combined condition monitoring and anomaly detection feature, where the change in temperature for a given power output is predicted using a linear regression, which was included. Thresholds produced by the double standard deviation of the linear regression, are also plotted.

As stated, was it set as a condition that the generator had to run at even load for three hours to consider the temperatures stationary. This condition is seldom met, apart from when the generator is running at full load, or close to full load. For that reason, will Scenario 2 only include data full load, or close to full load.



## 4 Results

Some of the partial results were introduced in the previous section, section 3, to demonstrate the progress of the project work. The project progressed incrementally, and it was a conscious decision to present the partial results this way. The main results, however, are presented below. First the initial results, the Proof of Concept that displays the use of SCADA data for monitoring, are presented. Next, the final results for the monitoring of the difference in air temperature over the coolers, for each cooler and their mean, known as Scenario 1, are presented. Third, and last, will the results illustrating the expected temperature given output power, Scenario 2, be shown.

### 4.1 Proof of Concept

The result is shown in figure 4.1. The upper and lower thresholds are calculated from the double standard deviation of the mean for all the coolers. These thresholds are then applied to each cold air measurement of generator 1. The plots share the same x- and y-axes. The variation in cooling effect between the coolers are apparent. Where cooler 1 has the lowest temperature, and cooler 3 has the highest temperature. Cooler 2 and 4 are quite similar. The data used for the Proof of Concept are qualitative, and will therefore most likely include the errors mentioned in section 3.3.3. The focus of the Proof of Concept was to demonstrate the use of temperature values from a SCADA system for immediate detection of anomalies. The time on the x-axis was seen as irrelevant, and has therefore been removed.

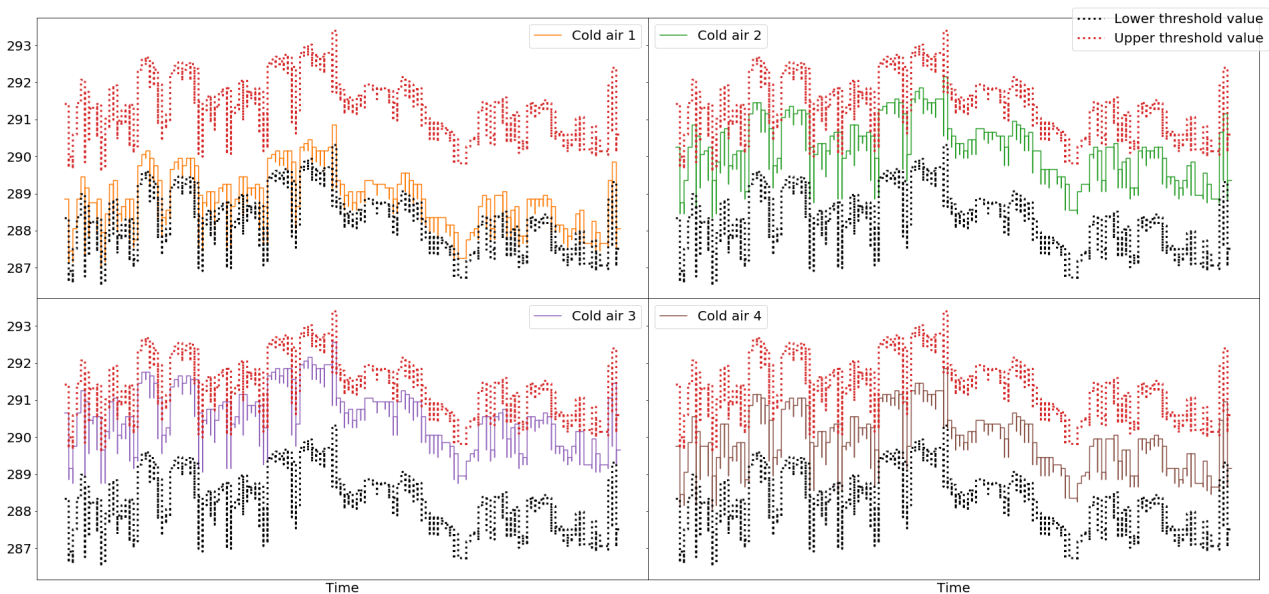


Figure 4.1: Proof of Concept for each individual cooler for generator 1

## 4.2 Scenario 1

The same analyses have been performed on the data from generator 1 and 2. The results for generator 1 are given first, and the results for generator 2 will follow. A walkthrough of the results will be presented to clarify in which order they are presented. Next will each result be described, after the result has been presented.

The results of scenario 1 for generator 1 for each individual cooler are shown in figures 4.2, 4.3, 4.4 and 4.5. The mean of the temperature for the air coolers for generator 1 is shown in figure 4.6. The mean of the temperature for the air coolers for generator 1, without the extreme values – which here are defined as the highest and lowest setpoint for the output power – is shown in figure 4.7.

The results of scenario 1 for generator 2 for each individual cooler are shown in figures 4.8, 4.9, 4.10 and 4.11. The mean of the temperature for the air coolers for generator 2 is shown in figure 4.12. The mean of the temperature for the air coolers for generator 2, without the extreme values – which here are defined as the highest and lowest setpoint for the output power – is shown in figure 4.13.

As stated in section 3, has data from the entire available period been used. All plots will therefore roughly share the same x-axis, but the exact dates are given in the table of exported data, table 3.3. The plots for the coolers of generator 1 and 2, and their mean, will share y-axis respectively. The scale for the y-axis has intentionally been kept, according to which generator the results belong, to make comparison of the coolers easier.

For all the plots, except for those of the mean, figure 4.6 and 4.12 and those of the mean without the extreme values, figure 4.7 and 4.13, has the difference in temperature for the air coolers been calculated and plotted as a scatter for each time series. The scatters are colored according to the value of the output power, and the colorbar on their right indicate the value. A regression has also been calculated and included for all coolers.

Figures 4.6, 4.12, and figure 4.7, 4.13 have the mean of the change in temperature for the coolers plotted as a scatter. They too, are colored according to the output power value. The setpoint of the power has also been included. Note that figures 4.6 and 4.7 share y-axes, the same does figures 4.12 and 4.13.

**Generator 1** Figure 4.2 show that the temperature difference over air cooler 1 of generator 1, is systematically larger then the other coolers. It is about 1,5 K higher than the lowest, which is figure 4.5. Figure 4.3, which represent cooler 2, is almost 1 K below cooler 1, but higher than the other two. Figure 4.4, for cooler 3, is slightly lower than cooler 2, but higher than cooler 4. There are some differences in change in temperature over the coolers, but seems to be fairly similar for all the coolers.

The figure representing the mean of all the coolers, figure 4.6, is placed in between the plots for cooler 2 and 3, and it has about the same slope as the plots for the coolers. According to the color of the scatters we can determine that most of the time series that have been used are fairly stationary. The first scatters, at the setpoint of 39,5 MW are a bit unstable, the same goes for the last scatters, all the way to the right. Figure 4.7, which is a plot of

the mean without the two extreme values at a setpoint of 39,0 and 41,0 MW, presents a leaner slope than the others.

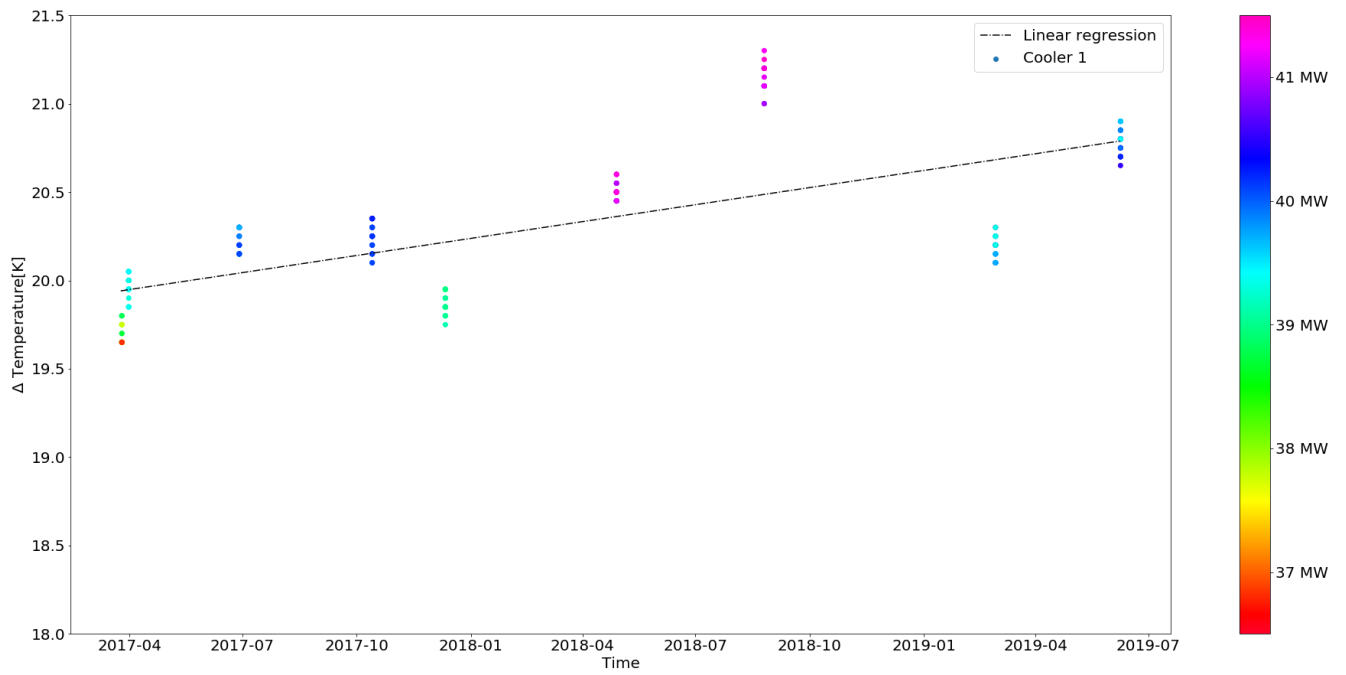


Figure 4.2: Difference in temperature over air cooler 1 for generator 1, linear regression, and color according to the output power

## 4. Results

---

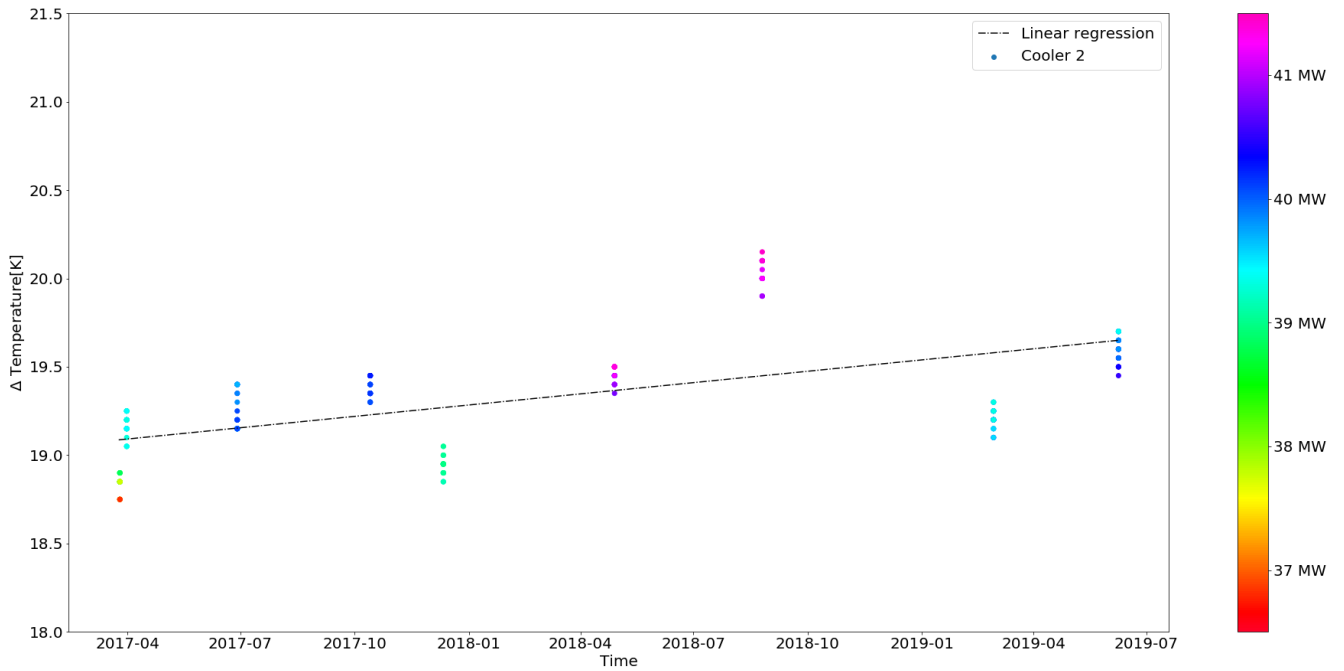


Figure 4.3: Difference in temperature over air cooler 2 for generator 1 and linear regression, and color according to the output power

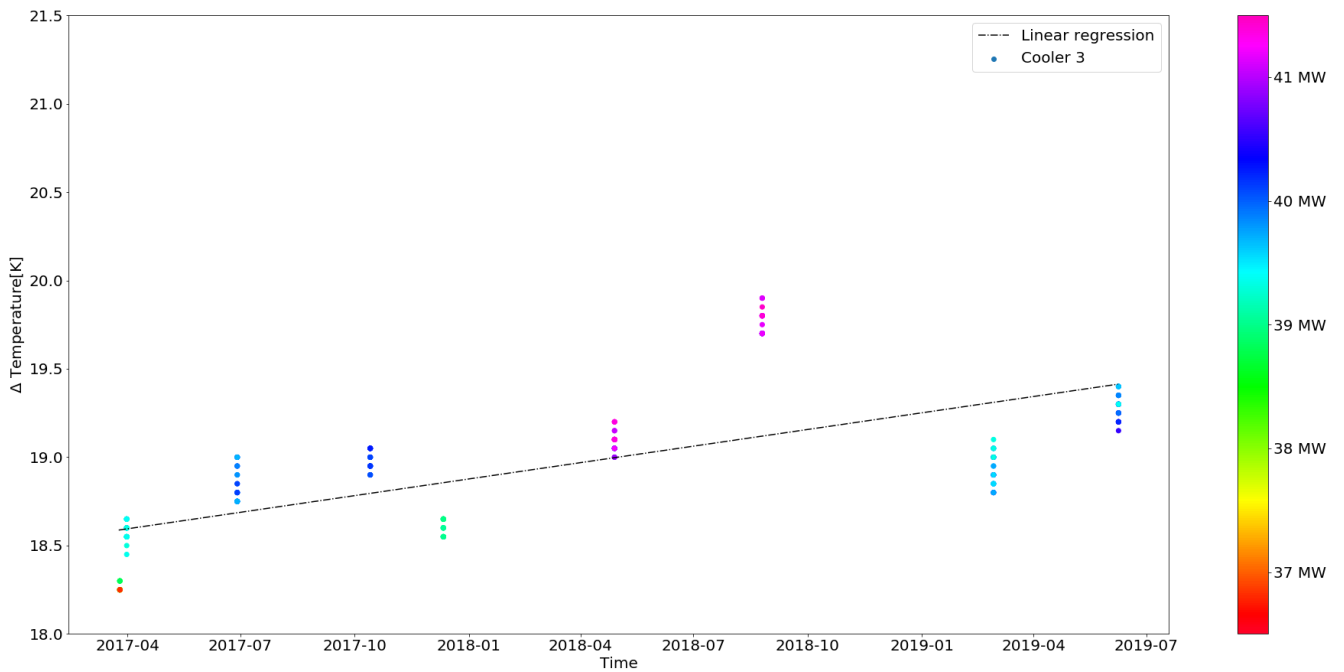


Figure 4.4: Difference in temperature over air cooler 3 for generator 1 and linear regression, and color according to the output power

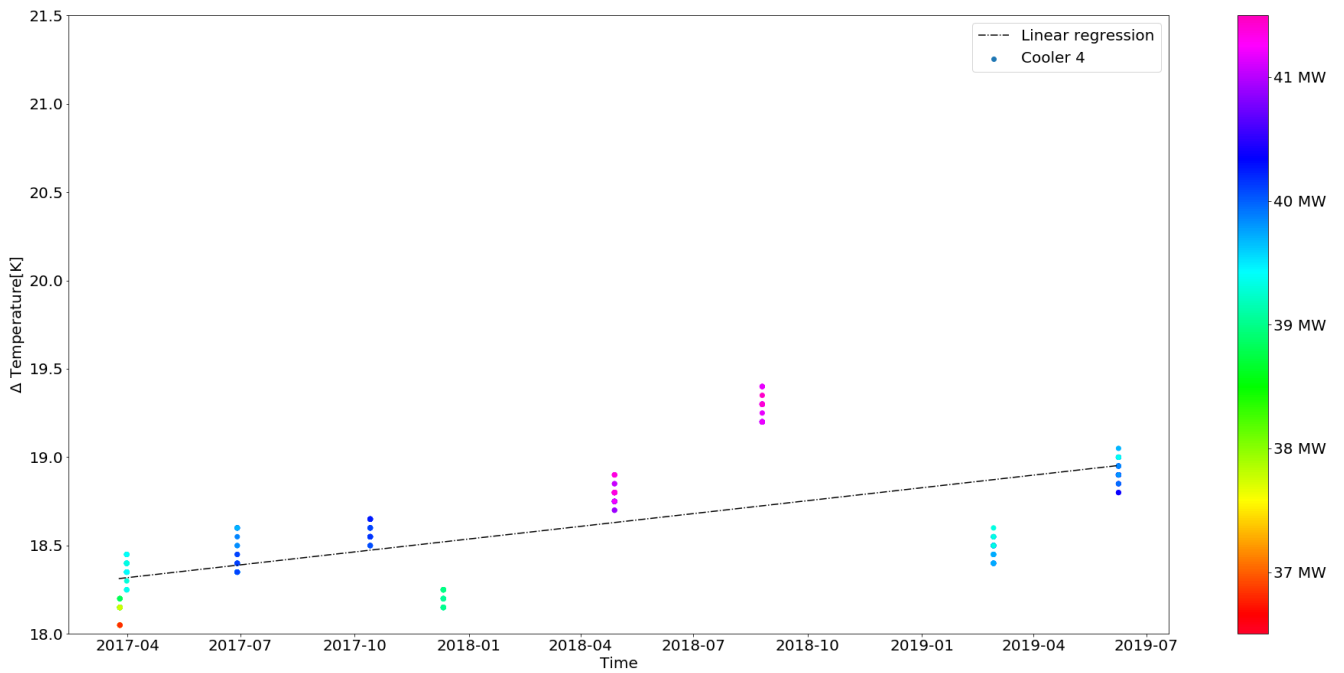


Figure 4.5: Difference in temperature over air cooler 4 for generator 1 and linear regression, and color according to the output power

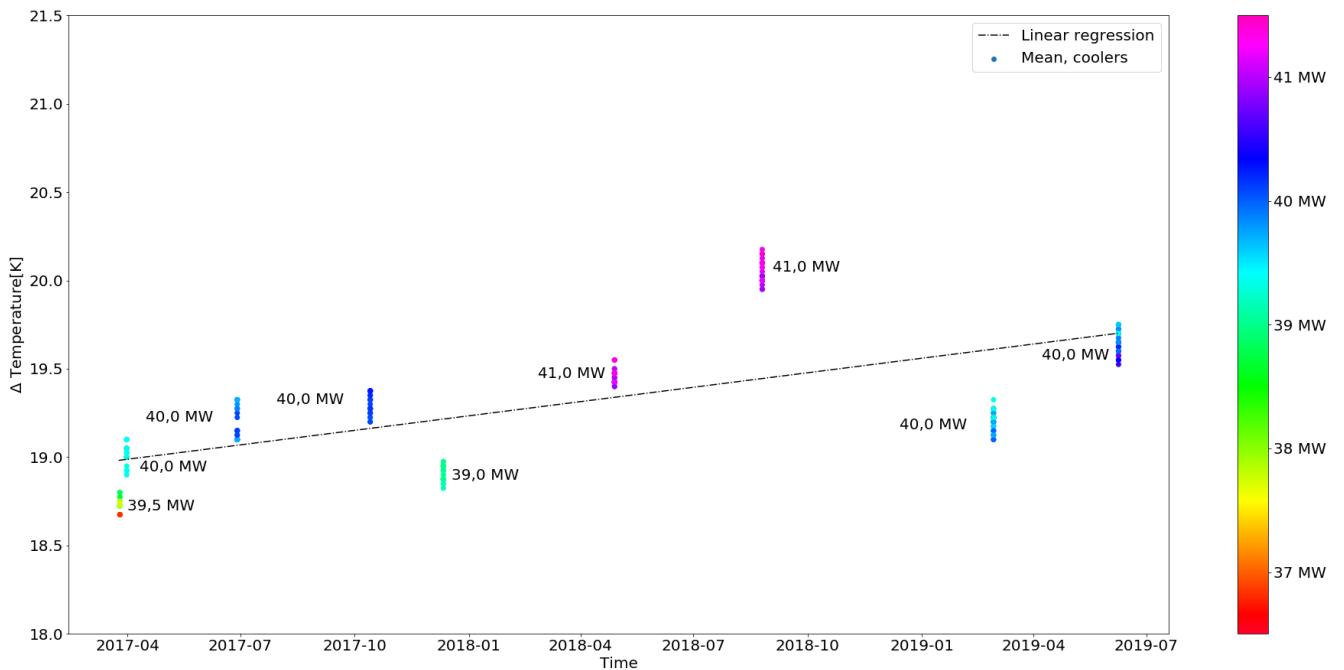


Figure 4.6: Difference in temperature over the air coolers for generator 1, with the mean temperature of the coolers, linear regression and setpoint of the power



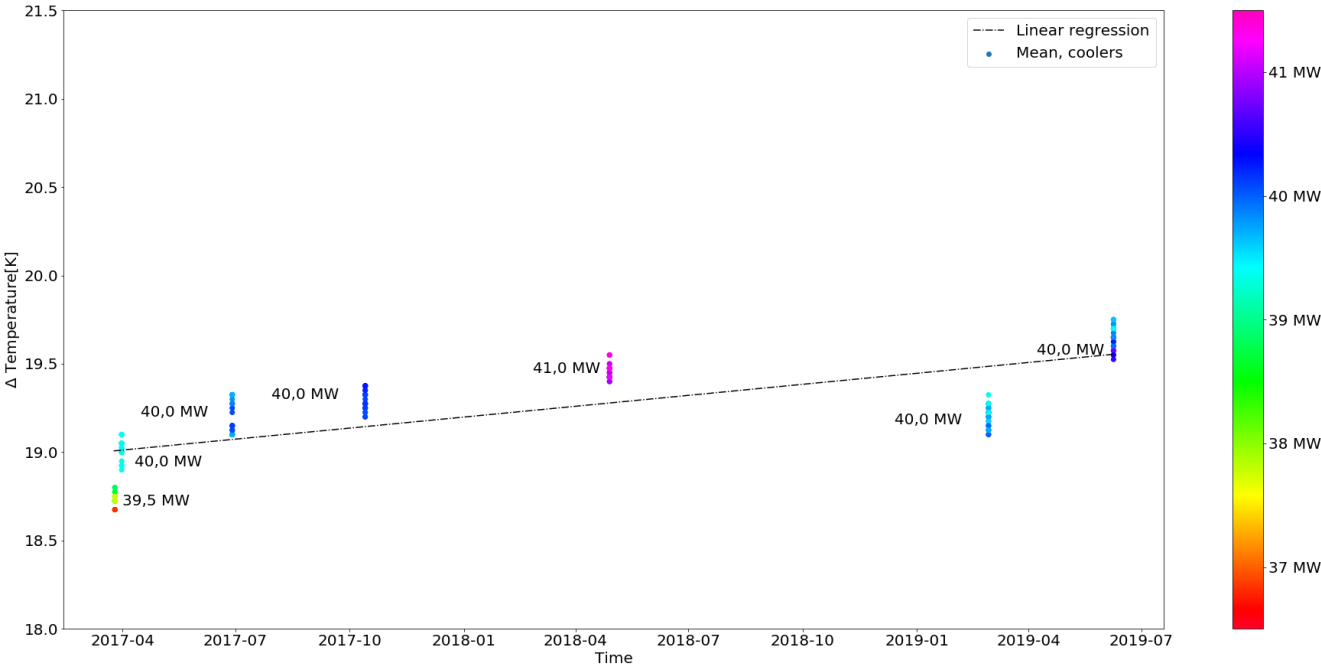


Figure 4.7: Difference in temperature over the air coolers for generator 1, without the two extreme values, with the mean temperature of the coolers, linear regression and setpoint of the power

**Generator 2** Figure 4.8 illustrates the temperature difference for air cooler 1 of generator 2. Compared to the other coolers of generator 2, is the change over time quite steep. Whereas linear regressions for the other coolers are almost horizontal, is there a clear slope here. Figure 4.9 has some slope on the regression line, but not as steep as cooler 1. Figure 4.10, which represent cooler 3, has a significantly lower starting point than the others, and its regression is almost horizontal. Figure 4.11, start at the same point temperature difference as the cooler 1 and 2, but separates from the other by having a horizontal linear regression.

The figure representing the mean of all the coolers for generator 2, figure 4.12, is clearly influenced by all the coolers. Its regression line is not horizontal, neither is as steep as the one for cooler 1. The colors of the scatter reveal that the values that have been used are quite stationary, except for the time series from the end of June 2019, where there might have been some fluctuations. Figure 4.13, which represents the mean without the two extreme values, have a negative slope, which was not expected.

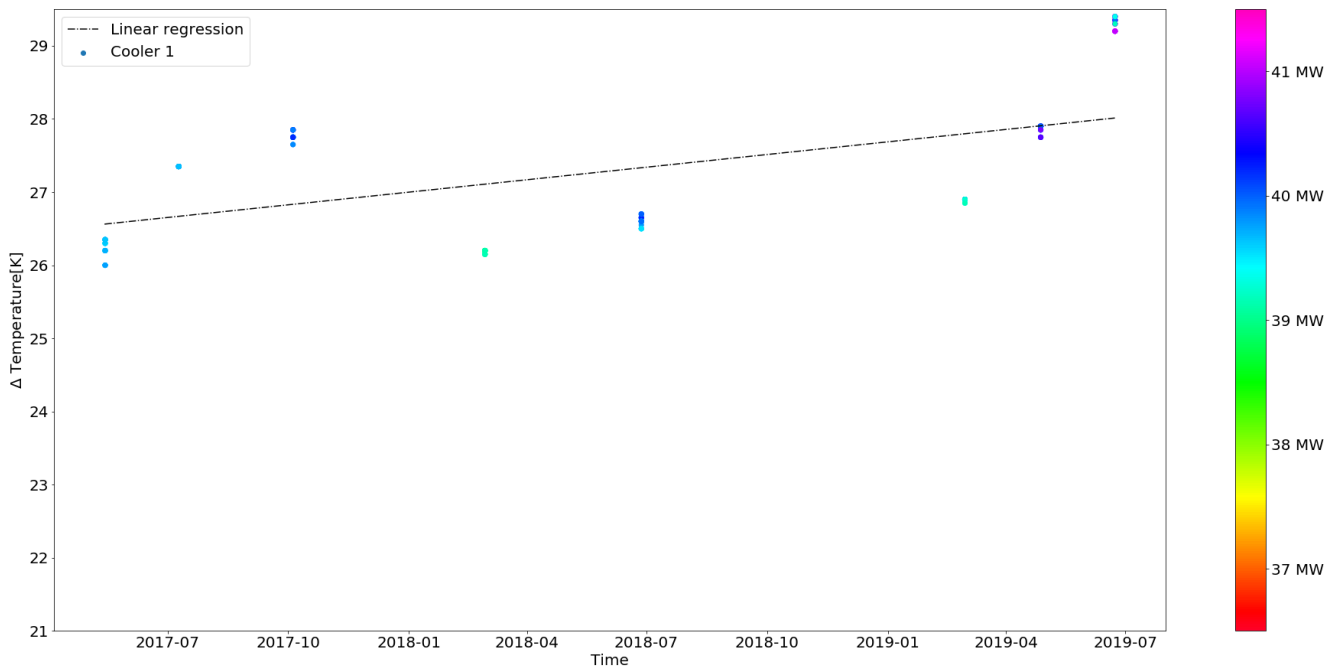


Figure 4.8: Difference in temperature over air cooler 1 for generator 2, linear regression, and color according to the output power

## 4. Results

---

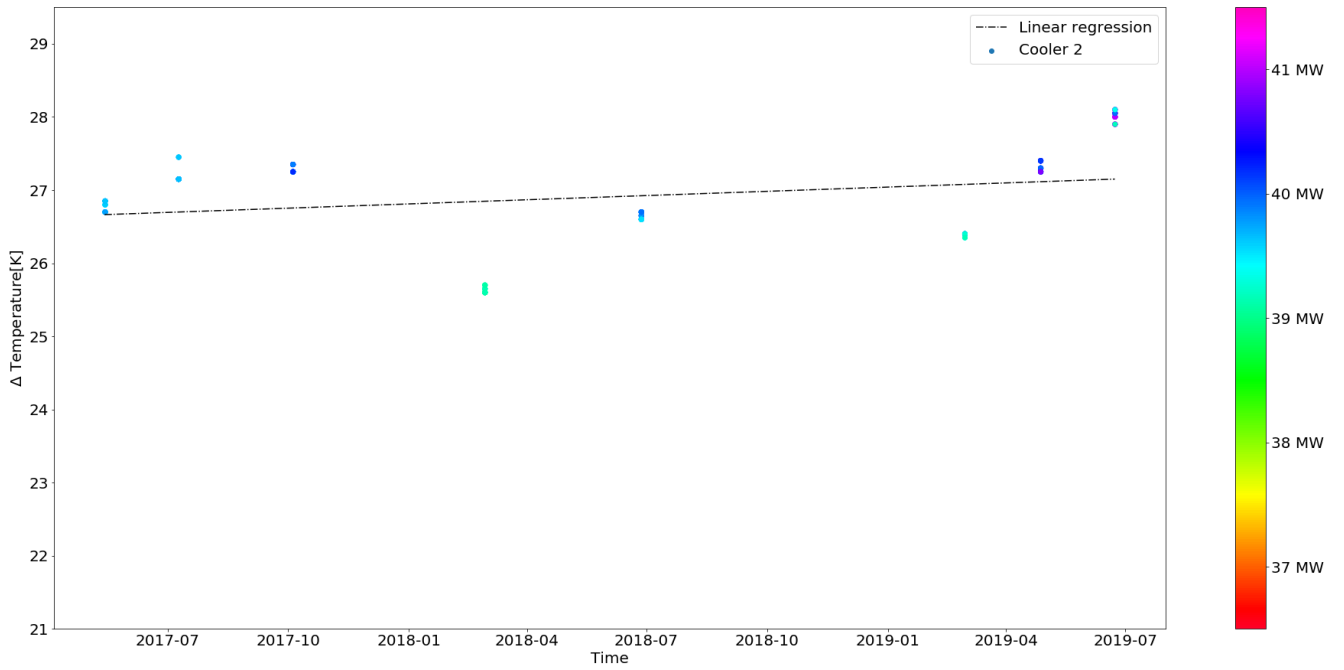


Figure 4.9: Difference in temperature over air cooler 2 for generator 2, linear regression, and color according to the output power

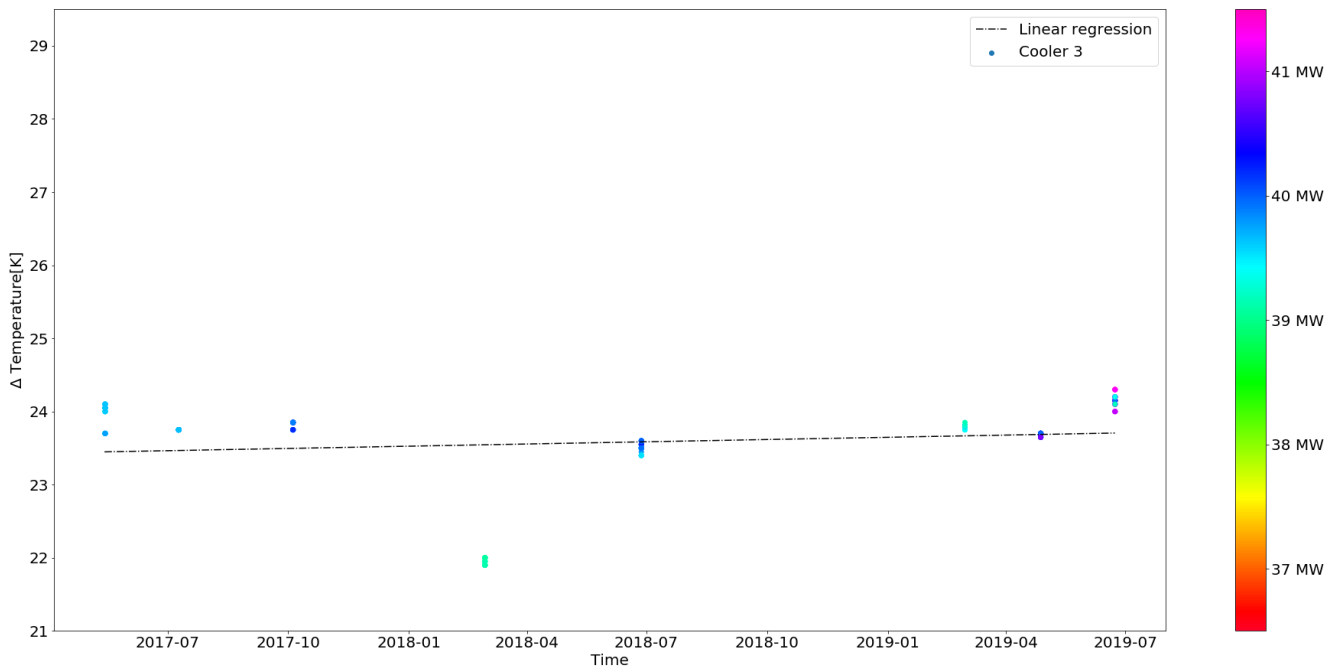


Figure 4.10: Difference in temperature over air cooler 3 for generator 2, linear regression, and color according to the output power

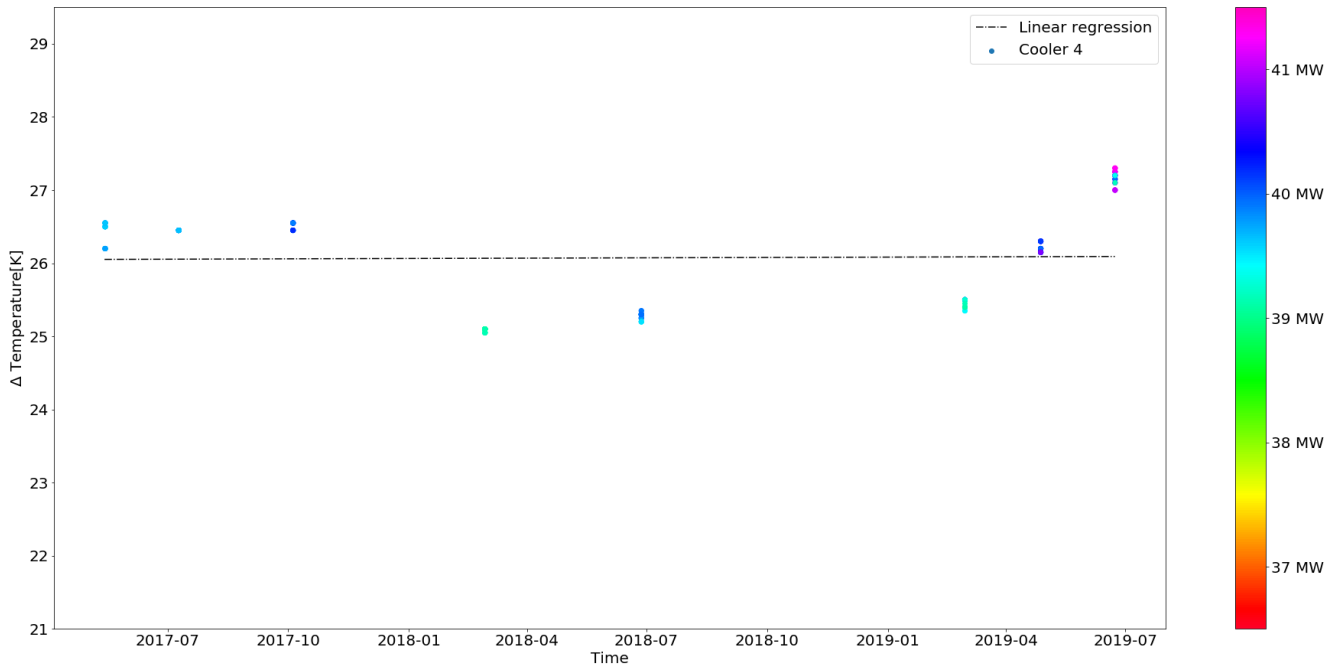


Figure 4.11: Difference in temperature over air cooler 4 for generator 2, linear regression, and color according to the output power

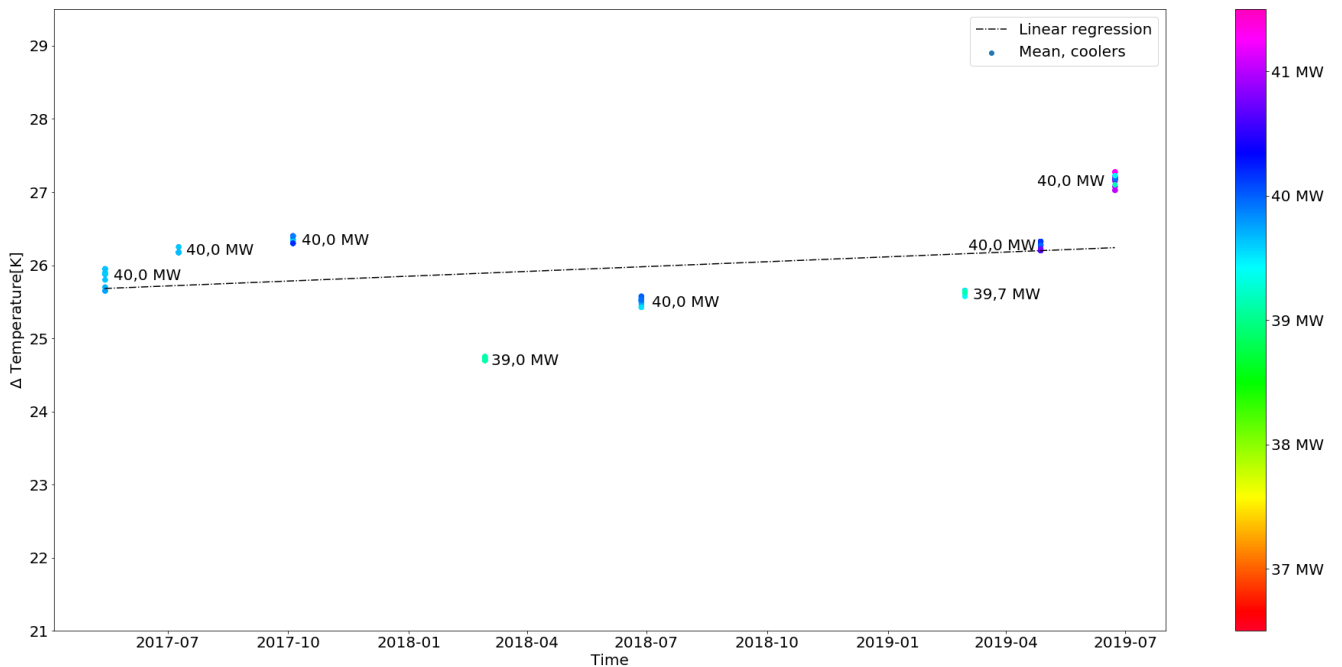


Figure 4.12: Difference in temperature over the air coolers for generator 2, with the mean temperature of the coolers, linear regression and setpoint of the power

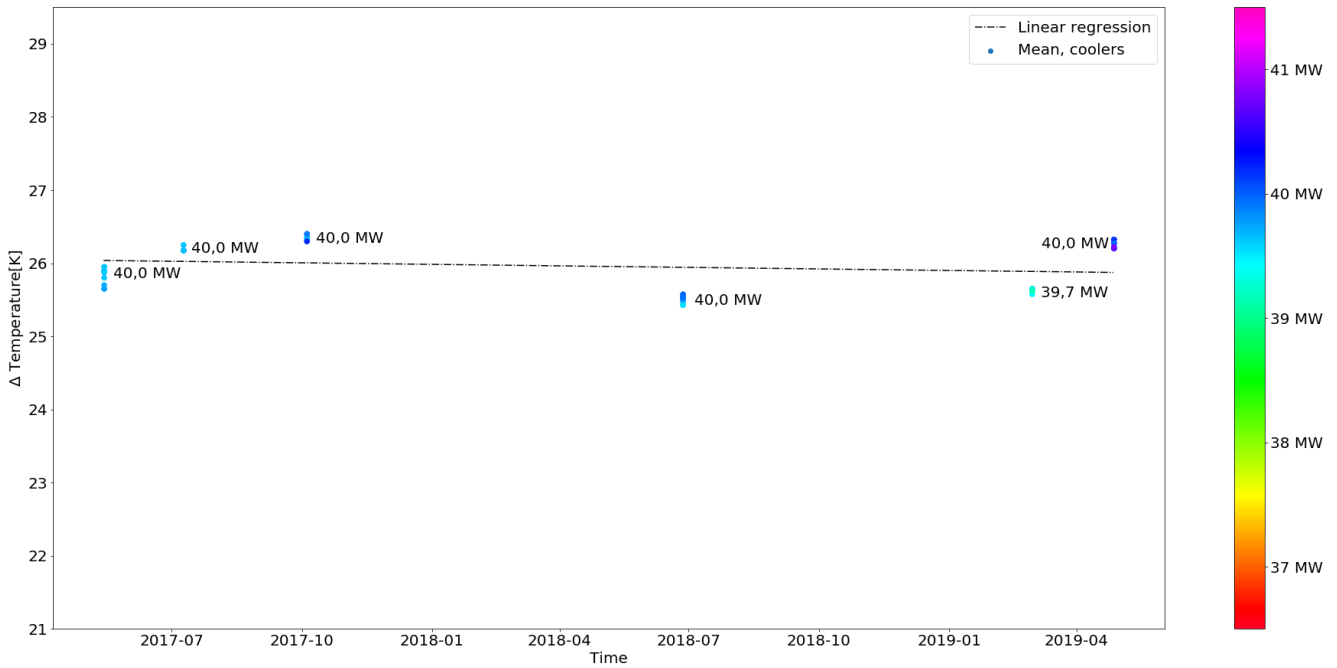


Figure 4.13: Difference in temperature over the air coolers for generator 2, without the two extreme values, with the mean temperature of the coolers, linear regression and setpoint of the power

### 4.3 Scenario 2

Scenario 2's results are given in figure 4.14 and 4.15. The difference in temperature for the air over the coolers has been calculated, and the mean of those calculations are plotted as a scatter for the power output. Their linear regression has been calculated, and an upper and lower threshold, at double standard deviation, have been included. Note also that both the y- and x-axes differ from one figure to another. Time series mentioned in table 3.3, are used for both generators.

There is a clear clustering at between 39,0 and 40,5 MW in figure 4.14. Here, the  $\Delta$  temperature is around 18,5-19,5 K. We can see that most values are within the double standard deviation threshold. In figure 4.15 there is also a cluster, but it is not dense as the one in the previous figure. It is also located at around 39,5-40,0 MW. There are some values located outside the thresholds.

At first sight the slope of the regressions might look similar, but by closer examination of the y-axes it can be determined that figure 4.15 has a steeper inclination than figure 4.14.

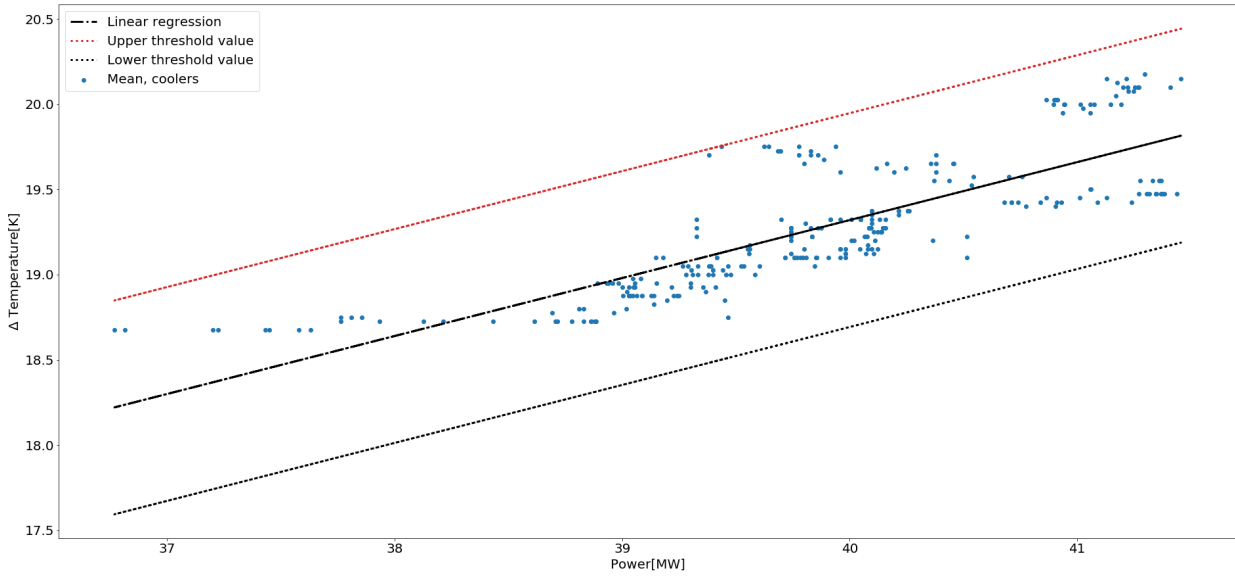


Figure 4.14: Temperature differences at various power outputs for generator 1

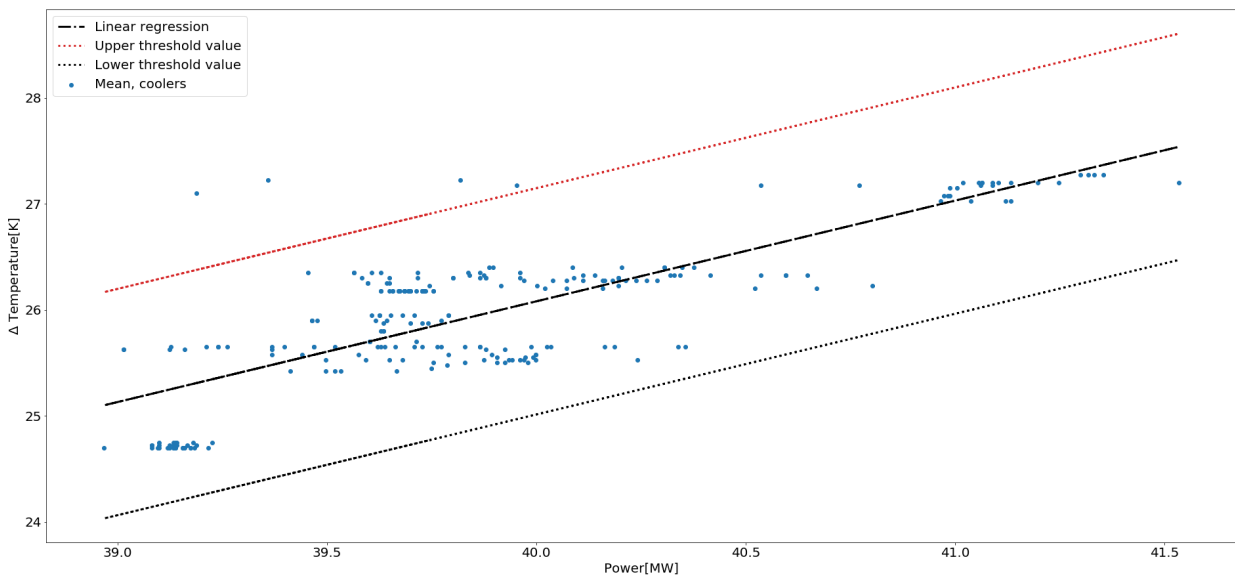


Figure 4.15: Temperature differences at various power outputs for generator 2



## 5 Discussion

### 5.1 Pre-processing

The pre-processing of the data require some data manipulation. Since the SCADA system often samples the measurements upon change, will there most likely be a large number of different sampling frequencies. To be able to compare the data it must have the same sampling frequencies, but to achieve the same frequency must the data be manipulated. It was decided to sample the data in minute intervals and forward-fill the values in between.

Minute intervals were chosen for most analyses. The change in air temperature is a fairly slow process, due to the time constants of the air, and it was assumed that no information would be lost. The forward-fill method was chosen because it resembles the behaviour of the control system, where most values are sampled upon update. Interpolation techniques were also investigated, but gave results that were quite similar. It could however, give other results and potentially fabricate values in between two measurements.

There is a method in the pre-processing that drops any duplicates. The first duplicate is kept, as a real value. It was not uncovered if it is most correct to keep the first or last of the duplicates for the data it was applied to.

### 5.2 Data

After an appropriate time span for export was found, has the quality of the data been very high. Not a single error or outlier has been identified. This indicates that SCADA data is very well suited for analysis. The data used in the analyses were inside, or very close, the double standard deviation of the mean of four measurements. This testifies to good data.

The source of the errors in the quantitative data, should be located and repaired. The errors seemingly had valid time stamps, which made them even harder to identify, but there is a very low chance for the measurements to be correct. The system and the export function is in a trail period, and Voith is aware of the problem with the synthesized data.

The analysis is only performed on historic data. The analyses would have to be re-written to be used on live-data.

A clear weakness is that only data at full load, or close to full load, has been analyzed.

The sensors that measure the temperature of the air may be place at different distances from the outlet of the air coolers, which may introduce systematic faults.

Eidsiva Kraft AS[49] monitors temperatures in hydropower plants. The use data from their operations center, which has a much lower sampling frequency then what is available in the SCADA system. Despite this, they have been able to produce models for condition monitoring.

Analysis of SCADA data is probably quite common, but little information is given online. It is assumed that a lot of companies do it, but that they want to protect their work from



competing businesses. The only code available on GitHub was partly in Chinese.

### 5.3 Uncertainties

The sampling of the data will introduce some uncertainties. The available data gives an accuracy of 0,05%, as stated in section 2.1.4.

The resampling and forward-filling will of course introduces some uncertainties as well.

Other uncertainties may be the seasonal change in the temperature of the cooling water. The ambient temperature in the hydropower plant may also change slightly, but since it is placed into the mountain, it is not very likely.

### 5.4 Scripting

In the Python script is the `iloc`-method used for dynamic and static referencing of the columns and rows in the DataFrames. This method is very convenient, but demands a lot of resources. The run-time of the code, which on average was 1 second, could have been vastly improved by changing referencing method. The largest data set that was assembled used around 17 minutes for the pre-processing. It would therefore be worth to look into other methods of referencing for larger data sets.

### 5.5 Methods and tools

The methods and tools that have been applied are well suited for the tasks analyzing and plotting data. Scripting, and scripts, takes some time to set up and get used to, but for repetitive tasks that involve large quantities of data it is much better suited, than a graphical method like Excel. The analysis and plots that have been developed could surely have been developed in Excel as well, and the two or three iterations would probably finished sooner, but when several analysis are scheduled, scripting is regarded a superior option. The scripting does not require any direct interaction with the data, and the risk of introducing random errors is regarded lower. There can by all means occur errors in a script as well, but the possibility of random errors are lower in a script. The choice of Python as scripting language was a deliberate choice. There are range of other languages that could have been used, where MATLAB is perhaps the strongest contender. It too, offers very good options for plotting and data science. But since Voith was the project owner it was considered easier for them to build on code that is Open source, if they would wish to do so. Many of the great libraries offered by MATLAB are proprietary and requires that the businesses buy access.

### 5.6 Proof of concept

The Proof of Concept illustrates that SCADA data can be used for Condition Monitoring of a generator air cooler. Should there be a rapid change in temperature, due to leakage or

some other unlikely event, there will be a rapid change in the temperature measurement for one of the coolers. For this change to "picked up" by the system it has to be larger than the double standard deviation. Which will give the system some robustness.

It can also be refined, in a way that it requires a particular number of measurements within a specified time to be outside the thresholds, or that it must be outside the thresholds for a specific amount of time or something like that. It is not smart analysis at all, but it could prove useful either way.

It does not handle small changes like growing, which would cause a small (up to 10 K) change in temperature over a long time period. Which does make it as good for condition monitoring as scenario 1, but it could be useful for anomaly detection.

## 5.7 Scenario 1

One important assumption in this scenario is that an increase in temperature difference over a cooler for a given load indicates that the condition of of cooler is declining. The idea behind this theory is that the feedback-controlled controller that handles the adjustable valve that controls the flow to the generator air cooler, will increase the flow of cooling water, if the temperature rises. As algae grow on the water side the of the pipes of the air cooler over time, the the thermal conductivity in the air cooler will decrease. The controller will then increase the flow of cooling water to sustain the same temperatures in the generator. Where the temperature that is fed to the controller is situated is unknown. But the assumption perfectly explain why the air coolers, who are thoroughly rinsed once a year experience an increase in the temperature difference over time. Since we do not have the temperature of the water going in or out of the generator air coolers, the assumption is applied to this scenario.

The plot for the individual coolers clearly show that there are differences between them. If they are compared to the plot of the mean for the coolers it is apparent for generator 1, that cooler 1 is the most effective cooler. Cooler 2 and 3 have somewhat the same efficiency, and 4 has the lowest.

It could have been interesting to compare the coolers with the absolute temperature, not only the temperature difference. It could then function as an additional anomaly system, where deviations for the mean could trigger an alarm.

The mean for generator 1, when it is corrected by removing the extreme values, have almost the exact same slope which could indicate the time series that have been used are quite representative.

For generator 2, are the slopes of the regressions not so steep, some of them are even almost horizontal, which may indicate that their condition is good. It is not clear to say, based on only 8 time series. There are, however, some differences between the coolers. Cooler 1 looks to be the most effective here as well, 2 and 4 are in the middle and 3 is the worst performing. The mean shows that the temperature rises slightly over the period. When it corrected for the extreme values, it has a slightly negative slope, but again, it is only based on eighth time series, and should not be given to much weight.

It looks as though the trend is that the temperature difference over the coolers is that is increasing. Which in that case will confirm the hypothesis given in this section.

### 5.8 Scenario 2

This scenario, or concept is maybe a more fitting term for it, could be useful for anomaly detecting. The results clearly show that there is a nearly linear relationship between the difference in the mean of the temperature difference over the coolers and the power output. Which is to be expected, from the theory on electrical machines. Either way, it proves that the data is fairly consistent, and it could have been used to categorize a value as within what is considered normal or it is an anomaly.

From the plots it is clear that temperature rises more quickly in generator 2, as the power output increases. This may be because the coolers in generator 1 has a higher efficiency.

Reactive power has not been included, and the reactive power may cause the temperature in the machine to rise. Some days were checked for reactive load, and the reactive power was then zero, or very close to zero. But it is not specifically checked on the dates of the exported time series. There is not much data available that give stationary temperatures for lower power outputs. More data could have strengthened the hypotheses and results for this concept. The regressions are only based on 9 time series for generator 1 and 8 for generator 2, which gives the results high uncertainty.

### 5.9 Is it worth it?

Hydropower plants have a very high availability. Eidsiva kraft has an availability of 98,9%[49] on some of their hydropower plants. This indicates that companies are not willing to spend a lot of money on highly specialized solutions for condition monitoring to increase their availability by, perhaps, 0,1%. Any developed solutions can not be to expensive. Given that hydropower plants have a very high reliability it may be better to focus on anomaly detection. It is when the production must be stopped, it gets expensive. Nonetheless, can millions of kroner be saved in maintenance costs from better planning and targeted maintenance[50]. It is also worth mentioning that there is already software available that facilitates for condition monitoring available, like OSI soft Pi.

### 5.10 Why condition monitoring

Condition monitoring may save the hydropower plant operator money, by allowing a component be fixed before it breaks down entirely. Given that it is correctly configured, of course. This also makes sense from an environmental point of view, where fixing always is better than replacing.

Condition monitoring will probably become more popular in the times that, as 5G allows low latency communication wirelessly. The economies of scale will probably also give better and cheaper sensors and supporting systems.

During a data breach, or any other event where outsiders try to influence or affect a system, the intruder's goal may be to physically destroy the target, by sending abnormal parameters that causes it to break down. This was what happened when the Stuxnet virus targeted an Iranian Nuclear Facility, and via its SCADA system successfully accomplished its goal[51]. Condition Monitoring of a system may help it increase its Cyber Security, by detecting and acting on such an anomaly before it takes full effect.[52]

Earlier, it was common to have someone present in the hydropower plant at all times. Today, when most hydropower plants can be remote controlled this is not as common. Thereby, one loses the added security of having someone who knows the plant well present in the plant. Condition monitoring can function as the hydropower plants representative, and if it is given enough data, and is trained well, it will know everything worth knowing about the plant.



## 6 Conclusion and further work

This section will follow up on what was discussed in the previous chapter. It will then draw the conclusions for the project. Ultimately will the outlook and further work of the project be presented.

### 6.1 Conclusion

The project's goal was to investigate the potential for condition monitoring of a generator cooling system in a hydropower plant using physical models on data from a SCADA system. The project succeeded in investigating the potential for condition monitoring, it also found concepts that could be used for condition monitoring of the generator air coolers in a hydropower plant. It does not monitor the whole cooling system, only what was regarded the most important components were selected and using the analyses that were developed the expected degradation patterns could be proven, to some extent.

More data should be gathered to verify trends and patterns that were uncovered, but SCADA data has beyond a doubt been proven useful in the work towards condition monitoring of a generator cooling system in a hydropower plant

The hydropower industry is known for their conservatism, but implementation of digital services like data collection and analysis, might push hydropower plant operators towards investing in more efficient and profitable solutions. To develop more modern and profitable solutions, historic data is needed to predict what will happen in the future, it is there tempting to end with a quote from the famous Danish philosopher Søren Kierkegaard that says: "Life can only be understood backwards; but it must be lived forwards" [53]

### 6.2 Further work

The first thing that would have been addressed in the future work with the project, would be the format of the exported data. Today, there are inconsistencies in the data that is exported, and the format vary a lot. This is not optimal, when one wishes to automate as much of the work as possible.

Further work should also include adding more data to analyses. Especially data at stationary conditions for lower power outputs. More data could improve the accuracy and possibly provide information on the cooling at lower temperatures. If this information were to be combined with the existing data, this could reveal information on the behaviour of the feedback controlled supply of cooling water to the generator air coolers. It could also improve the second scenario that was investigated, where the temperature was predicted based on the power output for high outputs only. Lower output temperature predictions would nuance the data and the predictions making them more robust.

An extension of the work with the thesis could also include investigating the possibility of including some kind of analysis functionality in the SCADA system itself. For example by comparing the temperatures of the cold air from the generator air coolers, or some other

kind of analysis. It could provide some low latency analyses, that have the benefit of being available in the SCADA system itself.

Further more, it should include some verification of the analyses that have been developed. For example by conducting some control measures at the plant, or by verifying that the temperatures will change following the scheduled replacement of the air coolers this fall.

Now that data has been collected for a period of over two years, it might be fruitful to include some Machine learning methods as well. They could prove valuable, especially for anomaly detection, where a Machine Learning model can process a lot of data simultaneously and categorize variables as normal or as anomalies.

Economies of scale are beginning to take effect in the IoT and sensor market. This means that sensors and equipment for monitoring purposes are cheap, and will only get cheaper and become more accessible in the future. This will allow the use Condition Monitoring techniques for vibration and acoustic monitoring, as well. Some experience with Condition Monitoring will probably come in handy, once more types of monitoring is included. The forthcoming introduction of 5G networks, where latency and bandwidth is no issue will probably help pave the way for the fourth industrial revolution, where even generator air coolers are connected to the Internet.[54]

---

## References

- [1] Lauren Feiner. *Amazon passes Microsoft in market value, becomes largest*. <https://www.cnbc.com/2019/01/07/amazon-passes-microsoft-market-value-becomes-largest.html>. (Accessed on 07/01/2019).
- [2] *Industri 4.0 - digitalisering av tradisjonell industri - IKT-Norge*. <https://www.ikt-norge.no/tema/industri-4-0-digitalisering-av-tradisjonell-industri/>. (Accessed on 07/03/2019). IKT-Norge,
- [3] Ruben Ravnå and Per Schjølberg. *Industry 4.0 and Maintenance*. Tech. rep. Brussels: European Committee for Standardization, July 2017.
- [4] Darius Hedgebeth. “Data-driven decision making for the enterprise: an overview of business intelligence applications”. In: *Vine* 37.4 (2007), pp. 414–420.
- [5] *Elektrisitet - årlig - SSB*. <https://www.ssb.no/energi-og-industri/statistikker/elektrisitet/aar>. (Accessed on 07/01/2019). Statistisk sentralbyrå,
- [6] *Tilknytning til Europa - kjøp og salg av kraft - NVE*. <https://www.nve.no/reguleringsmyndigheten-for-energi-rme-marked-og-monopol/engrosmarkedet/tilknytning-til-europa-kjop-og-salg-av-kraft/>. (Accessed on 07/01/2019). Norges vassdrags- og energidirektorat,
- [7] Knut A. Rosvold. *Effektkjøring - kraftverk. I Store norske leksikon*. [https://snl.no/effektkj%C3%B8ring\\_-\\_kraftverk](https://snl.no/effektkj%C3%B8ring_-_kraftverk). (Accessed on 07/01/2019). Store norske leksikon, Dec. 2017.
- [8] *Forskning skal løse sandfangproblem – Siste nytt fra HydroCen*. <https://hydrocen.blog/2018/04/04/forskning-skal-lose-sandfangproblem/>. (Accessed on 07/01/2019). HydroCen,
- [9] Lars Søreide. “Hva betyr endret kjøremønster for maskinparkens levetid?” <https://docplayer.me/16299466-Hva-betyr-endret-kjoremonster-for-maskinparkens-levetid-forum-for-generatorer-18-19-09-2007-lars-soreide-bkk-produksjon.html>. 2007.
- [10] Mette Eltvik. “Sediment erosion in Francis turbines”. PhD thesis. Norwegian University of Science and Technology, 2013.
- [11] Trygve Holtebekk. *SI-systemet – Store norske leksikon*. <https://snl.no/SI-systemet>. (Accessed on 05/10/2019). Store norske leksikon, May 2019.
- [12] *Tall, tid og dato*. <https://www.sprakradet.no/sprakhjelp/Skriveregler/Dato/>. (Accessed on 05/07/2019). Språkrådet,
- [13] Y. A. Çengel and J. M. Cimbala. *Fluid Mechanics: Fundamentals and Applications, Third Edition*. McGraw-Hill, 2014. ISBN: 9781259011221.
- [14] Hermod Brekke. *Pumper og turbiner*. Trondheim: NTNU Vannkraftlaboratoriet, 199. ISBN: 9781259011221.
- [15] Eirik Øgaard. *Francis-turbin - Vasskrafta*. <http://www.vasskrafta.no/turbinar/francis-turbin-article214-479.html>. (Accessed on 02/04/2019). Norsk Vasskraft- og Industristadmuseum,



- [16] Theodore Wildi. *Electrical Machines, Drives and Power Systems: Pearson New International Edition*. 6th. Essex: Pearson Education Limited, 2014. ISBN: 9781292024585.
- [17] Svein Bua, Magnus Dalva, and Olav Vaag Thorsen. *Roterende elektriske maskiner*. 3rd. Universitetsforlaget AS, 1980. ISBN: 82-00-03306-6.
- [18] Bjørn Pedersen. *Varmeveksler – Store norske leksikon*. <https://snl.no/varmeveksler>. (Accessed on 07/04/2019). Store norske leksikon, Oct. 2017.
- [19] *Håndbok i Tilstandsstyrt Vedlikehold, bok nr. 246*. ABB Energi AS, May 1993.
- [20] *Instruction manual for Generator/Motor Cooler QLKE, QDKE, QDKR8*. Modine.
- [21] Jørn Heggset. *Arbeidsnotat: Generator: Skadetyper, kriterier for tilstandsfastlegging, konstruksjonsløsning. Versjon 2*. SINTEF Energiforskning AS, Feb. 2011.
- [22] *Programmable Logic Controllers PLCs — ABB*. <https://new.abb.com/plc/programmable-logic-controllers-plcs>. (Accessed on 07/04/2019). ABB,
- [23] Knut A. Rosvold. *Kontrollanlegg – i kraftforsyningen – Store norske leksikon*. [https://snl.no/kontrollanlegg\\_-\\_i\\_kraftforsyningen](https://snl.no/kontrollanlegg_-_i_kraftforsyningen). (Accessed on 07/05/2019). Store norske leksikon, Feb. 2017.
- [24] Paul Bjørn Andersen. *automatisering – Store norske leksikon*. <https://snl.no/automatisering>. (Accessed on 07/06/2019). Store norske leksikon, Aug. 2018.
- [25] John Codrington. *The Hydro Turbine Governor and Why it is Important to Understand it*. [/http://www.ieee.ca/epec10/codrington1.pdf](http://www.ieee.ca/epec10/codrington1.pdf). (Accessed on 07/05/2019). Hatch Inc,
- [26] Ivar Gunvaldsen. *Bryter – elektrisitetslære – Store norske leksikon*. [https://snl.no/bryter\\_-\\_elektrisitetstl%C3%A6re](https://snl.no/bryter_-_elektrisitetstl%C3%A6re). (Accessed on 07/05/2019). Feb. 2019.
- [27] Paul Bjørn Andersen. *Analog-digital-omformer – Store norske leksikon*. <https://snl.no/analog-digital-omformer>. (Accessed on 07/05/2019). Store norske leksikon, Dec. 2012.
- [28] Ragnar Johnsen. *Digital signalbehandling – Store norske leksikon*. [https://snl.no/digital\\_signalbehandling](https://snl.no/digital_signalbehandling). (Accessed on 07/05/2019). Store norske leksikon, Jan. 2018.
- [29] *Basics of Data Acquisition - eDAQ Wiki*. [https://www.edaq.com/wiki/Basics\\_of\\_Data\\_Acquisition](https://www.edaq.com/wiki/Basics_of_Data_Acquisition). (Accessed on 07/05/2019). eDAQ,
- [30] K. Kim et al. *Use of SCADA Data for Failure Detection in Wind Turbines - 51653.pdf*. <https://www.nrel.gov/docs/fy12osti/51653.pdf>. (Accessed on 07/05/2019). Aug. 11.
- [31] Ragnar Johnsen. *sampling – Store norske leksikon*. <https://snl.no/sampling>. (Accessed on 07/05/2019). Store norske leksikon, Feb. 2018.
- [32] Øyvind Holm. *Praktisk erfaring med uthenting av data fra kontrollanlegg – erfaringer fra Brattset*. [/https://www.energiforsk.se/media/26336/voith\\_brattset\\_holm.pdf](https://www.energiforsk.se/media/26336/voith_brattset_holm.pdf). (Accessed on 07/05/2019). Voith Hydro, May 2019.
- [33] M. Ghosal and V. Rao. “Fusion of Multirate Measurements for Nonlinear Dynamic State Estimation of the Power Systems”. In: *IEEE Transactions on Smart Grid* 10.1 (Jan. 2019), pp. 216–226. ISSN: 1949-3053. DOI: 10.1109/TSG.2017.2737359.
- [34] Miguel A. Sanz-Bobi. “Review of analytics methods supporting Anomaly detection and Condition Based Maintenance”. Dec. 2019.

- 
- [35] Jasmina Obradovic et al. “USING BPMN AND INTOUCH HMI SOFTWARE FOR MODELING OF SAMPLING SYSTEM IN REFINERIES”. English. In: *Metalurgia International* 17.11 (2012). Copyright - Copyright Fundatia Metalurgia Romana F.M.R 2012; Last updated - 2012-10-24, pp. 72–79. URL: <https://search.proquest.com/docview/1114884388?accountid=12870>.
- [36] Kåre Bjørvik and Per Hveem. *Reguleringsteknikk*. 3rd. Trondheim: Kybernetets forlag, 2014. ISBN: 978-82-92986-21-9.
- [37] James Northcote-Green. *Control and automation of electrical power distribution systems*. eng. Boca Raton, Fla, 2007.
- [38] “Preface”. In: *Practical Electrical Network Automation and Communication Systems*. Ed. by Cobus Strauss. Oxford: Newnes, 2003, pp. viii–ix. ISBN: 978-0-7506-5801-0. DOI: <https://doi.org/10.1016/B978-075065801-0/50000-0>. URL: <http://www.sciencedirect.com/science/article/pii/B9780750658010500000>.
- [39] In: *Industrial Process Automation Systems*. Ed. by B.R. Mehta and Y.J. Reddy. Oxford: Butterworth-Heinemann, 2015, p. iii. ISBN: 978-0-12-800939-0. DOI: <https://doi.org/10.1016/B978-0-12-800939-0.00027-9>. URL: <http://www.sciencedirect.com/science/article/pii/B9780128009390000279>.
- [40] *Rotating electrical machines, Part 1: Rating and performance, NEK IEC 60034-1:2017*. <https://www.standard.no/no/Nettbutikk/produktkatalogen/Produktpresentasjon/?ProductID=916243>. (Accessed on 07/06/2019). INTERNATIONAL ELECTROTECHNICAL COMMISSION, May 2017.
- [41] Kjell Dahle. *Kraftverkene i Orkla – Sølv i 25 år*. 3rd. Trondheim, 2008. ISBN: 978-82-303-1105-9.
- [42] *Brattset - TrønderEnergi*. <https://tronderenergi.no/produksjon/kraftverk/brattset>. (Accessed on 04/23/2019). TrønderEnergi,
- [43] Norconsult/NVE. *Veileder i planlegging, bygging og drift av små kraftverk*. Tech. rep. (Accessed on 07/07/2019). 2003.
- [44] Rune Mathisen. *Teknikk og industriell produksjon - Temperaturmålinger - NDLA*. <https://ndla.no/subjects/subject:28/topic:1:105763/topic:1:54000/resource:1:117437>. (Accessed on 07/06/2019). Oct. 2018.
- [45] *Industrial platinum resistance thermometers and platinum temperature sensors, IEC 60751:2008*. <https://www.standard.no/en/PDF/FileDownload/?redir=true&filetype=Pdf&preview=true&item=339942&category=5>. (Accessed on 07/06/2019). INTERNATIONAL ELECTROTECHNICAL COMMISSION,
- [46] Jørn Vatn. *eLæring - PK6021 - Vedlikeholdsoptimalisering*. Tech. rep. (Accessed on 07/07/2019). NTNU, Apr. 2018.
- [47] *Maintenance –Maintenance terminology*. European Standard. Brussels: European Committee for Standardization, July 2017.
- [48] T. M. Welte et al. *MonitorX – Experience from a Norwegian-Swedish research project on industry 4.0 and digitalization applied to fault detection and maintenance of hydropower plants*. Tech. rep. (Accessed on 07/07/2019).
-

- [49] Joakim Gundersen. *Elektronisk tilstandsovervåkning - Digitalisering av teknisk kompetanse og erfaring*. Presentation at Produksjonsteknisk konferanse. (Accessed on 07/09/2019). Eidsiva Vannkraft AS, Mar. 2019.
- [50] Thomas Welte. *Digitalisation can save millions in hydropower maintenance costs - #SINTEFblog*. <https://blog.sintef.com/sintefenergy/electric-power-components/digitalisation-can-save-millions-in-hydropower-maintenance-costs/>. (Accessed on 07/10/2019). May 2017.
- [51] Ralph Langner. “Stuxnet: Dissecting a cyberwarfare weapon”. In: *IEEE Security & Privacy* 9.3 (2011), pp. 49–51.
- [52] Anders Strangstad and Vegard Guttormsen. *213: Cyber.Sec: Anders Strangstad: Cybersikkerhet*. <https://player.fm/series/lorntech/ep-213-cybersec-anders-strangstad-cybersikkerhet5>. LØRN.TECH, Feb. 2019.
- [53] Søren Kierkegaard. “Søren Kierkegaards Skrifter”. In: *Journalen JJ:167* 18 (1843), p. 306.
- [54] Odd Richard Valmot. *5G er en teknologirevolusjon som vil tilby en enorm datahastighet - Tu.no*. <https://www.tu.no/artikler/5g-er-en-teknologirevolusjon-som-vil-tilby-en-enorm-datahastighet/452083>. (Accessed on 07/09/2019). Dec. 2019.

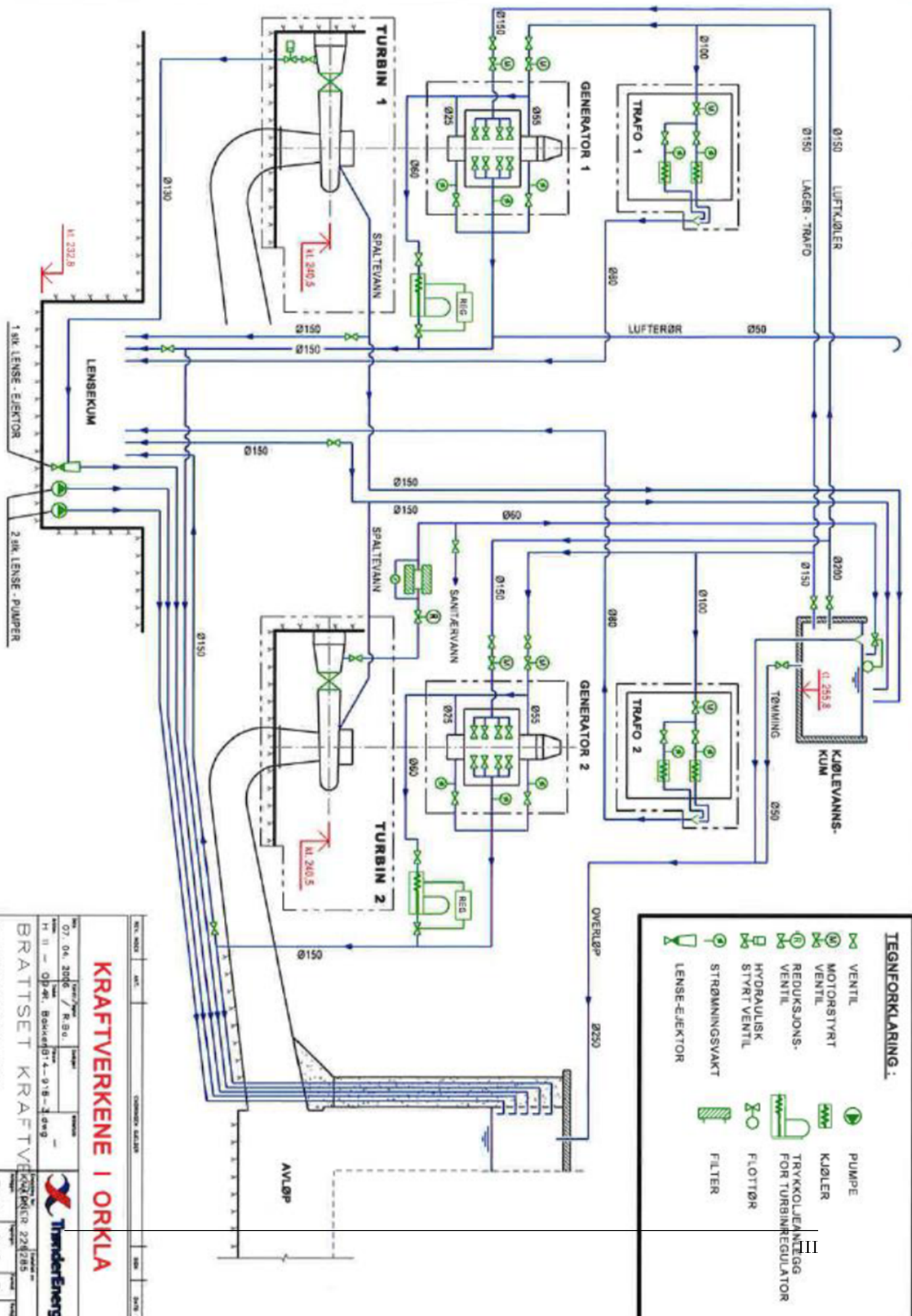
## Appendix A Piping and instrumentation diagrams

Below are two piping and instrumentation diagrams for Brattset's cooling system given. Both are received from TrønderEnergi, who operates the plant. Minor updates and adjustments, as mentioned in a paragraph in section 2.1.5, have been made to the cooling system, without updating the diagrams.












## Appendix B Script for the Proof of Concept plot

This appendix include a Python script that pre-process the CSV-files exported from Voith's Analyzer. Some calculations are performed, and the difference in temperature over the generator air cooler is plotted.

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Fri May 24 10:13:17 2019
4
5  @author: lauvlandm
6
7  PREREQUISITES:
8      The CSV files that are exported from the analyzer tool must use ; as
9      ↪ (field) separator and , as decimal separator
10
11  NOTES:
12      The values in the exported CSV are stored in reversed chronological
13      ↪ order,
14      meaning that the last valid value is placed first.
15
16  SPECIFICS:
17      This script is designed to handle the four cold air measurements that
18      ↪ belong to each generator
19  """
20  import numpy as np
21  import matplotlib.pyplot as plt
22  import matplotlib.dates as mdates
23  import time
24  import pandas as pd
25  import warnings
26  from IPython import get_ipython
27  get_ipython().run_line_magic('matplotlib', 'qt')          #Shows plots in
28  ↪ separate window
29  #get_ipython().run_line_magic('matplotlib', 'inline') #Gives inline
30  ↪ plots
31
32  #Cold air
33  homePath = 'C:/path/'
34
35  #start time
36  t1 = time.perf_counter()
37  #print('Processing started '+ time.asctime())
38
39  #####import data#####
```

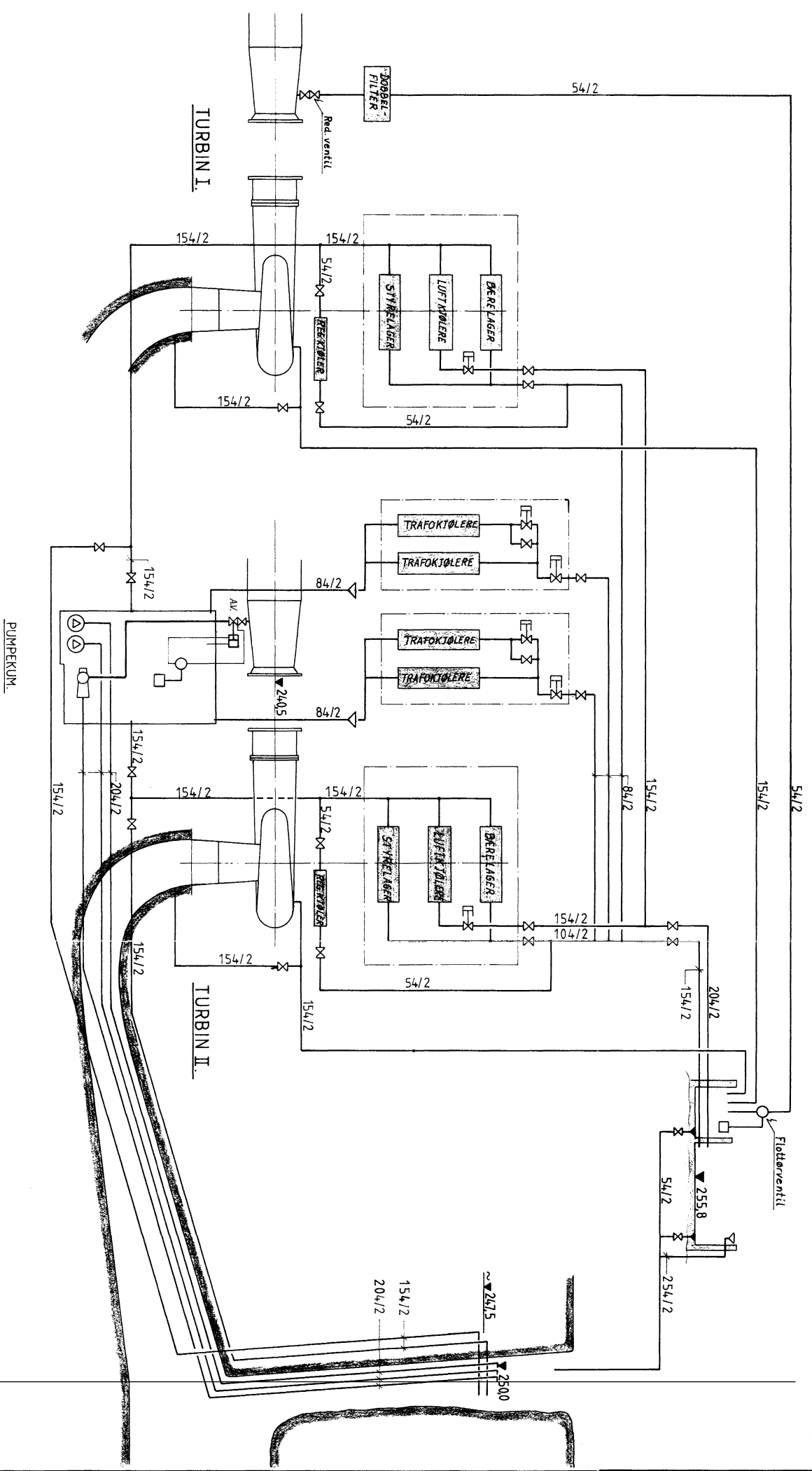


**TEGNEFORKLARING:**

-  VENTIL
-  MOTORSTYRT VENTIL
-  REDUKSJONS-VENTIL
-  HYDRAULISK STYRT VENTIL
-  STRØMNINGSVAKT
-  LENSE-EJEKTØR
-  PUMPE
-  KJØLER
-  TRYKKJØLENTEGG FOR TURBINREGULATOR
-  FLOTØR
-  FILTER

**KRAFTVERKENE I ORKLA**

REV. NODD	REV.	CHANGING SCALE	NO	DATE
<p><b>BRATTSET KRAFTVERK</b></p> <p>Kjølevannsanlegg</p> <p>PRINSSIPSKJEMA</p>				
<p>07. 04. 2006 / R.B.G.</p> <p>H II - Oppt. Bøker Ø14-918 - 1. del</p>		<p>14. 04. 2006 / R.B.G.</p> <p>Kjølevannsanlegg 22/285</p>		
		<p>814 916 3</p>		



Date	23.2.81	Kontroll / Tilsyn	Godkjent	Målestokk	Etablert for	Egnet til
	H.B.					
<b>BRATSET KRAFTVERK</b>						
KJØLEVANNSANLEGG						
816568 - 1						
VVS KLØND A.S. TRONDHEIM						

---

```

35
36 filename = 'Cold air unit 1_05-23-2019_16-17' #Cropped for better view
37
38 filetype = '.csv'
39 path = homePath + filename + filetype           #Seprated into an extra
    ↪ variable to allow re-use of filename
40
41 df = pd.read_csv(path, sep=';', skiprows=1)     #Skips first row, because
    ↪ of meta- data(separator)
42 #df = pd.read_csv(path, sep=';')              #Kun for test, må bruke
    ↪ import over pga meta-data
43
44 df = df.iloc[::-1].reset_index(drop=True)      #Reverses the order of the
    ↪ data to make it chronological and resets the index accordingly
45
46 ##### Resampling and combining measurements in one Data Frame
    ↪ #####
47 """
48 The method creates a large resampled dataframe that consists of the
49 timestamps and measurements for each signal/measurement in the exported
    ↪ CSV-file
50
51 Prerequisites
52 -----
53 CSV-file with one, or more, measurements.
54 The CSV must be separated by a (;), and the delimiter must be(,).
55
56 Parameters
57 -----
58 Dataframe containing imported data.
59
60 Returns
61 -----
62 Pandas Dataframe
63 """
64
65 noColumns = len(df.columns)
    ↪ #Finds the number of columns in the imported data stored in df
66 noMeasurements = int(noColumns/7)
    ↪ #Finds the number of measurements. Each measurement contains 7
    ↪ columns
67
68 appended_data = []
    ↪ #Empty list to store list of dataframes
69

```

---



```

70 j=0
   ↪ #Counter for column access
71 for i in range(noMeasurements):
72     datetime=pd.to_datetime(df.iloc[:,j]).dropna()
   ↪ #Collects timestamp and converts it to python datetime-format and
   ↪ removes NaTs
73     measure_nan = df.iloc[:,j+5].dropna()
   ↪ #Drops NaNs from the data frame
74     measurement = measure_nan.apply(lambda x: x.replace(',',
   ↪ '.',').astype('float') #Collects measurement, replaces , with
   ↪ ., in order to be able to convert it to a float
75     ts = pd.concat([datetime, measurement], axis=1)
   ↪ #Concatenate corresponding timestamp and measurement
76     remove_duplicates = ts.drop_duplicates(subset=['Date ' + str(i+1)],
   ↪ keep='first') #Drops value when there are duplicate timestamps.
   ↪ Keeps the first value
77     drop_first_row = remove_duplicates.drop(remove_duplicates.index[[0]])
   ↪ #The first value often has a very large gap, to ensure that it is
   ↪ not given too much weight it is dropped
78     reindex = drop_first_row.set_index('Date ' + str(i+1))
   ↪ #Re-index to get datetime as index, necessary as index for
   ↪ resampling-method
79     #S is seconds, T is minute. '3T' is 3 minutes, H is hour, D is day
80     # resample = reindex.resample('S').pad()
   ↪ #Second-sampling with forwardfill(resample)
81     resample = reindex.resample('T').pad()
   ↪ #Minute-sampling with forwardfill(resample)
82     # resample = reindex.resample('H').pad()
   ↪ #Hour-sampling with forwardfill(resample)
83     # resample = reindex.resample('D').pad()
   ↪ #Day-sampling with forwardfill(resample)
84     resample_drop_first_row = resample.drop(resample.index[[0]])
   ↪ #If the value provided to the resampling method is second-sampled
   ↪ it will give a NaN on the first value, and be reflected on the
   ↪ second, now minute-sampled value
85     df_index = resample_drop_first_row.reset_index()
   ↪ #Reindexes the dataframe to get it zero-indexed. This is
   ↪ necessary in order to have the same index for all dataframes,
   ↪ which is a requirement for appending
86     appended_data.append(df_index)
   ↪ #Appends the dataframe to a list of dataframes
87     j+=7
   ↪ #Increments the counter with the number of cols per
   ↪ measurement, to gather all the measurements

```

---

```

88 appended_data = pd.concat(appended_data, axis=1)
   ↪ #Concatenate the dataframes to one large dataframe
89 resampled_df = appended_data
   ↪ #Renames the large dataframe by assigning it to a new variable that
   ↪ better explains how the data has been preprocessed
90
91 #### Method for finding the first common timestamp in a dynamic dataframe
   ↪ #####
92 noColumns_complete = len(resampled_df.columns)
   ↪ #Finds the number of columns in the imported data stored in df
93 noMeasurements_complete = int(noColumns_complete/2)
   ↪ #Finds the number of measurements. Each measurement(now) contains 2
   ↪ columns; timestamp and measurement
94
95 ###Method below finds the first common date in the first row, ergo the
   ↪ last starting date
96 k=0
97 dt1 = resampled_df.iloc[0,0]
   ↪ #First timestamp in first column
98 for i in range(noMeasurements_complete-1):
   ↪ #Does the operation for every measurement
99     dt2 = resampled_df.iloc[0, (k+2)]
   ↪ #Second timestamp, located in the third column
100     if dt1 < dt2:
   ↪ #Compares the first and last timestamp to find the largest(most
   ↪ recent)
101         mostRecent = dt2
   ↪ #If dt2 is the largest, it is assigned to the mostRecent
   ↪ variable
102         dt1 = dt2
   ↪ #dt2 is also assigned to the dt1 vairable to compare this
   ↪ value to the timestamp of the next measurement
103     else:
   ↪ #if dt1 is more recent than dt2, the value of dt1 is kept as
   ↪ mostRecent timestamp
104         mostRecent = dt1
105     k += 2
   ↪ #Iterate the K-variable to go to next datetime-column
106
107 ##Method below crops the dataframe
108 cropped_data = []
   ↪ #Empty list to store cropped dataframes
109 m=0
110 for l in range(noMeasurements_complete):
   ↪ # Will perform an iteration for every measurement

```

---

## B. Script for the Proof of Concept plot

---

```
111     res_index = next((i for i, j in enumerate(resampled_df.iloc[:, m]) if
    ↪     j == mostRecent), None) #"Search" each column to find the index
    ↪     for the datetime stored in mostRecent-variable from previous
    ↪     method
112 #Method above is inspired by SilenGhost's reply on Stackoverflow, last
    ↪     visited
    ↪     12.06.19(https://stackoverflow.com/questions/3229626/python-finding-index-of-first-occurrence-of-a-value-in-a-list)
113     datetime_crop = resampled_df.iloc[res_index:, m]
    ↪     # Crops every datetime-column from the top down to the index
    ↪     provided by previous value which is the most recent common date
114     measurement_crop = resampled_df.iloc[res_index:, m+1]
    ↪     # Crops every measurement-column from the top down to the index
    ↪     provided by previous value
115     ts_crop = pd.concat([datetime_crop, measurement_crop], axis=1)
    ↪     #Concatenates the cropped datetime and measurement columns to a
    ↪     timeseries
116     ts_crop = ts_crop.reset_index(drop=True)
    ↪     #Resets the index for the timeseries, since it has been cropped
    ↪     top-down, since last index reset
117     cropped_data.append(ts_crop)
    ↪     #Appends the cropped dataframe to a list of dataframes(together
    ↪     with the ones already cropped)
118     m+=2
    ↪     #Increments the counter with the number of columns per
    ↪     measurement, to gather all the measurements
119     cropped_data = pd.concat(cropped_data, axis=1)
    ↪     #Concatenate the dataframes to one large dataframe
120     cropped_resampled_data = cropped_data.dropna()
    ↪     #Drops any row that contains a NaN(at end of the complete
    ↪     dataframe(due to uneven number of samples for each measurement/uneven
    ↪     length of timeseries))
121
122     ##### Method for removing duplicate timestap columns
    ↪     #####
123     reindexed_cropped_resampled_data =
    ↪     cropped_resampled_data.set_index(cropped_resampled_data.columns[0])
    ↪     #Set one common index for all measurement values
124     reindexed_cropped_resampled_data.index.names = ['Timestamp']
    ↪     #Renames the common index column
125
126     #Finds the number of columns and timestamps for drop of dynamic number of
    ↪     columns
127     noColumns_prd = len(reindexed_cropped_resampled_data.columns)
    ↪     #Finds the number of columns in the imported data stored in df
```

```

128 noTimestamps_prd = int(noColumns_prd/2)
    ↪ #Finds the number of measurements
129
130 #o = noTimestamps_prd+1
    ↪ #For addressing coloumn name in method below
131 o = noColumns_prd-2
    ↪ #For addressing coloumn index in method below
132 for n in range(noTimestamps_prd ):
133 #     preprocessed_reindexed_data.drop(columns=["Date " + str(o)], axis =
    ↪ 1, inplace=True) #For addressing coloumn name
134
    ↪ reindexed_cropped_resampled_data.drop(reindexed_cropped_resampled_data.columns
    ↪ axis = 1, inplace=True) #For addressing coloumn index
135 #     o -= 1 #For addressing coloumn name in method above
136     o -= 2 #For addressing coloumn index in method below
137 df_preprocessed = reindexed_cropped_resampled_data #The preprocessed
    ↪ dataframe, renamed to emphasize that this is the finished dataframe
138 #df_preprocessed.to_excel(filename + 'seconds_preprocessed.xlsx')
    ↪ #Writes the (often very) large dataframe to an Excel-file(Very time
    ↪ consuming)
139
140 #####Analysis#####
141
142 #Mean for all temperatures by row
143 df_preprocessed['Mean'] = df_preprocessed.mean(axis=1)
144 df_preprocessed['Mean+1%'] = df_preprocessed.iloc[:,4]*1.01
145 df_preprocessed['Mean-1%'] = df_preprocessed.iloc[:,4]*0.99
146 df_preprocessed['St.dev. '] = df_preprocessed.iloc[:,4].std(axis=0)
147 df_preprocessed['St.dev.-'] = df_preprocessed.iloc[:,4] -
    ↪ (df_preprocessed.iloc[:,7]*2)
148 df_preprocessed['St.dev.+'] = df_preprocessed.iloc[:,4] +
    ↪ (df_preprocessed.iloc[:,7]*2)
149
150 print('df_preprocessed[St.dev.]')
151 print(df_preprocessed['St.dev.'])
152
153
154 #df_preprocessed.to_excel(filename + 'med bånd.xlsx')
155
156 ##### Plot #####
157 #plt.plot(df_preprocessed)
158
159 ##### Fire plott i "vindu" for å illustrere alarmgrenser??
160 fig, axes = plt.subplots(2,2, sharex=True, sharey=True)
161

```

```
162 x = df_preprocessed.index
163 y1 = df_preprocessed[['Measurement 1']]
164 y2 = df_preprocessed[['Measurement 2']]
165 y3 = df_preprocessed[['Measurement 3']]
166 y4 = df_preprocessed[['Measurement 4']]
167 y5 = df_preprocessed['St.dev.-']
168 y6 = df_preprocessed['St.dev.+']
169 y7 = df_preprocessed['Mean+1%']
170 y8 = df_preprocessed['Mean-1%']
171
172
173 # one plot on each subplot
174 Cold_air_1, = axes[0][0].plot(x,y1, 'C1', label='Cold air 1')
175 axes[0][0].plot(x,y5, 'k:', label='Lower threshold value', linewidth=3)
176 axes[0][0].plot(x,y6, 'C3:', label='Upper threshold value',linewidth=3)
177
178 Cold_air_2, = axes[0][1].plot(x,y2, 'C2', label='Cold air 2')
179 axes[0][1].plot(x,y5, 'k:', label='Lower threshold value', linewidth=3)
180 axes[0][1].plot(x,y6, 'C3:', label='Upper threshold value',linewidth=3)
181
182 Cold_air_3, = axes[1][0].plot(x,y3, 'C4', label='Cold air 3')
183 axes[1][0].plot(x,y5, 'k:', label='Lower threshold value', linewidth=3)
184 axes[1][0].plot(x,y6, 'C3:', label='Upper threshold value',linewidth=3)
185 axes[1][0].set_xlabel('Time')
186
187 Cold_air_4, = axes[1][1].plot(x,y4, 'C5', label='Cold air 4')
188 threshold_1, = axes[1][1].plot(x,y5, 'k:', label='Lower threshold value',
    ↪ linewidth=3)
189 threshold_2, = axes[1][1].plot(x,y6, 'C3:', label='Upper threshold
    ↪ value',linewidth=3)
190 axes[1][1].set_xlabel('Time')
191
192 plt.xticks([])
193 axes[0][0].legend(handles=[Cold_air_1], loc=1)
194 axes[0][1].legend(handles=[Cold_air_2], loc=2)
195 axes[1][0].legend(handles=[Cold_air_3], loc=1)
196 axes[1][1].legend(handles=[Cold_air_4], loc=2)
197
198 fig.legend(handles=[threshold_1, threshold_2], loc=1)
199 plt.rcParams.update({'font.size': 20})
200
201
202 plt.show()
203 plt.rcParams.update({'font.size': 20})
204
```

```
205
206 ##### stop time
    ↪ #####
207 t2 = time.perf_counter()
208 dt = t2-t1
209
210 #print('Processing ended '+ time.asctime())
211 print('Calculating time: ' + str(dt) + ' s')
212
213 warnings.simplefilter('error')
```

## Appendix C Scenario 1 script

This appendix does also include a Python script that pre-process the CSV-files exported from Voith's Analyzer. It presents the same pre-processing as the previous script. Some calculations are performed, mean change in temperatur for 9 time series is plotted with the power output indicated by a rainbow-colored color bar. The set point is added manually.

```
1  # -*- coding: utf-8 -*-
2  """
3  Created on Thu Jun 27 17:03:14 2019
4
5  PREREQUISITES:
6      The CSV files that are exported from the analyzer tool must use ; as
↪ (field) separator and , as decimal separator
7      The exported files must include following signals:
8          Generator_cold_air_1-4
9          Generator_warm_air_1 and 2
10         Power
11         Setpoint_active_power
12
13
14  NOTES:
15      The values in the exported CSV are stored in reversed chronological
↪ order,
16      meaning that the last valid value is placed first.
17      If the file is inspected in Excel it must then be edited to include
↪ the separator
18
19  SPECIFICS:
20      This script is designed to handle the four cold air values
21      combined with the two hot air measurements, as well as the power
22      for each of the generators
23  """
24  import numpy as np
25  import matplotlib.pyplot as plt
26  import matplotlib.dates as mdates
27  import time
28  import pandas as pd
29  from sklearn.linear_model import LinearRegression
30  import datetime as dt
31  import warnings
32  from IPython import get_ipython
33  get_ipython().run_line_magic('matplotlib', 'qt')           #Shows plots in
↪ separate window
```

```

34 #get_ipython().run_line_magic('matplotlib', 'inline') #Gives inline
    ↪ plots
35
36 #Cold and warm air
37 #homePath = 'C:/path/Unit 1/'
38 homePath = 'C:/path/Unit 2/'
39
40 #start time
41 t1 = time.perf_counter()
42 #print('Processing started '+ time.asctime())
43
44 #####import data#####
45 #####                               UNIT 1
46 #filelist = ['Unit 1 scenario 1_25032017', 'Unit 1 scenario 1_31032017',
    ↪ # 'Unit 1 scenario 1_05052017',
47 #           'Unit 1 scenario 1_28062017', 'Unit 1 scenario 1_13102017',
    ↪ 'Unit 1 scenario 1_11122017',
48 #           'Unit 1 scenario 1_28042018', 'Unit 1 scenario 1_25082018',
49 #           'Unit 1 scenario 1_27022019', 'Unit 1 scenario
    ↪ 1_08062019']#Dynamic list of input files
50
51 #####                               UNIT 2
52 filelist = ['Unit 2 scenario 1_14052017', 'Unit 2 scenario 1_09072017',
53            'Unit 2 scenario 1_04102017', 'Unit 2 scenario 1_27022018',
54            'Unit 2 scenario 1_27062018', 'Unit 2 scenario 1_28022019',
55            'Unit 2 scenario 1_27042019', 'Unit 2 scenario 1_23062019']
56
57 appended_data_dates = []
    ↪ #Empty list to store list of dataframes
58
59 for i in range(len(filelist)):
60     filetype = '.csv'
61
62     path = homePath + filelist[i] + filetype           #Seprated into an
    ↪ extra variable to allow re-use of filename
63     print('\n'+filelist[i])
64
65     df = pd.read_csv(path, sep=';', skiprows=1)        #Skips first row,
    ↪ because of meta- data(separator)
66
67     df = df.iloc[::-1].reset_index(drop=True)        #Reverses the order of
    ↪ the data to make it chronological and resets the index
    ↪ accordingly
68

```



```
69     #### Resampling and combining measurements in one Data Frame
    ↪ #####
70     """
71     The method creates a large resampled dataframe that consists of the
72     timestamps and measurements for each signal/measurement in the
    ↪ exported CSV-file
73
74     Prerequisites
    ↪ -----
75     CSV-file with one, or more, measurements.
76     The CSV must be separated by a (;), and the delimiter must be(,).
77
78     Parameters
    ↪ -----
79     Dataframe containing imported data.
80
81     Returns
    ↪ -----
82     Pandas Dataframe
83     """
84
85     noColumns = len(df.columns)
    ↪ #Finds the number of columns in the imported data stored in df
86     noMeasurements = int(noColumns/7)
    ↪ #Finds the number of measurements. Each measurement contains 7
    ↪ columns
87
88     appended_data = []
    ↪ #Empty list to store list of dataframes
89
90     j=0
    ↪ #Counter for column access
91     #noMeasurements=1#FJERNES
92     for i in range(noMeasurements):
93         datetime=pd.to_datetime(df.iloc[:,j]).dropna()
    ↪ #Collects timestamp and converts it to python datetime-format
    ↪ and removes NaTs
94         measure_nan = df.iloc[:,j+5].dropna()
    ↪ #Drops NaNs from the data frame
95         measurement = measure_nan.apply(lambda x: x.replace(',','.',
    ↪ #Collects measurement, replaces ,
    ↪ with ., in order to be able to convert it to a float
96         ts = pd.concat([datetime, measurement], axis=1)
    ↪ #Concatenate corresponding timestamp and measurement
```

```

100     remove_duplicates = ts.drop_duplicates(subset=['Date ' +
        ↪ str(i+1)], keep='first') #Drops value when there are duplicate
        ↪ timestamps. Keeps the first value
101     #
        ↪ remove_duplicates.drop(remove_duplicates.head(2).index,inplace=True)
        ↪ #The first TWD values often have a very large gap, to ensure that
        ↪ it is not given too much weight, it is dropped
102     #
        ↪ remove_duplicates.drop(remove_duplicates.tail(1).index,inplace=True)
        ↪ #The last values often has a very large gap, to ensure that it is
        ↪ not given too much weight, it is dropped
103     reindex = remove_duplicates.set_index('Date ' + str(i+1))
        ↪ #Re-index to get datetime as index, necessary as index for
        ↪ resampling-method
104     # print('reindex')
105     # print(reindex)
106     #S is seconds, T is minute. '3T' is 3 minutes, H is hour, D is
        ↪ day
107     # resample = reindex.resample('S').pad()
        ↪ #Second-sampling with forwardfill(resample)
108     resample = reindex.resample('T').pad()
        ↪ #Minute-sampling with forwardfill(resample)
        ↪ #Day-sampling with forwardfill(resample)
109     resample_drop_first_row = resample.drop(resample.index[[0]])
        ↪ #If the value provided to the resampling method is
        ↪ second-sampled it will give a NaN on the first value, and be
        ↪ reflected on the second, now minute-sampled value
110     df_index = resample_drop_first_row.reset_index()
        ↪ #Reindexes the dataframe to get it zero-indexed. This is
        ↪ necessary in order to have the same index for all dataframes,
        ↪ which is a requirement for appending
111     appended_data.append(df_index)
        ↪ #Appends the dataframe to a list of dataframes
112     j+=7
        ↪ #Increments the counter with the number of columns per
        ↪ measurement, to gather all the measurements
113     appended_data = pd.concat(appended_data, axis=1)
        ↪ #Concatenate the dataframes to one large dataframe
114     resampled_df = appended_data
        ↪ #Renames the large dataframe by assigning it to a new variable
        ↪ that better explains how the data has been preprocessed
115     #print(resampled_df)
116
117

```

```
118     #### Method for finding the first common timestamp in a dynamic  
119     ↳ dataframe #####  
noColumns_complete = len(resampled_df.columns)  
120     ↳ #Finds the number of columns in the imported data stored in df  
noMeasurements_complete = int(noColumns_complete/2)  
121     ↳ #Finds the number of measurements. Each measurement(now) contains  
122     ↳ 2 columns; timestamp and measurement  
  
123     ####Method below finds the first common date in the first row, ergo  
124     ↳ the last starting date  
k=0  
125     dt1 = resampled_df.iloc[0,0]  
126     ↳ #First timestamp in first column  
for i in range(noMeasurements_complete-1):  
127     ↳ #Does the operation for every measurement  
128     dt2 = resampled_df.iloc[0, (k+2)]  
129     ↳ #Second timestamp, located in the third column  
    if dt1 < dt2:  
130     ↳ #Compares the first and last timestamp to find the  
131     ↳ largest(most recent)  
        mostRecent = dt2  
132     ↳ #If dt2 is the largest, it is assigned to the mostRecent  
133     ↳ variable  
        dt1 = dt2  
134     ↳ #dt2 is also assigned to the dt1 vairable to compare this  
135     ↳ value to the timestamp of the next measurement  
    else:  
136     ↳ #if dt1 is more recent than dt2, the value of dt1 is kept as  
137     ↳ mostRecent timestamp  
        mostRecent = dt1  
138     k += 2  
139     ↳ #Iterate the K-variable to go to next datetime-column  
print('Most recent common date: '+ str(mostRecent))  
  
140     ####Method below crops the dataframe  
cropped_data = []  
141     ↳ #Empty list to store cropped dataframes  
m=0  
142     for l in range(noMeasurements_complete):  
143     ↳ # Will perform an iteration for every measurement  
144     res_index = next((i for i, j in enumerate(resampled_df.iloc[:, m])  
145     ↳ if j == mostRecent), None) "Search" each column to find the  
146     ↳ index for the datetime stored in mostRecent-variable from  
147     ↳ previous method
```

---

```

140 #Method above is inspired by SilenGhost's reply on Stackoverflow,
    ↪ last visited
    ↪ 12.06.19(https://stackoverflow.com/questions/3229626/python-
    ↪ finding-index-of-first-non-empty-item-in-a-list/3229644)
141 datetime_crop = resampled_df.iloc[res_index:, m]
    ↪ # Crops every datetime-column from the top down to the index
    ↪ provided by previous value which is the most recent common
    ↪ date
142 measurement_crop = resampled_df.iloc[res_index:, m+1]
    ↪ # Crops every measurement-column from the top down to the
    ↪ index provided by previous value
143 ts_crop = pd.concat([datetime_crop, measurement_crop], axis=1)
    ↪ #Concatenates the cropped datetime and measurement columns to
    ↪ a timeseries
144 ts_crop = ts_crop.reset_index(drop=True)
    ↪ #Resets the index for the timeseries, since it has been
    ↪ cropped top-down, since last index reset
145 cropped_data.append(ts_crop)
    ↪ #Appends the cropped dataframe to a list of
    ↪ dataframes(together with the ones already cropped)
146 m+=2
    ↪ #Increments the counter with the number of cols per
    ↪ measurement, to gather all the measurements
147 cropped_data = pd.concat(cropped_data, axis=1)
    ↪ #Concatenate the dataframes to one large dataframe
148 cropped_resampled_data = cropped_data.dropna()
    ↪ #Drops any row that contains a NaN(at end of the complete
    ↪ dataframe(due to uneven number of samples for each
    ↪ measurement/uneven length of timeseries))
149
150 ### Method for removing duplicate timestap columns #####
151 reindexed_cropped_resampled_data =
    ↪ cropped_resampled_data.set_index(cropped_resampled_data.columns[0])
    ↪ #Set one common index for all measurement values
152 reindexed_cropped_resampled_data.index.names = ['Timestamp']
    ↪ #Renames the common index column
153
154 #Finds the number of columns and timestamps for drop of dynamic
    ↪ number of columns
155 noColumns_prd = len(reindexed_cropped_resampled_data.columns)
    ↪ #Finds the number of columns in the imported data stored in df
156 noTimestamps_prd = int(noColumns_prd/2)
    ↪ #Finds the number of measurements
157

```

---

```

158     #o = noTimestamps_prd+1
        ↪ #For addressing coloumn name in method below
159     o = noColumns_prd-2
        ↪ #For addressing coloumn index in method below
160     for n in range(noTimestamps_prd ):
161         #     preprocessed_reindexed_data.drop(columns=["Date " + str(o)],
        ↪ axis = 1, inplace=True)           #For addressing coloumn
        ↪ name
162         reindexed_cropped_resampled_data.
            ↪ drop(reindexed_cropped_resampled_data.columns[o], axis = 1,
            ↪ inplace=True)           #For addressing coloumn index
163         #     o -= 1     #For addressing coloumn name in method above
164         o -= 2     #For addressing coloumn index in method below
165     df_preprocessed = reindexed_cropped_resampled_data #The preprocessed
        ↪ dataframe, renamed to emphasize that this is the finished
        ↪ dataframe
166
167     #####Analysis #####
168
169     #Lage ein gjennomsnitt av alle verdier på ein rad
170     df_preprocessed['Mean cold air'] =
        ↪ df_preprocessed.iloc[:,0:4].mean(axis=1)
171     df_preprocessed['Mean warm air'] =
        ↪ df_preprocessed.iloc[:,4:6].mean(axis=1)
172     df_preprocessed['Diff. mean'] =
        ↪ df_preprocessed.iloc[:,9]-df_preprocessed.iloc[:,8]
173     df_preprocessed['Cooler 1'] =
        ↪ df_preprocessed.iloc[:,9]-df_preprocessed.iloc[:,0]
174     df_preprocessed['Cooler 2'] =
        ↪ df_preprocessed.iloc[:,9]-df_preprocessed.iloc[:,1]
175     df_preprocessed['Cooler 3'] =
        ↪ df_preprocessed.iloc[:,9]-df_preprocessed.iloc[:,2]
176     df_preprocessed['Cooler 4'] =
        ↪ df_preprocessed.iloc[:,9]-df_preprocessed.iloc[:,3]
177     df_preprocessed['Mean, coolers'] =
        ↪ df_preprocessed.iloc[:,11:15].mean(axis=1)
178
179     #     print(df_preprocessed.dtypes)
180     print('First valid index: ' + str(df_preprocessed.first_valid_index()))
181     print('Last valid index: ' + str(df_preprocessed.last_valid_index()))
182     #     appended_data_dates = []
183     appended_data_dates.append(df_preprocessed)
184     scenario_1 = pd.concat(appended_data_dates, axis=0)
        ↪ #Concatenate the dataframes to one large dataframe
185

```

```

186 #print(scenario_1.dtypes)
187 #scenario_1.to_excel('Scenario_1.xlsx') # Output to Excel-file
188
189 ##### Scatter plot #####
190 plt.rcParams.update({'font.size': 20})
191 #fig, sctr = plt.subplots()
192 #sctr2 = sctr.twinx()
193
194 x1 = scenario_1.index
195 y1 = scenario_1['Mean, coolers']
196 #y1 = scenario_1['Cooler 1']
197 #y1 = scenario_1['Cooler 2']
198 #y1 = scenario_1['Cooler 3']
199 #y1 = scenario_1['Cooler 4']
200 y2 = scenario_1['Measurement 7']
201
202 #plt.set_ylabel('Temperature [K]')
203 plt.ylabel(r'$\Delta$ Temperature[K]')
204 plt.xlabel('Time')
205
206 plt.plot([],[]) #Has to be present to "correct" the plot
207
208 sctr = plt.scatter(x1, y1, c = y2, cmap='gist_rainbow')
209 plt.colorbar(sctr, format='%d MW')
210 plt.clim(36.5,41.5) #Limits for colorbar UNIT 1
211
212 ##Regression
213 x2=x1.map(dt.datetime.toordinal)#converts datetime datatype into a
    ↪ numerical value that can be applied in the regression
214
215 X = x2.values.reshape(-1, 1) # values converts it into a numpy array
216 Y = y1.values.reshape(-1, 1) # -1 means that calculate the dimension of
    ↪ rows, but have 1 column
217
218 linear_regressor = LinearRegression() # create object for the class
219 linear_regressor.fit(X, Y) # perform linear regression
220 Y_pred = linear_regressor.predict(X) # make predictions
221 #print(Y_pred)
222
223 plt.plot(X, Y_pred, 'k-.', label='Linear regression')
224 #plt.ylim(bottom=18, top=21.5) #Limits for y-axis UNIT 1
225 #plt.ylim(bottom=21, top=29.5) #Limits for y-axis UNIT 1
226 plt.legend()
227
228 plt.show()

```

```
229 ##### stop time
    ↪ #####
230 t2 = time.perf_counter()
231 dt = t2-t1
232
233 #print('Processing ended '+ time.asctime())
234 print('Calculating time: ' + str(dt) + ' s')
235
236 warnings.simplefilter('error')
```

## Appendix D Scenario 2 script

This appendix also include a Python script that pre-process the CSV-files exported from Voith's Analyzer. Some calculations are performed, and the difference in temperature over the generator air cooler is plotted as a function of its power output.

```

1  # -*- coding: utf-8 -*-
2  """
3  Created on Thu Jun 27 17:03:14 2019
4
5  PREREQUISITES:
6      The CSV files that are exported from the analyzer tool must use ; as
   ↪ (field) separator and , as decimal separator
7      The exported files must include following signals:
8          Generator_cold_air_1-4
9          Generator_warm_air_1 and 2
10         Power
11         Setpoint_active_power
12
13  NOTES:
14      The values in the exported CSV are stored in reversed chronological
   ↪ order,
15      meaning that the last valid value is placed first.
16      If the file is inspected in Excel it must then be edited to include
   ↪ the separator
17
18  SPECIFICS:
19      This script is designed to handle the four cold air values
20      combined with the two hot air measurements, as well as the power
21      for each of the generators
22  """
23  import numpy as np
24  import matplotlib.pyplot as plt
25  import matplotlib.dates as mdates
26  import time
27  import pandas as pd
28  from sklearn import datasets, linear_model
29  from sklearn.metrics import mean_squared_error, r2_score
30  import datetime as dt
31  import warnings
32  from IPython import get_ipython
33  get_ipython().run_line_magic('matplotlib', 'qt')           #Shows plots in
   ↪ separate window
34  #get_ipython().run_line_magic('matplotlib', 'inline') #Gives inline
   ↪ plots

```



## D. Scenario 2 script

---

```
35
36 #Cold and warm air
37 homePath = 'C:/path/Unit 1/'
38 #homePath = 'C:/path/Unit 2/'
39
40 #start time
41 t1 = time.perf_counter()
42 #print('Processing started '+ time.asctime())
43
44 #####import data#####
45 ##### UNIT 1
46 filelist = ['Unit 1 scenario 1_25032017', 'Unit 1 scenario 1_31032017', #
47 ↪ 'Unit 1 scenario 1_05052017',
48 ↪ 'Unit 1 scenario 1_28062017', 'Unit 1 scenario 1_13102017',
49 ↪ 'Unit 1 scenario 1_11122017',
50 ↪ 'Unit 1 scenario 1_28042018', 'Unit 1 scenario 1_25082018',
51 ↪ 'Unit 1 scenario 1_27022019', 'Unit 1 scenario
52 ↪ 1_08062019'] #Dynamic list of input files
53
54 ##### UNIT 2
55 #filelist = ['Unit 2 scenario 1_14052017', 'Unit 2 scenario 1_09072017',
56 # 'Unit 2 scenario 1_04102017', 'Unit 2 scenario 1_27022018',
57 # 'Unit 2 scenario 1_27062018', 'Unit 2 scenario 1_28022019',
58 # 'Unit 2 scenario 1_27042019', 'Unit 2 scenario 1_23062019']
59
60 appended_data_dates = []
61 ↪ #Empty list to store list of dataframes
62
63 for i in range(len(filelist)):
64     filetype = '.csv'
65
66     path = homePath + filelist[i] + filetype #Seprated into an
67     ↪ extra variable to allow re-use of filename
68     print('\n'+filelist[i])
69
70     df = pd.read_csv(path, sep=';', skiprows=1) #Skips first row,
71     ↪ because of meta- data(separator)
72
73     df = df.iloc[::-1].reset_index(drop=True) #Reverses the order of
74     ↪ the data to make it chronological and resets the index
75     ↪ accordingly
76
77     ## Resampling and combining measurements in one Data Frame #####
78     """
79     The method creates a large resampled dataframe that consists of the
```

---

```

72     timestamps and measurements for each signal/measurement in the
↪ exported CSV-file
73
74     Prerequisites
75     -----
76     CSV-file with one, or more, measurements.
77     The CSV must be separated by a (;), and the delimiter must be(,).
78
79     Parameters
80     -----
81     Dataframe containing imported data.
82
83     Returns
84     -----
85     Pandas Dataframe
86     """
87
88     noColumns = len(df.columns)
89     ↪ #Finds the number of columns in the imported data stored in df
noMeasurements = int(noColumns/7)
90     ↪ #Finds the number of measurements. Each measurement contains 7
91     ↪ columns
92
93     appended_data = []
94     ↪ #Empty list to store list of dataframes
95
96     j=0
97     ↪ #Counter for column access
98     #noMeasurements=1#FJERNES
99     for i in range(noMeasurements):
100         datetime=pd.to_datetime(df.iloc[:,j]).dropna()
            ↪ #Collects timestamp and converts it to python datetime-format
            ↪ and removes NaTs
        measure_nan = df.iloc[:,j+5].dropna()
            ↪ #Drops NaNs from the data frame
        measurement = measure_nan.apply(lambda x: x.replace(',',
            ↪ '.')).astype('float')    #Collects measurement, replaces ,
            ↪ with ., in order to be able to convert it to a float
        ts = pd.concat([datetime, measurement], axis=1)
            ↪ #Concatenate corresponding timestamp and measurement
        remove_duplicates = ts.drop_duplicates(subset=['Date ' +
            ↪ str(i+1)], keep='first') #Drops value when there are duplicate
            ↪ timestamps. Keeps the first value

```

---

```
101     reindex = remove_duplicates.set_index('Date ' + str(i+1))
      ↪ #Re-index to get datetime as index, necessary as index for
      ↪ resampling-method
102     #S is seconds, T is minute. '3T' is 3 minutes, H is hour, D is
      ↪ day
103     # resample = reindex.resample('S').pad()
      ↪ #Second-sampling with forwardfill(resample)
104     resample = reindex.resample('T').pad()
      ↪ #Minute-sampling with forwardfill(resample)
105     # resample = reindex.resample('H').pad()
      ↪ #Hour-sampling with forwardfill(resample)
106     # resample = reindex.resample('D').pad()
      ↪ #Day-sampling with forwardfill(resample)
107     resample_drop_first_row = resample.drop(resample.index[[0]])
      ↪ #If the value provided to the resampling method is
      ↪ second-sampled it will give a NaN on the first value, and be
      ↪ reflected on the second, now minute-sampled value
108     df_index = resample_drop_first_row.reset_index()
      ↪ #Reindexes the dataframe to get it zero-indexed. This is
      ↪ necessary in order to have the same index for all dataframes,
      ↪ which is a requirement for appending
109     appended_data.append(df_index)
      ↪ #Appends the dataframe to a list of dataframes
110     j+=7
      ↪ #Increments the counter with the number of cols per
      ↪ measurement, to gather all the measurements
111     appended_data = pd.concat(appended_data, axis=1)
      ↪ #Concatenate the dataframes to one large dataframe
112     resampled_df = appended_data
      ↪ #Renames the large dataframe by assigning it to a new variable
      ↪ that better explains how the data has been preprocessed
113     #print(resampled_df)
114
115
116 # Method for finding the first common timestamp in a dynamic dataframe
      ↪ ##
117     noColumns_complete = len(resampled_df.columns)
      ↪ #Finds the number of columns in the imported data stored in df
118     noMeasurements_complete = int(noColumns_complete/2)
      ↪ #Finds the number of measurements. Each measurement(now) contains
      ↪ 2 columns; timestamp and measurement
119
120     ###Method below finds the first common date in the first row, ergo
      ↪ the last starting date
121     k=0
```

```

122 dt1 = resampled_df.iloc[0,0]
    ↪ #First timestamp in first column
123 for i in range(noMeasurements_complete-1):
    ↪ #Does the operation for every measurement
124 dt2 = resampled_df.iloc[0, (k+2)]
    ↪ #Second timestamp, located in the third column
125 if dt1 < dt2:
    ↪ #Compares the first and last timestamp to find the
    ↪ largest(most recent)
126     mostRecent = dt2
    ↪ #If dt2 is the largest, it is assigned to the mostRecent
    ↪ variable
127     dt1 = dt2
    ↪ #dt2 is also assigned to the dt1 variable to compare this
    ↪ value to the timestamp of the next measurement
128 else:
    ↪ #if dt1 is more recent than dt2, the value of dt1 is kept as
    ↪ mostRecent timestamp
129     mostRecent = dt1
130 k += 2
    ↪ #Iterate the K-variable to go to next datetime-column
131 print('Most recent common date: ' + str(mostRecent))
132
133 ##Method below crops the dataframe
134 cropped_data = []
    ↪ #Empty list to store cropped dataframes
135 m=0
136 for l in range(noMeasurements_complete):
    ↪ # Will perform an iteration for every measurement
137     res_index = next((i for i, j in enumerate(resampled_df.iloc[:, m])
    ↪ if j == mostRecent), None) "Search" each column to find the
    ↪ index for the datetime stored in mostRecent-variable from
    ↪ previous method
138 #Method above is inspired by SilenGhost's reply on Stackoverflow,
    ↪ last visited
    ↪ 12.06.19(https://stackoverflow.com/questions/3229626/python-finding-index-of
139 datetime_crop = resampled_df.iloc[res_index:, m]
    ↪ # Crops every datetime-column from the top down to the index
    ↪ provided by previous value which is the most recent common
    ↪ date
140 measurement_crop = resampled_df.iloc[res_index:, m+1]
    ↪ # Crops every measurement-column from the top down to the
    ↪ index provided by previous value

```

```
141     ts_crop = pd.concat([datetime_crop, measurement_crop], axis=1)
      ↪ #Concatenates the cropped datetime and measurement columns to
      ↪ a timeseries
142     ts_crop = ts_crop.reset_index(drop=True)
      ↪ #Resets the index for the timeseries, since it has been
      ↪ cropped top-down, since last index reset
143     cropped_data.append(ts_crop)
      ↪ #Appends the cropped dataframe to a list of
      ↪ dataframes(together with the ones already cropped)
144     m+=2
      ↪ #Increments the counter with the number of cols per
      ↪ measurement, to gather all the measurements
145     cropped_data = pd.concat(cropped_data, axis=1)
      ↪ #Concatenate the dataframes to one large dataframe
146     cropped_resampled_data = cropped_data.dropna()
      ↪ #Drops any row that contains a NaN(at end of the complete
      ↪ dataframe(due to uneven number of samples for each
      ↪ measurement/uneven length of timeseries))
147
148     #print(cropped_resampled_data)
149
150     ##### Method for removing duplicate timestamp columns
      ↪ #####
151     reindexed_cropped_resampled_data =
      ↪ cropped_resampled_data.set_index(cropped_resampled_data.columns[0])
      ↪ #Set one common index for all measurement values
152     reindexed_cropped_resampled_data.index.names = ['Timestamp']
      ↪ #Renames the common index column
153
154     #Finds the number of columns and timestamps for drop of dynamic
      ↪ number of columns
155     noColumns_prd = len(reindexed_cropped_resampled_data.columns)
      ↪ #Finds the number of columns in the imported data stored in df
156     noTimestamps_prd = int(noColumns_prd/2)
      ↪ #Finds the number of measurements
157
158     #o = noTimestamps_prd+1 For addressing column name in method below
159     o = noColumns_prd-2 #For addressing column index in method below
160     for n in range(noTimestamps_prd ):
161         #     preprocessed_reindexed_data.drop(columns=["Date " + str(o)],
      ↪ axis = 1, inplace=True) #For addressing column
      ↪ name
162
      ↪ reindexed_cropped_resampled_data.drop(reindexed_cropped_resampled_data.col
      ↪ axis = 1, inplace=True) #For addressing column index
```

```

163 # o -= 1 #For addressing coloumn name in method above
164 o -= 2 #For addressing coloumn index in method below
165 df_preprocessed = reindexed_cropped_resampled_data #The preprocessed
↳ dataframe, renamed to emphasize that this is the finished
↳ dataframe
166 #df_preprocessed.to_excel(filename + 'seconds_preprocessed.xlsx')
↳ #Writes the (often very) large dataframe to an Excel-file(Very
↳ time consuming)
167
168 #####Analysis #####
169
170 #Lage ein gjennomsnitt av alle verdier på ein rad
171 df_preprocessed['Mean cold air'] =
↳ df_preprocessed.iloc[:,0:4].mean(axis=1)
172 df_preprocessed['Mean warm air'] =
↳ df_preprocessed.iloc[:,4:6].mean(axis=1)
173 df_preprocessed['Diff. mean'] =
↳ df_preprocessed.iloc[:,9]-df_preprocessed.iloc[:,8]
174 df_preprocessed['Cooler 1'] =
↳ df_preprocessed.iloc[:,9]-df_preprocessed.iloc[:,0]
175 df_preprocessed['Cooler 2'] =
↳ df_preprocessed.iloc[:,9]-df_preprocessed.iloc[:,1]
176 df_preprocessed['Cooler 3'] =
↳ df_preprocessed.iloc[:,9]-df_preprocessed.iloc[:,2]
177 df_preprocessed['Cooler 4'] =
↳ df_preprocessed.iloc[:,9]-df_preprocessed.iloc[:,3]
178 df_preprocessed['Mean, coolers'] =
↳ df_preprocessed.iloc[:,11:15].mean(axis=1)
179 print('First valid index: '+ str(df_preprocessed.first_valid_index()))
180 print('Last valid index: '+ str(df_preprocessed.last_valid_index()))
181 #End here
182 # appended_data_dates = []
183 appended_data_dates.append(df_preprocessed)
184 scenario_1 = pd.concat(appended_data_dates, axis=0)
↳ #Concatenate the dataframes to one large dataframe
185
186 #print(scenario_1.dtypes)
187 #scenario_1.to_excel('Scenario_1.xlsx') # Output to Excel-file
188
189 ##### Scatter plot
↳ #####
190 plt.rcParams.update({'font.size': 20})
191
192 #x1 = scenario_1.index
193 y = scenario_1['Mean, coolers']

```

```
194 #y = scenario_1['Mean cold air']
195 #y = scenario_1['Mean warm air']
196 x = scenario_1['Measurement 7']
197
198 #My regression, inspired by
   ↪ https://www.edvancer.in/step-step-guide-to-execute-linear-regression-python/
199 x_reg = np.array(x)
200 y_reg = np.array(y)
201
202 ax = (((np.mean(x_reg)*np.mean(y_reg))-np.mean(x_reg*y_reg))/
203        ((np.mean(x_reg)*np.mean(x_reg))-np.mean(x_reg*x_reg)))
204 ax = round(ax,2)
205 b = (np.mean(y_reg) - np.mean(x_reg)*ax)
206 b =round(b,2)
207 print(ax)
208 print(b)
209
210 reg_line = [(ax*x_reg)+b for x_reg in x]
211
212 regressionlabel = ('Linear regression: ' + str(ax)+'x + ' + str(b))
213 std = np.std(reg_line)
214 #print('STD: ' + str(std))
215
216 plt.ylabel(r'$\Delta$ Temperature[K]')
217 plt.xlabel('Power [MW]')
218
219 plt.scatter(x, y)
220 plt.plot(x, reg_line, 'k-.', label = 'Linear regression', linewidth=3)
221 plt.plot(x, (reg_line + (std*2)), 'k:', label = 'Upper threshold value',
   ↪ linewidth=3)
222 plt.plot(x, (reg_line - (std*2)), 'k:', label = 'Lower threshold value',
   ↪ linewidth=3)
223
224 plt.legend()
225 plt.show()
226
227 ##### stop time #####
228 t2 = time.perf_counter()
229 dtime = t2-t1
230
231 #print('Processing ended ' + time.asctime())
232 print('Calculating time: ' + str(dtime) + ' s')
233
234 warnings.simplefilter('error')
```

