Automated multilinear parameter identification from big data of buildings

Master of Science Thesis

Matthias Daniel Escrivá García

May 19, 2022

Master's thesis submitted as part of the master's examination
in the master program Renewable Energy Systems -
Environmental and Process Engineering
at the Department of Environmental Engineering,
in the Faculty of Life Sciences
at University of Applied Sciences Hamburg

1. Supervisor : Prof. Dr. Gerwald Lichtenberg
2. Supervisor : Dr.-Ing. Thorsten Müller-Eping

Submitted on May 19, 2022

Abstract

This master thesis proposes a new automated process for the multilinear parameter identification for anomaly detection in buildings. This is useful for buildings that work with big data since the amount of data points is in the order of thousands. A data grouping algorithm based on Regular Expressions is built from scratch in order to complement the low-rank parameter identification algorithm made a priori. A building that relies on big data located in Hamburg, Germany, provides a large amount of data points that are used in this thesis as a case study to create the new automated process, which will rely on as less human action as possible. Furthermore, in order to design this process, the data point grouping algorithm relies on an existing standardized scheme for data point tagging.

Keywords: Parameter identification, Big data, Building systems, Data grouping, Regular expressions

Table of Contents

List of Figures

List of Tables

Acknowledgement

I would like to express my gratitude to Prof. Dr. Lichtenberg for supporting me and giving me advice throughout the whole year to do this master thesis. My deepest thanks to Leona Schnelle too, who has supported me throughout the whole making of my thesis.

I am very grateful to my family and friends who have also supported me during the year, specially my parents that have helped me achieve my objective of studying in Germany.

I would also like to thank my supervisors and readers from the HAW Hamburg, to take the time to read and grade my master thesis, which I consider a great achievement.

Finally, I am grateful to the people from the company that has provided the real data to work with. They have also been a great help for the research.

Hamburg,
May 19, 2022

Matthias Daniel Escrivá García

Chapter 1

Introduction

This Chapter serves as an introduction to the background, the object and the scope of this master thesis, as well as the structure of this document, where all the research and data has been recorded.

1-1 Background

Research project SONDE (Supervision and Optimization of New Buildings with Data Exploration) [1] uses parameter identification methods using low rank CP MTI (Canonical Polyadic Multilinear Time-Invariant) models to detect faults in buildings.

This master thesis comes from the necessity of analysing the big amount of measurement data coming from a building that relies on big data to work, situated in Hamburg, Germany. A MATLAB algorithm for CP tensor-based MTI parameter identification in buildings has been created, but only tested in a simple and small testing room [2].

Moreover, the necessity of a normalized data point naming scheme has increased over the years since buildings are relying more and more on big data to control the different processes that take place to generate the necessary indoor comfort, specially in big and multi-use buildings such as office and conference buildings and shopping centers. These buildings usually have thousands of data points that generate data at the same time, and it is difficult to keep track of all the information, specially when it is not identified correctly.

1-2 Object of thesis

The object of this thesis is to create an automated method to perform the multilinear parameter identification from big data of buildings. This method should rely on one algorithm that can be run once and performs both the retrieving of the data points from the building and the parameter identification of the data for fault detection. In order to do so, a new algorithm shall be created in conjunction with the already-existing low rank CP MTI parameter identification algorithm.

1-3 Scope of thesis

The purpose of this Section is to identify the scope of this master thesis. The retrieving of the data will be done by a new algorithm built as part of this thesis, while the parameter identification is done by an already-existing MATLAB algorithm developed by [2].

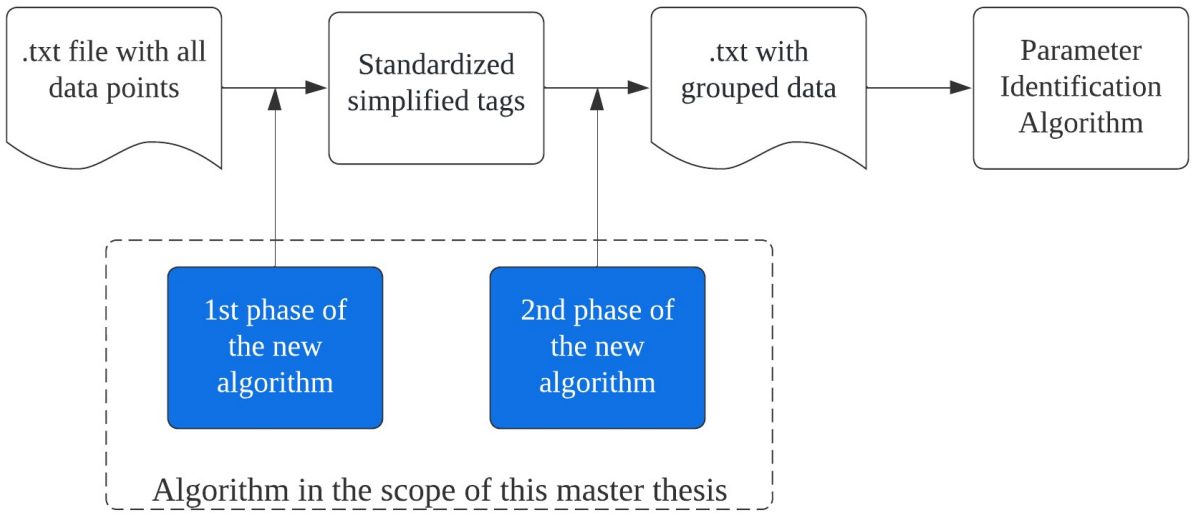

Figure 1-1: Basic scheme of the algorithm in the scope of this master thesis

To better understand the scope, Fig. 1-1 shows a simple scheme of the whole process. As shown in this scheme, the algorithm to which the scope is limited is composed by two phases: the translation of the original tags from the server into a standardized data point naming scheme, and the grouping of the information into an .txt file to use as an input into the parameter identification algorithm. Of course, the process is to be automated as much as possible, as indicated in the object of this thesis.

1-4 Structure of this document

In this Section, a summary of the structure used in this document is described. The document is structured into 6 Chapters, including this one as the Introduction.

Chapter 2 gathers a summary of the State of the Art so far in the field of Building Automation and Control Systems (BACS), which are the origin of the real data gathered for this thesis. Furthermore, an overview on multilinear models in building systems and the low-rank multilinear model that is used in this thesis for parameter identification are presented. Lastly, a review on a standardized method of labelling data points, known as the BUDO scheme, is shown, together with an explanation of what string pattern matching consists of, making an introduction to Regular Expressions which are used in this thesis to create the new algorithm.

After describing the State of the Art of all the items in matter, the new automated process created in this thesis is presented in Chapter 3. Moreover, the case study building is described together with the studied data points to which the BUDO scheme is applied manually.

Chapters 4 and 5 have the purpose to extend the details of the new automated process, explaining how the algorithms work and how they are coupled together. Chapter 6 offers the conclusions and further improvements and research that could be done.

Chapter 2

State of the art

This Chapter's purpose is to present the State of the Art so far that is associated to this master thesis' object of creating an automated parameter identification process. This includes the Building Automation and Control Systems, which are the origin of the data obtained from the case study building; the multilinear models and parameter identification applied to building systems, which are used for fault detection; and the BUDO standardized scheme for data point naming, which will also prove useful when applying the Regular Expressions that are also mentioned in this Chapter.

2-1 Data acquisition in buildings

The increasing complexity and amount of data provided by building automation and control systems has forced the investigation and development of new methods to evaluate large amounts of data, as well as to detect faults. SONDE project, led by Prof. Dr. Dirk Jacob and Prof. Dr. Gerwald Lichtenberg focuses on this matter [1]. A short introduction about BACS is made in this Section, to explain the origin from which the data used in this project is collected.

2-1-1 Building Automation and Control Systems

According to the standard ISO 16484-2, a Building Automation and Control System or BACS is defined as follows:

> "system, comprising all products and engineering services for automatic controls (including interlocks), monitoring, optimization, for operation, human intervention, and management to achieve energy-efficient, economical, and safe operation of building services"

BACS are used in Europe for mechanical and HVAC systems that are referenced in the standard ISO 16484, which includes 5 parts and is aimed to provide assistance to achieve an acceptable environment indoors, as well as energy efficiency, for the design of new buildings and improvement of old buildings. Modern BACS are based on solid-state devices and PLCs, and are also referred to as Home and Building Electronic Systems (HBES) [3].

2-1-2 Components of the BACS

The basic components of a BACS include: sensors, controllers, output devices, data communication protocols and dashboard [4]. Fig. 2-1 shows an overview of how these components are organized, which is interesting to better understand how the BACS and data acquisition works. Of course, the components of a BACS are not limited to the ones appearing in this diagram.

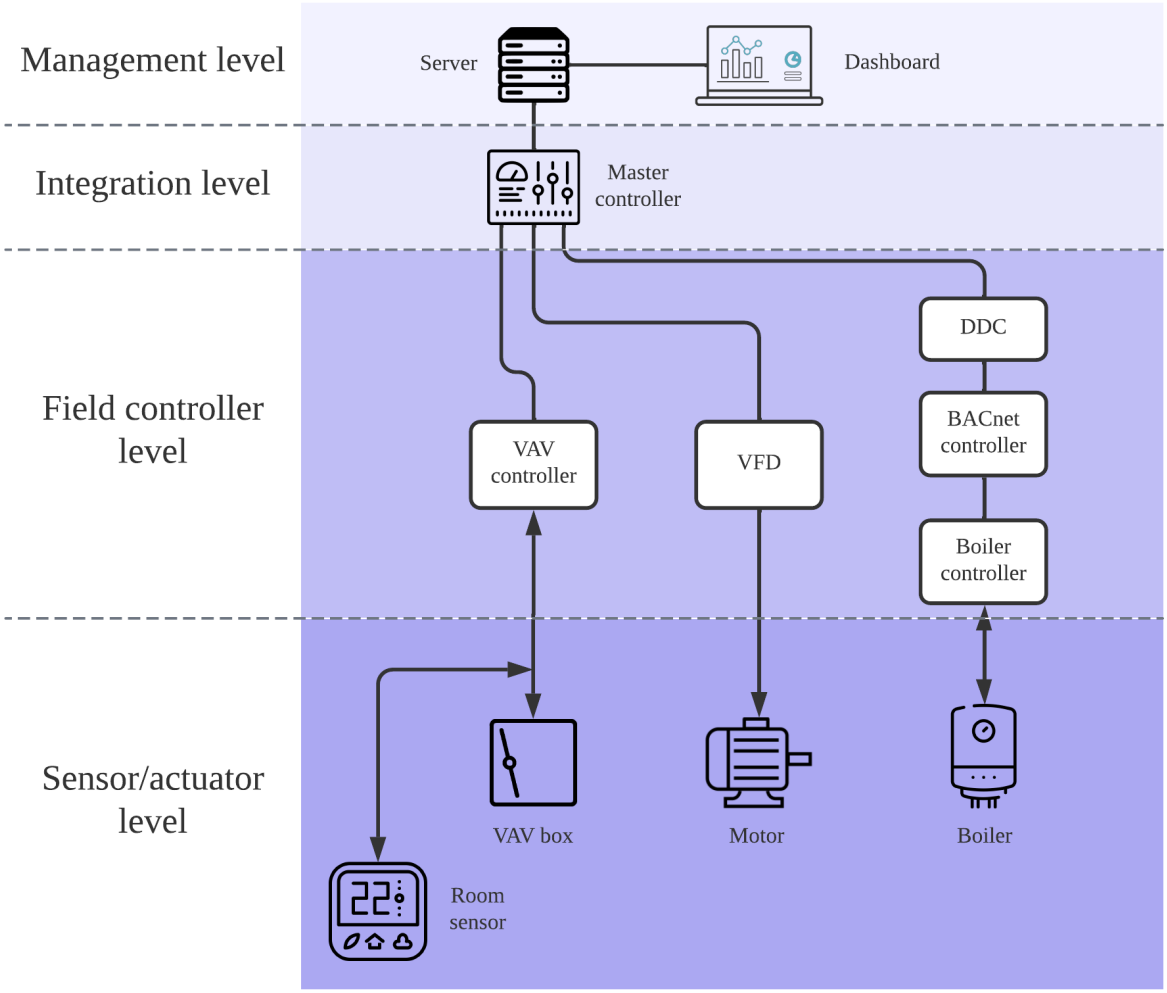

Figure 2-1: BACS components and levels diagram [5]

Sensors and actuators are situated in the first level, and their purpose is to gather the required data directly from the building or to adjust the behaviour of the different components. This means that sensors/actuators have to be situated in the building, together with the controllers that integrate the field level, which are connected physically to the sensors/actuators in order to monitor and control them.

Sensors/actuators and controllers provide a set of data points that go from analog outputs from sensors (such as temperature, air volume flow or air quality, among others) to set-points that are used by the controllers in order to monitor and modify parameters such as the position of variable air volume (VAV) boxes or the frequency at which a motor is working, among many others.

All of these data points can be monitored and controlled from the server in the management level, which sends orders to the master controller that acts as an intermediary between the server and all of the controllers. The server includes an user-interface from which the operator

can easily monitor and control the BACS. All the data acquisition for this thesis is done through the management level.

Data communication protocols are defined in ISO 16484-5, which refers to BACnet, the ASHRAE's standard (refer to ASHRAE 135-2016) for a Data Communication Protocol for Building Automation and Control Networks. Data communication protocols are a set of rules that apply to the exchange of data in computer networks. Even though the BACnet protocol defines rules such as fan operating schedules or pump status alarms among many others, it does not provide a standard way of naming data points, which is a problem when buildings rely on big data to work (see Section 2-3-1).

2-2 Multilinear models in building systems

The increasing amount of data produced by buildings is making the systems more complex, and thus increasing the amount of faults [6]. Building systems as well as heating systems, behave as multilinear [7]. Multilinear Time-Invariant (MTI) models can be represented using tensors [8], the elements of which increase exponentially with higher orders. Therefore, decomposition methods are to be used so as tu reduce the computation time. Canonical Polyadic (CP) decomposition is a good example [8], and it is appropriate for MTI models [9].

It is also important to note that automation systems, described in Section 2-1-1, provide data using a fixed sampling time, which is why discrete time MTI models are used for this application [7].

Taking all of this into account, and as proposed by [2], a CP-decomposed, normalized rank-1 MTI model for parameter identification is used in this thesis for anomaly detection. Depending on the complexity of the system, this may be enough to capture all the dynamics.

2-2-1 CP-decomposed normalized rank-1 MTI model

The simplified rank-1 MTI model offers a computation time several levels lower than higher-rank MTI models. Firstly, as an introduction to the explanation of this simplified model, discrete-time MTI models are summarised [2]. A discrete-time MTI model is given by

$$\mathbf{x}(k+1) = \langle\, \mathsf{F} \mid \mathsf{M}(\mathbf{x}(k), \mathbf{u}(k)) \,\rangle\,, \tag{2-1}$$

where $\mathsf{M}(\mathbf{x}, \mathbf{u})$ is a monomial tensor

$$\mathsf{M}(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} 1 \\ u_m \end{pmatrix} \circ \cdots \circ \begin{pmatrix} 1 \\ u_1 \end{pmatrix} \circ \begin{pmatrix} 1 \\ x_n \end{pmatrix} \circ \cdots \circ \begin{pmatrix} 1 \\ x_1 \end{pmatrix}, \tag{2-2}$$

which is a rank-1 tensor with a dimension $\mathbb{R}^{\times^{(n+m)}2}$, where $\mathbf{x} \in \mathbb{R}^n$ is the state vector, and $\mathbf{u} \in \mathbb{R}^m$ is the input vector.

The dynamics of the system are given by the parameter tensor F, which is given, in CP-decomposed form, by

$$\mathsf{F} = [\mathbf{F}_\Phi, \mathbf{F}_{u_m}, \cdots, \mathbf{F}_{u_1}, \mathbf{F}_{x_n}, \cdots, \mathbf{F}_{x_1}] \cdot \boldsymbol{\lambda} \in \mathbb{R}^{n \times^{(n+m)}2} \tag{2-3}$$

Using CP tensor decomposition, we are able to normalize the factor matrices (from $\mathbf{F}_{u_m}$ to $\mathbf{F}_{x_1}$) in F to column norm of one.

Finally, a normalized rank-1 MTI model is given by the equation (2-1), but with a parameter tensor

$$\widetilde{\mathsf{F}} = [\tilde{\mathbf{f}}_{\Phi}, \tilde{\mathbf{f}}_{u_m}, \cdots, \tilde{\mathbf{f}}_{u_1}, \tilde{\mathbf{f}}_{x_n}, \cdots, \tilde{\mathbf{f}}_{x_1}] \cdot \lambda \,, \tag{2-4}$$

where matrix $\mathbf{F}_{\Phi}$ is reduced to a vector

$$\tilde{\mathbf{f}}_{\Phi} = (\phi_1 \ \phi_2 \ \cdots \ \phi_n)^T \in \mathbb{R}^n \,, \tag{2-5}$$

and $\tilde{\mathbf{f}}_{u_j} \in \mathbb{R}^2$ and $\tilde{\mathbf{f}}_{x_i} \in \mathbb{R}^2$ are vectors with any norm $||\tilde{\mathbf{f}}_{u_j}|| = ||\tilde{\mathbf{f}}_{x_i}|| = 1$ of one.

In this model, each element of the tensor is defined by the value of vector $\boldsymbol{\lambda}$ and an angle [2]. A visual example can be seen in Fig. 2-2. In this Figure, factor vectors are represented in 2-D, where the non-normalized vectors ($\lambda = 1$) are represented by the dashed-line arrows, and the normalized vectors ($\lambda = 10$) are represented by the continuous-line arrows and all have a length of 1. The angles (represented by the red lines), as expected, do not change. Parameters denoted by x_i in the vertical axis represent the dependency of a transition on the signal x_i.

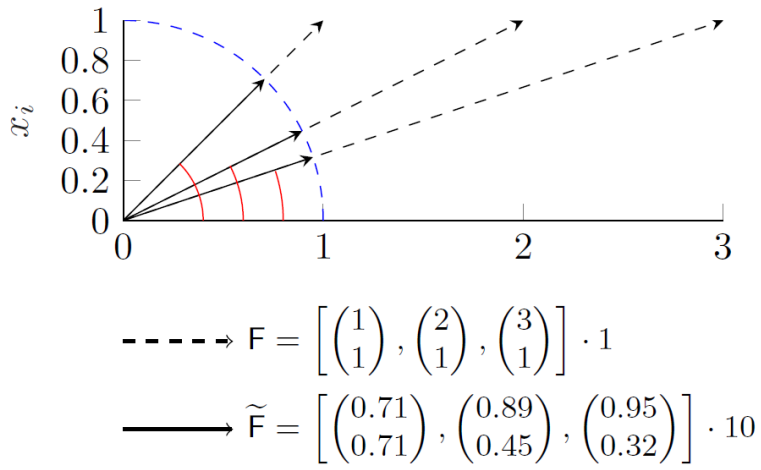

Figure 2-2: Visual example of CP tensor decomposition normalization [2]

2-2-2 Parameter identification with low-rank multilinear model

The proposed grey box parameter identification algorithm in [2] is used in this master thesis for anomaly detection. This algorithm uses the simplified MTI model described in Section 2-2-1, and it is implemented using the programming and numeric computing platform MATLAB, developed by MathWorks, using the MTI toolbox introduced in [9].

2-3 Data point tagging

In this Section, a tool for data point labelling is introduced, which is going to be used in this thesis to rename the data points coming from the BACS of the case study, in order to analyse and group them. An explanation of the utility of this tool in the context of this thesis is also done in this Section.

2-3-1 The BUDO scheme

Building Unified Data point naming schema for Operation management or BUDO is a tool created to name data points coming from buildings in a standardised and unified way, based on

40 standards provided by different operators of BACS [10]. This tool can be found on-line for free, and uses a spreadsheet to create the new names based on the scheme in Fig. 2-3.

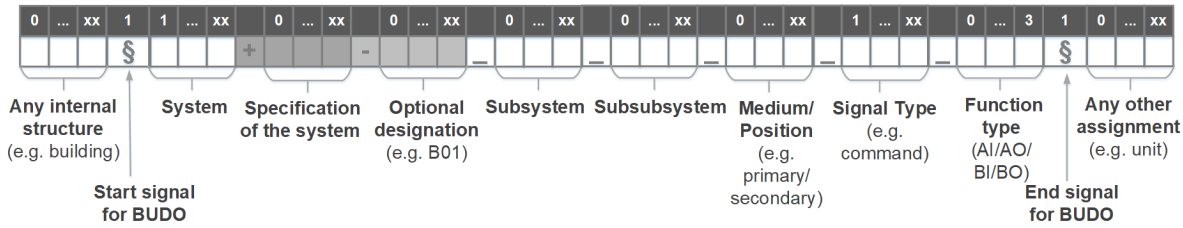

Figure 2-3: Structure of the naming of data points in the BUDO scheme

As seen in Fig. 2-3, it includes two types of labels:

- Non-standardized labels: includes a set of cells to include free categories, both at the start and at the end of the name. These free categories are useful to identify the specific buildings that are going to be tagged, so that any operator can use the BUDO tags for their own buildings.
- Standardized labels: includes a set of cells that have pre-set names. These are located between the symbols "§", which denote the start and the end of the standardized labelling.

The standardized labels include the following categories in a hierarchical structure: System, Subsystem, Subsubsystem, Medium/Position, Signal type and Function type. These include a large variety of specifications that can be selected in the spreadsheet file, as well as optional designations or numberings to, again, refer to specific data points from the operators' buildings.

Each one of the standardized labels are automatically translated in the spreadsheet into a new name that is the BUDO tag. Table 2-1 shows some examples of BUDO vocabulary, which can all be found in [11].

Assignment	Specification	BUDO
sensor	temperature	SEN+T
meter	electrical	MET+EL
measured value	temperature	MEA+T
setpoint		SP+
volume flow controller		CTRL.VF+

Table 2-1: Example of the vocabulary of the BUDO scheme [11]

BUDO considers plenty of standards from operators, technology and research in order to define the vocabulary and the categories [12]. Specially standards from operators are taken into account. This not only ensures that vocabularies and categories are correctly designed, but also contributes to the creation of a standardized method that may one day be officially defined because of the increasing number of buildings that rely on big data.

2-3-2 Utility and application of the BUDO scheme

Some commercial and industrial buildings come with a high amount of data points, which are usually named according to specific subjective criteria of the operator. These names do not always provide well-structured information about what the data points are representing.

Specially buildings with hundreds or even thousands of data points can benefit from a standardized data point naming [13]. These can provide very useful information for the people that have to work with the BACS of a specific building to, for example, detect errors, specially for new possible workers that are not familiar with the system from the start. The SONDE project [1] is a good example of project that would benefit from this to accomplish an automated anomaly detection.

There are currently no industry-wide accepted standards for data point tagging, which makes it difficult to analyse big data volumes, to the point that the purpose of energy saving and efficiency of the BACS cannot be achieved as a result of poor traceability [14]. This also forces building operators to rely on their building automation vendors, since they are the only ones capable of processing the BACS installed in their building [14].

Four case studies are presented in [10], in which real systems' data point names are translated into BUDO tags. These examples show that the BUDO scheme can be applied to existing buildings as well, and it can be done without specialized training via the drop-down menu of the spreadsheet file. However, when it comes to buildings that have thousands of points, doing this task manually will take plenty of time. This is why the team that created the BUDO scheme is working in a machine learning-based program to automatically translate the data point identifiers into the BUDO standard. This tool is known as Artificial Intelligence based Key Interface for Data point metadata nOrmalisation or AIKIDO [14].

At the time of realization of this thesis the AIKIDO tool was not available, thus a simpler solution based on Regular Expressions (see Section 2-4) is used.

2-4 String pattern matching

String pattern matching (Fig. 2-4) occurs in the field of science and information processing, specially in computer science where processes such as text editing, lexical analysis and information retrieval occur frequently [15].

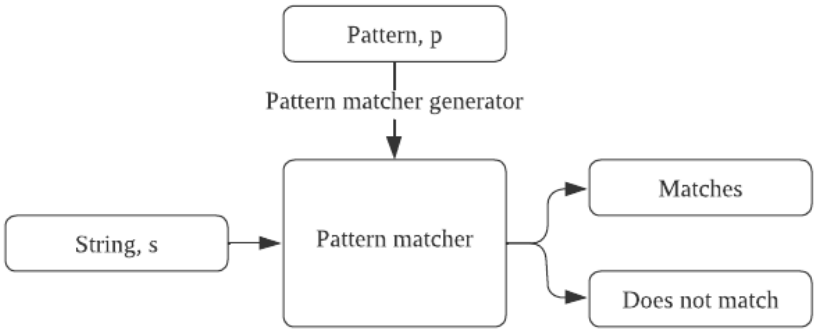

Figure 2-4: Basic scheme of pattern matching in a string [15]

The basic problem is to find a keyword as a substring in a sequence of characters. As an example, the string "Germany" contains the substring "German". One of the most common uses of string pattern matching lays in search engines or programs such as e-mail managers or text editors. In this thesis, the string matching will be done through the use of Regular Expressions in order to find and group data according to the string information contained within.

Regular Expressions (RegExps) define regular languages and, in practice, they are used for string matching [16]. These can express the strings to be matched in a declarative way and, therefore, RegExps are an algebraic description of the language [17].

As an example, search engines like the one in Fig. 2-5 use the notation of Regular Expressions so that the user can describe a pattern to be matched and find a specific file or web page.

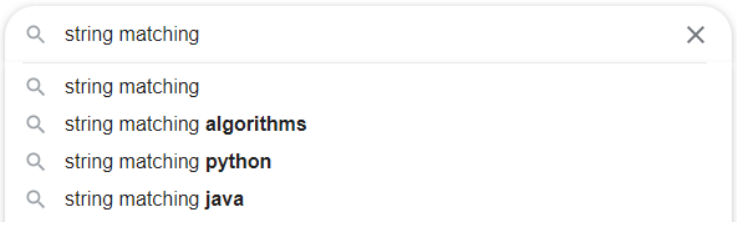

Figure 2-5: The Google search engine is a good example of RegExps use

As previously stated, since RegExps are an algebraic description of the language, they follow the pattern of any algebra, which consist on basic expressions such as constants and variables which are then used to build more expressions with the use of operators [17].

As an example of possible syntax in the RegExp matching, Perl programming language metacharacters can be found in Appendix A-1, as shown in the Perl documentation [18]. Not all programming languages use the same operators, but are usually similar. Other programming languages that include RegExps in the standard library are Java and Python.

Chapter 3

Creation of new automated process

The new automated process for parameter identification designed in this thesis is defined in this Chapter, together with the description of the real case building to which said process is applied. The BUDO scheme is also manually applied to some of the data points of the building.

3-1 Automation of parameter identification process

In this Section, the new automated process designed for the parameter identification of buildings is explained, which is created in accordance to the object and the scope of this thesis (see Sections 1-2 and 1-3).

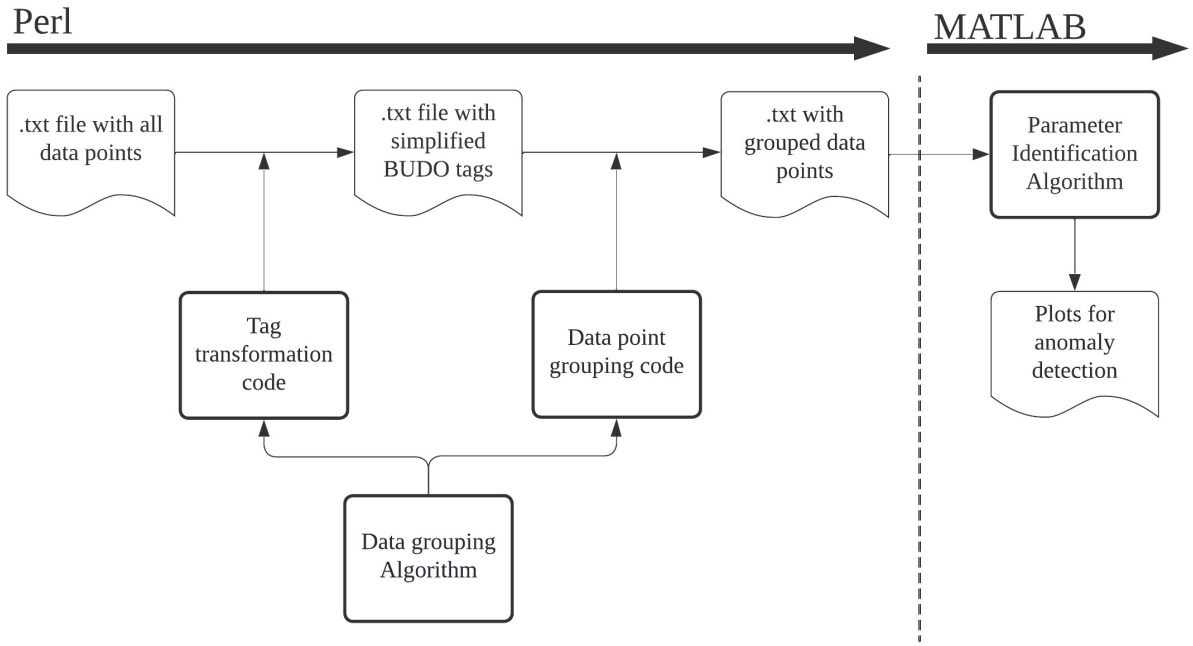

Figure 3-1: Scheme of the new automated process

As shown in Fig. 3-1, the new process consists of two main algorithms that are to be coupled together:

1. the data point grouping algorithm (see Chapter 4), and
2. the low-rank parameter identification algorithm (see Chapter 5)

The objective of the data point grouping is to gather as many data points from the building as possible according to the string in the data point names in order to create an output file to use in the parameter identification process. To group the data in this file, Regular Expressions are used to match the string data based on the tags proposed by the standardized BUDO scheme. This is all implemented via an algorithm built from scratch for this thesis.

The file generated in the data grouping algorithm is then introduced in the low-rank parameter identification algorithm, which belongs to [2]. The parameter identification is then applied more easily since the data is grouped accordingly into a sparse matrix.

These two algorithms are to be combined together in order to automate the process as much as possible, so that one algorithm is capable of combining both, with as less human action as possible (see Section 5-3).

3-2 Case study

In this Section, the case study building that is used for this thesis is presented, together with an analysis of some data points that belong to some specific rooms. The BUDO scheme is applied to these data points as an example of use of this standardized naming method. This will be useful for the first phase of the data grouping algorithm.

3-2-1 Multi-use building in Hamburg

The data used for this thesis comes from a multi-purpose smart building located in Hamburg's digital innovation district in Germany, which relies on big data to work. The data points of this building provide a real example of how the information comes with a building of such magnitude.

The uses of this building include co-working spaces, meeting rooms, event spaces, as well a restaurant, café, library and other facilities such as a sport room and a music studio. The amount of purposes for which this building is used needs a complex BACS to manage all the different requirements for the indoor comfort.

The amount of data points provided by the building, as well as the poor information provided by the labelling, makes it difficult to analyse the data. The server allows to monitor each data point in a graphical way, but it requires to manually select the data points to be represented. This means that previous knowledge about the data points is required, specially in order to know which ones are related to the zone/room that is being analysed.

3-2-2 Data points from the building

The BACS that is implemented in the building provides numeric data from as much as 17,300 data points simultaneously each minute. Most of the names of the data points provide little or no information about the data they contain.

As a first approach, the provided data points can be classified as:

- States: parameters that are affected by the Inputs and Disturbances, and cannot be controlled directly. These have to be between certain limits in order to ensure the indoor comfort. For example, room temperature and air quality are classified as States.
- Inputs: these parameters can be controlled by the BACS or the operator and are used to regulate the States. For example, the air supply and exhaust are classified as Inputs.
- Disturbances: parameters that have an impact on the States but cannot be controlled by the BACS or the operator. For example, outside temperature and occupancy are classified as Disturbances.
- Set-points: these are parameters that are used to control Inputs and do not have a direct impact on the States. For example, upper and lower temperature limit set-points are classified as Set-points.

This classification is important for both algorithms explained in Section 3-1, and are further used and explained in Section 4-2. Following up, a set of data points from the building are provided as an example of how these are displayed in the server.

Data points from rooms 3.01 and 3.45

To simplify the information, this thesis focuses on the data points of room 3.01 and room 3.45. While room 3.01 is an open space, room 3.45 is used as a co-working office, meaning that they have different uses and, as a result of this, different energy and indoor comfort requirements.

The main data points associated to these rooms, which have been identified with the help of the BACS operator, can be seen in Appendix A-2. These are the IDs as shown in the server where the information is stored. Furthermore, some of the data points that contain information from the weather station in the building are shown.

A short description of the functionality of the data contained within the represented data points from rooms 3.01 and 3.45 can be found in Table 3-1.

Object	Unit	Description
Room temperature	ºC	Temperature of the room, measured through a temperature sensor in the room
Air quality	ppmCO2	Concentration of CO2 in parts per million, measured through a sensor in the room
Ceiling temp.	ºC	Temperature of the room, measured through a temperature sensor located on the ceiling
Power	W	This data point measures the power that is consumed only by the lighting and the smart sensors in the room
Air Supply [real]	m3/h	Fan coils distribute the air coming from the air handling unit into the rooms. The air volume flow controller is located in the conduct before the fan coil
Air Exhaust [real]	m3/h	The exhaust air comes out through ventilation grilles located on the wall. The air volume flow controller is located in the exhaust conduct

Object	Unit	Description
Air Supply [target]	%	This data point describes the percentage of the maximum air volume flow that the volume flow controller is letting in
Air Exhaust [target]	%	This data point describes the percentage of the maximum air volume flow that the volume flow controller is letting out
Occupancy	Binary	This data point is a binary input (BI). If the sensor detects movement in the room, the occupancy data is set to 1
Temp. set-point	K	This set-point is specified centrally and can be adjusted by the user in the range from +3 to -3 K
Ventilation set-point	-	The ventilation can be controlled in stages. It can take values from 1 to 3. (boost, aus, auto). This set-point is always in auto mode
Lighting	-	The light can be controlled in stages. It can take values from 1 to 7 (different intensities depending on the value)
Air quality set-point	ppmCO2	This set-point is used to set the upper limit of the CO2 concentration in the room
Temp. s-p adjust.	ºC	The data points associated to this parameter are used to adjust the temperature set-point in the room

Table 3-1: Data contained within the data points of rooms 3.01 and 3.45

As shown in the Tables in Appendix A-2, the names of the data points do not always provide information about what they contain, and some of the specific descriptions of Table 3-1 are only known because the BACS operator has provided the information (see supporting files 2 and 5 from the operator in Appendix B) and/or through the analysis of the data point values in the server, one by one.

Application of BUDO scheme to the real data

The poor traceability problem can be solved by using a standardized naming method such as the BUDO scheme (see Section 2-3-1), which is applied manually to the data points from rooms 3.01 and 3.45, as well as for the weather station data points in Appendix A-3. For the rest of the rooms of the building, the scheme is applied automatically by the first phase of the data point grouping algorithm. The use of the BUDO scheme is necessary for the second phase of the data point grouping, which is based on the string information contained within the BUDO tags. This is all further explained in Chapter 4.

As we can see in Appendix A-3, the BUDO tags for each room consist of a fix part before the § symbol, which identifies the building and the specific room; and a variable part after the § symbol which identifies the information contained within the data point. All the BUDO vocabulary can be found in [11].

Chapter 4

Data point grouping algorithm

This Chapter shows how the data point grouping has been performed, providing information about the output of this algorithm that is then going to be used in the parameter identification, and the used Regular Expressions to implement the process using the Perl programming language. The Perl codes are also explained in this Chapter. The data point grouping algorithm can be found in file 3, attached in Appendix B.

4-1 Structure and format of data point tags

There are mainly four types of labelling in the server case study building in Hamburg, as a result of different controllers being used. These labels include a fix part that identifies from which controller the data point comes, and a variable part that is unique to each data point. Table 4-1 shows the label structures using a data point from each type as an example.

Type	Fix part	Variable part
A	DAIKIN MasterStation III No 100050-100050_	RoomTemp_104-AI26633
B	ISP_Level_	01-101010_Raum_7_13_Licht-MSV42 03-103010_HBK_1a_03_L_RLT01_03_010_VVR40_ST_1-AO40 00-100010_RLT_1_Gesamt_Zaehler_Wirkleistung-AV129
C	Level0-smartdirector-D01-100040_	1923502:3.09_HBK Ceiling Temperature-AI1923502
D	pCOWeb	100080-100080_Humidity_Room-AV7

Table 4-1: Structure of the main types of data point tags in the server

At first glance, data points of **type A** and **type D** provide only a slight clue of what they contain, but do not provide data about the zone/room to which they belong. Meanwhile, **type C** labels provide both the data they contain and the room number. As for **type B** labels, the provided information depends on the different variable parts. A deeper look into the structure of each of these data points can be found in Appendix A-4.

Data point labels are not consistent, and that is why they need to be transformed first into BUDO tags in order to work using Regular Expressions systematically. To do so, a program based on Regular Expressions detecting the structure of the data points has been created to catch as many tags as possible (see Section 4-4).

Moreover, all the data point names can be downloaded into a .txt file, containing a label per each row (in total 17,300 rows). For the data grouping algorithm, all of the labels are going to

be identified by their row number in this .txt file, referred to as the **original data point file** (see file 1 in Appendix B). This is done so as to simplify the process, since the data point tags contain long strings with special characters.

4-2 Output of the data grouping algorithm

The grouping is made to organize the data points so that they can be introduced in the parameter identification algorithm described in Chapter 5. The chosen format is a .txt file with comma-separated values.

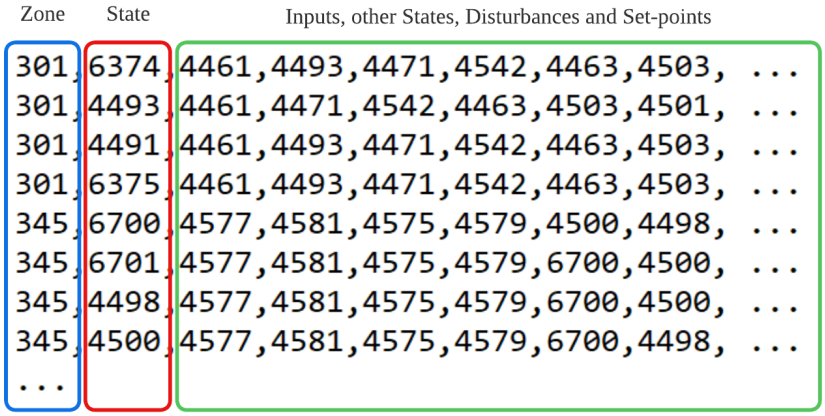

Figure 4-1: Grouping of the data in the .txt file

The information is grouped in the output .txt file as shown in Fig. 4-1. In this file, for each identified State, a new row is created. The first column shows the zone/room where the State signal is located, and the second column shows the identified State. The rest of the columns display the Inputs, other States, Disturbances and Set-points that could potentially have an impact on the identified State, based on the zone/room in which these are located. Each data point is identified by their row number in the original data point file.

It is also relevant to note the order in which each data point is printed. This order is decided according to the effect that the rest of the parameters may have on each identified State, and is as follows:

1. Inputs and/or other States
2. Disturbances
3. Set-points

Inputs, other States and Disturbances have a direct effect on the States, since they represent physical phenomena involved in the building. However, Set-points have an indirect effect on each State because they are used by the controllers to modify the Inputs. Fig. 4-2 shows the graphical representation of two data points: one of them is the air quality state (CO2) in room 3.45, while the other one is the set-point that is used to set the desired CO2 level in the same room. As we can see, the values of the set-point do not vary too much and are usually constant. Because of this, Set-points have less priority when applying the parameter identification and thus go at the end of the row.

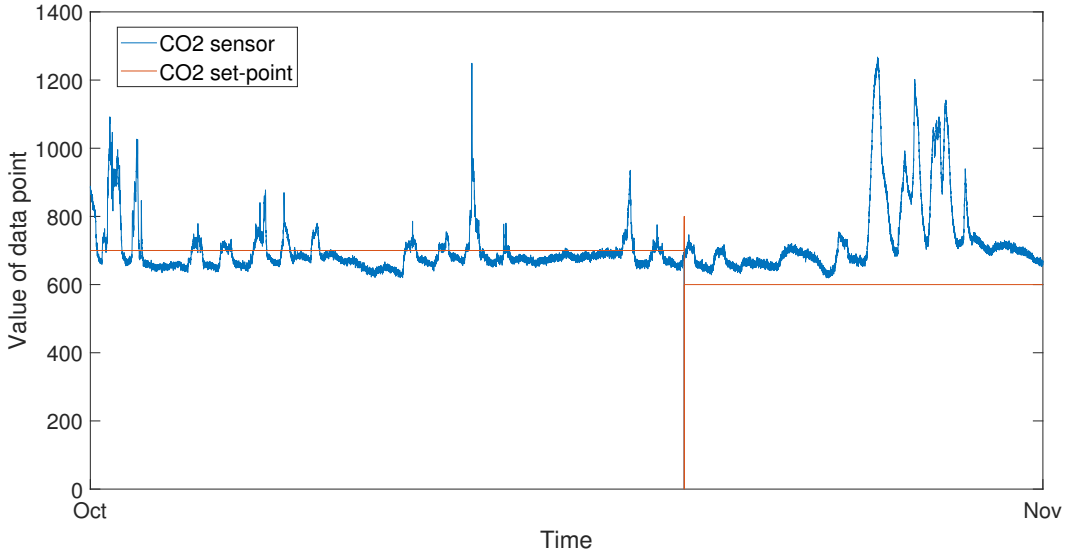

Figure 4-2: CO2 sensor measurement values as compared to CO2 set-point values

Note that other States are also signals that can affect each identified State, and this is why they are also included in the rest of the columns. Of course, these are signals that are not controlled directly and they change over time (they are do not have a fix behaviour, as we can see in the CO2 sensor representation in Fig. 4-2), but might have an influence on other signals.

The generated output file in this format is essentially a sparse matrix that is used as an input into the low-rank parameter identification algorithm. This is further explained in Section 5-1.

4-3 Used Regular Expressions

Two types of RegExps are going to be used in this master thesis in order to achieve the output file during the data grouping process. In order to show the RegExps, Perl nomenclature has been used. Perl is the programming language that has been used to create the data grouping algorithm, composed by two phases (see Section 4-4).

4-3-1 Structure detection RegExps

These are used in the data grouping algorithm to detect the structure of both the original data points from the server and the BUDO tags. There are two types: number detection and string detection RegExps. These are written in the nomenclature used by the Perl programming language as follows:

```
my $number = /([0-9]+)/;    # Number detection: used to detect any number in a ...
                            # ... specified position.
my $string = /([^_]+)/;     # String detection: used to detect any string composed ...
                            # ... by letters/numbers in a specified position.
```

These expressions are used for the detection of numbers and strings that are not specific, but rather a possible number/string that can be located in a specified position. The "_" symbol in the string detection expression is used to tell the program to match a string until it reaches that symbol. As an example, Code 4.1 shows a simple Perl program that uses these expressions

to detect the number `301` and the string `Temperature` inside the string data provided by the variable `$data`.

Listing 4.1: Simple code to print specified numbers and strings

```
my $data = "The data point name is 301_Temperature_Building1";

$data =~ /([0-9]+)_([^_]+)/;
my ($number,$string) = ($1,$2);

print $number;  # Prints the number "301".
print $string;  # Prints the string "Temperature".
```

So, in short, the point of these RegExps is to match any string or number that has not been yet specified, in order to catch the data point names to process them further using the specific, BUDO scheme-based strings, explained in Section 4-3-2.

4-3-2 BUDO string RegExps

These are used to detect the BUDO tags in the data point names in order to group them accordingly. Different RegExps are used according to the different types of data points, classified as: States, Inputs, Weather Station data points (considered as the Disturbances) and Set-points. The Regular Expressions are as follows:

```
$state =~ /SEN\+T|SEN\+AQ|MET\+EL/; # Matches "SEN+T" or "SEN+AQ" or "MET+EL".
$inputs | $states =~ /MEA/;         # Matches "MEA".
$weather =~ /WST_/;                 # Matches "WST_".
$setpoints =~ /SP|STEP|VAL/;        # Matches "SP" or "STEP" or "VAL".
```

These RegExps are based on the information in Section 3-2-2, where the data point names from room 3.01, room 3.45 and the weather station are translated into the BUDO scheme. The strings shown above are characteristic of each type of data point, which means that the data point type can be identified by the string data that the BUDO tag displays. Symbol "\" in the RegExps is used as an escape sequence so that the math operators (+) are matched as literal strings.

Note that States also contain the string `/MEA/` in the BUDO tag. This string is used in the algorithm to add other States in each row as data points that may as well have an impact on each State, as explained in Section 4-2. Moreover, string `/VAL/` is added to detect the data points from heating/cooling valves, considered as set-points due to its nature, caught by the first phase of the data point grouping algorithm (Section 4-4-1).

As stated above, these RegExps are chosen based on the available information from the case study and the manual application of the BUDO scheme; therefore, more or less different RegExps could be used depending on the studied building. All the BUDO nomenclatures mentioned above together with their meaning can be found in [11].

4-4 Implementation with Perl

The data point grouping algorithm has been implemented with the Perl programming language [19], which specializes in Regular Expressions. This language can be downloaded and used for free without any restrictions.

The algorithm is composed by two phases. The first phase is used to transform as many original data point tags from the server (named according to the BACS operator criteria) into simplified BUDO-based labels (Section 4-4-1). Meanwhile, the second phase groups the tags into the desired output .txt file (Section 4-4-2). The corresponding full Perl codes are available in the Appendix A-5 and the Perl files together with the .txt files that these use can be found in file 3 in Appendix B.

4-4-1 First phase: tag-transformation code

In order to catch as many data points as possible, the Code A.1 has been developed, which uses RegExps to get the data points based on the string data and on previous knowledge provided by the operator of the BACS. This program detects the structure of the data point names and simplifies them using BUDO nomenclature.

This code does NOT detect every data point available. Because of the characteristics of each data point name, this program is able to catch some of the **type B**, **C** and **D** tags (see Table 4-1). Appendix A-4 shows a deeper analysis of the original data point tags and whether they can be efficiently caught using Regular Expressions or not.

Code A.1 does not create a full BUDO tag, but rather a simplified one with a structure like the example: `301§MEA+VF`, which includes the room number and the minimal characteristic string to identify the data point type without the need of the full BUDO name.

As an example, Fig. 4-3 shows in a schematic way how the data point structures are detected with this code, using the RegExps presented in Section 4-3-1.

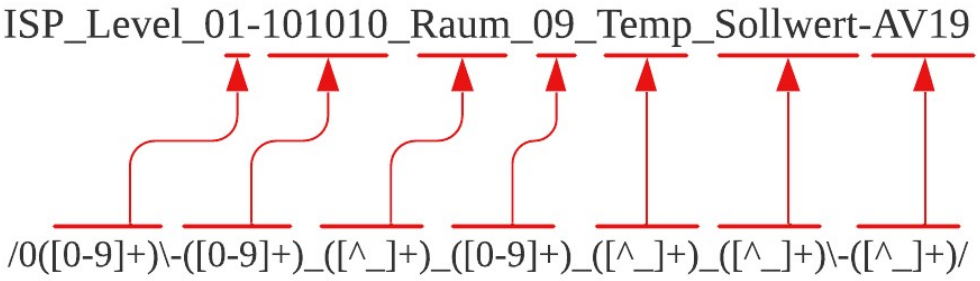

Figure 4-3: Example of detection of structure in Code A.1

Note that this program is specifically made to apply on the case study building. However, the RegExps and the content of each `if` and `elsif` can be modified to adapt this program to other buildings.

It also must be noted that Code A.1 will NOT catch the original data point labels from rooms 3.01 and 3.45 and from the weather station presented in Section 3-2-2 because these have been replaced manually in the original data point file by the corresponding BUDO tags. This ensures that these data points are detected correctly by the code in Section 4-4-2 and are added in the output file to then perform the parameter identification (Chapter 5) and make an overview of the results. Even so, the code could be modified to catch these tags or more based on the provided knowledge.

4-4-2 Second phase: data point-grouping code

Code A.2 is used to organise the data points to get the desired output indicated in Section 4-2. It is based on the use of hashes, which are a set of key-value pairs [18]. As an example, the following is a hash in which the country names are the keys, and the languages are the values:

```
my %countries = (  England => 'English',
                   France => 'French',
                   Spain => 'Spanish',
                   China => 'Chinese',
                   Germany => 'German'   );
```

It is also important to know about the **multidimensional hashes**, which are essentially hashes of hashes. An example of a simple multidimensional hash is shown below:

```
my %continent = (Europe => {
                              'Spain' => 'Spanish',
                              'France' => 'French',
                              'Germany' => 'German',
                           },
                 Asia => {
                              'China' => 'Chinese',
                              'India' => 'Indian',
                              'Iran' => 'Persian',
                           },
                 America => {
                              'USA' => 'English',
                              'Colombia' => 'Spanish',
                              'Brazil' => 'Portuguese',
                           },                              );
```

In Code A.2, the **first section** groups all the data points into a multidimensional hash that contains the room number and the names of the tags as keys, and the row numbers as values. The row numbers are retrieved from the original data point file by Perl using the `$.` expression, while the literal names of the tags are retrieved using the `$_` expression. All data points that are considered States are then grouped into another hash of hashes.

The **second section** of the Code A.2 is used to create the output multidimensional hashes in order to print them in the **third section**. The reason why there is one hash for the Inputs and another one for the Set-points is because Inputs are going to be printed in each row before the Set-points, as explained in Section 4-2. Other States are also included inside the same hash as the Inputs.

All Regular Expressions used to detect States, Inputs, Set-points and Weather Station data points are described in Section 4-3.

As we can see in the third section of the Code, the information in the output file is organized as follows:

```
print OUT (join ',',
$room_num,$state,@{$inputs{$room_num}{$state}},@wst,@{$setpoints{$room_num}{$state}});

# Order: room number, state, inputs/states, disturbances (weather), set-points.
```

Weather can have an effect on all the rooms of the building, and this is why weather data points are directly printed into an array instead of a hash, because they do not belong into a specific room. All of the weather-related disturbances are taken into account for each row.

Note that this code is made to apply specifically to the case study building, although Regular Expressions can be modified in order to adapt the code to any desired building which uses a standardized data point naming scheme.

Chapter 5

Parameter identification algorithm

This Chapter serves the purpose to explain how the data grouping and the anomaly detection through the parameter identification have been integrated together, as well as how the parameter identification is performed. Furthermore, interesting plots that could be used for anomaly detection are presented in this Chapter. The parameter identification algorithm can be found in file 4 in Appendix B.

5-1 Input sparse matrix

The input sparse matrix

$$\alpha = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1(n+m)} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2(n+m)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{n(n+m)} \end{pmatrix}, \tag{5-1}$$

into the parameter identification comes from the .txt file defined in Section 4-2.

The data grouping algorithm acts as a first filter so that the parameter identification algorithm does not have to go through all the data points to check if they have an effect on each identified State. The idea is to create the sparse matrix as this example

$$\alpha = \begin{pmatrix} * & 0 & 0 & * & \cdots & 0 \\ 0 & 0 & 0 & * & \cdots & * \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ * & 0 & 0 & 0 & \cdots & 0 \end{pmatrix}, \tag{5-2}$$

where the stars (*) represent any value different from 0. As we can see, this matrix would have mostly 0 since most of the data points of the building do not have an effect on a specific State. For example, the temperature of room 3.45 will not be influenced by the occupancy of all the rest of the rooms in the building, so these are already set as 0 thanks to the data grouping algorithm.

5-2 Parameter identification algorithm process

The parameter vectors as referred in Eq. (2-4) can be represented using their angle (α) as seen in Fig. 5-1.

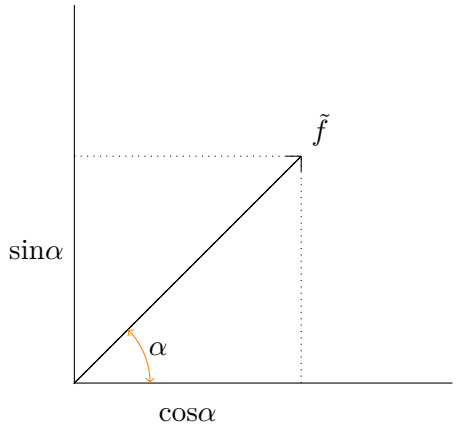

Figure 5-1: Representation of vector $\tilde{f}$ and its angle

Knowing this, these vectors can be represented using their angle in Eq. (2-1), which results in the equation in the form

$$\mathbf{x_i}(k+1) = \lambda_i \cdot \prod_j (cos(\alpha_{ij}) + sin(\alpha_{ij}) \cdot z_j(k)), \tag{5-3}$$

where $z(k)$ comprises both the states $x(k)$ and inputs $u(k)$ from the monomial tensor (2-2).

When substituting the values from the α sparse matrix (5-1) in the Eq. (5-3), for the values that are 0, the sums are equal to 1, meaning that these can be neglected in the product and thus offer no effect on the $\mathbf{x_i}$ value.

The parameter identification based on Eq. (5-3) is done using the method in [2]. The parameter identification is realized for each row of the α sparse matrix in order to fill it row by row with parameters. The algorithm is implemented with MATLAB (see attached file 4 in Appendix B). For each row of the .txt file, the MATLAB algorithm first identifies the zone and the State (first and second column), and then, for the identified State, it performs the parameter identification checking each data point in the rest of the columns.

The algorithm is able to cross-reference each of the row numbers from the input generated file to the original data point file. This way, it can retrieve the name of the data points that belong to each row number, and it is able to download the data from the server to perform the parameter identification.

5-3 Automation of the whole process

Unlike the data grouping algorithm, which is implemented using Perl, the parameter identification algorithm from [2] was created using MATLAB. However, other different codes can be run directly from MATLAB. This way, it is possible to run the whole process of data grouping and parameter identification with just one click. In order to do so, the `isfile` function in Matlab can be used, adding the following lines at the start of the program (or similar, depending on the file paths and names):

```
if ~isfile('filepath\PI_input.txt')    % Check if input .txt file exists.

    perl('filepath\tag_transform.pl'); % Run Perl algorithm.
    perl('filepath\tag_group.pl');
end
```

With these lines, MATLAB will first check if the input .txt file exists. If it exists, the program will continue as normal, without running the data grouping algorithms. If it does not exist, MATLAB will use the `perl()` command in order to run the .pl files using the Perl language (which is previously installed in the user's operative system).

5-4 Overview of plot results

In this Section, a series of plots with the results of the parameter identification are shown. These are limited to the rooms 3.01 and 3.45. In this case, all the plots correspond to the **year 2021**, since it is the most recent year in which there is data during the whole 12 months. The chosen time range for the parameter identification is the week and, therefore, the parameter identification is performed for each of the 52 weeks of the whole year.

5-4-1 Initial analysis and behaviour of data

As a first analysis of the results, plots for each one of the weeks have been made in which the value $\mathbf{x_i}$ is represented against each one of the parameters that affect a specific state. Also, the lambda (λ) parameter for that week is shown. As an example, the graph in Fig. 5-2 shows the values of the parameters for the temperature in room 3.45 for week 51 of the year.

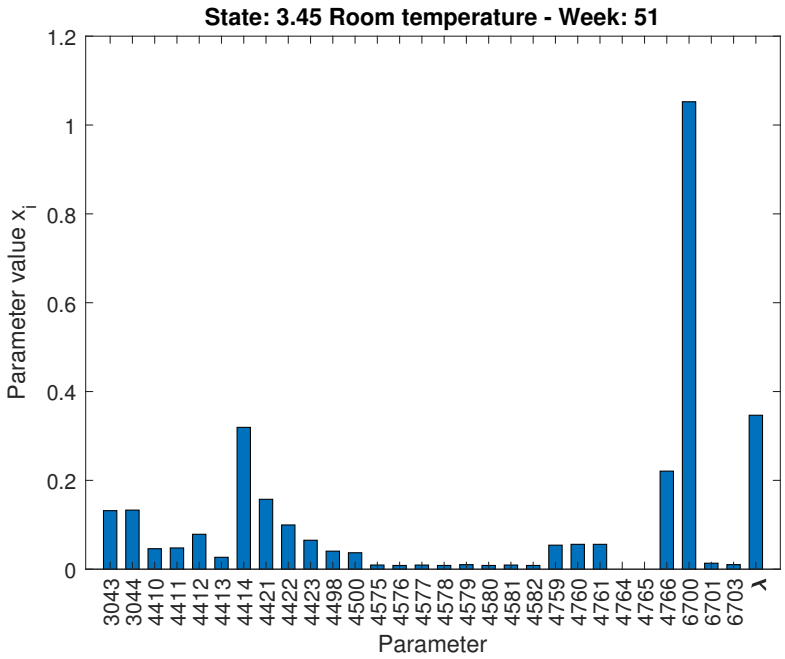

Figure 5-2: Parameter values for the temperature in room 3.45, for week 51

In this figure we can clearly see that the parameter that has the most influence on the room 3.45 temperature is the one that belongs in row number 6700 (see Appendix A-6), which is

the ceiling temperature. However, the results can vary quite a lot depending on the week, and using a plot for each week is not efficient nor intuitive (for each identified State, we would need a plot for each of the 52 weeks of the year). Moreover, with these plots we cannot establish a threshold within which we can assume a normal functioning of the building. The amount of parameters also makes it difficult to correctly label de x-axis with longer strings.

Furthermore, some of the plots came out as the one shown in Fig. 5-3, which does not show any useful information at a first glance.

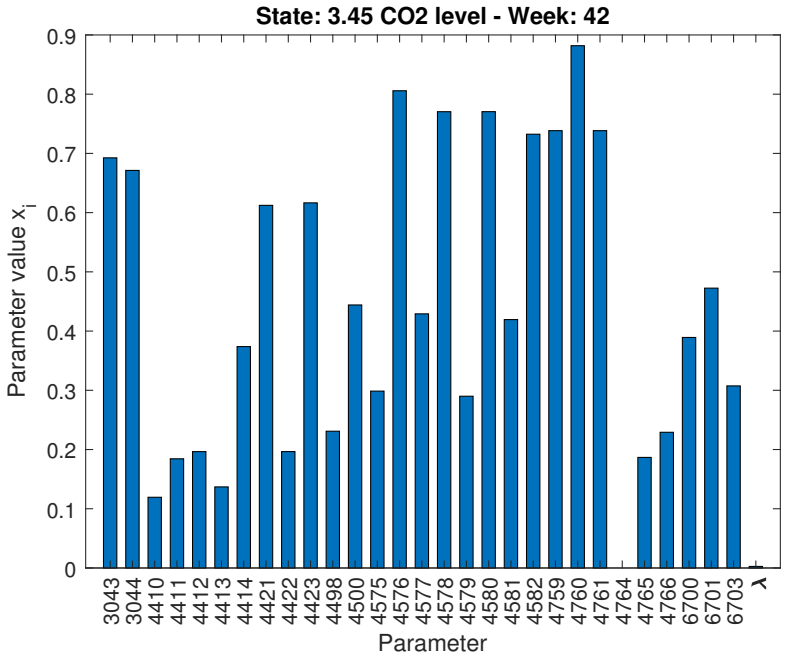

Figure 5-3: Parameter values for the CO2 level in room 3.45, for week 42

For these reasons, this plots can be discarded for the analysis of the parameter identification results. The chosen plots that could be used for anomaly detection are shown in Section 5-4-2.

Finally, the data coming from the server also shows inconsistency in the retrieving of the information. This can be due to various reasons such as errors in the server, or even maintenance of any component of the BACS.

5-4-2 Plots for anomaly detection

In this Section, the plots that have been chosen for possible anomaly detection in the building are shown. For a chosen State, these plots represent the λ value against the $\mathbf{x_i}$ values for a chosen parameter during all the 2021 year. This results in a cloud of points, each point representing 1 of the 52 weeks of the whole year.

Each week is represented in a different shape and colour depending on the season of the year they belong to. This could be useful since the weather conditions such as the outside temperature have an important effect on the behaviour of the BACS, which regulates the different inputs in order to provide the desired indoor comfort.

Using this plots, it could be possible to detect anomalies based on the position of the cloud of points, and further analysing the weeks that stray further away from the main cloud. A threshold could then be defined, outside which we can assume an anomalous behaviour. Following up, some of the plots are shown as examples on how the data can be represented. The further

analysis is limited to an overview, which provides a guideline on how the information provided by the proposed plots could be analysed more deeply to detect the anomalies.

In this case, the analysis of the plots is done manually, looking at the value of each point. However, classification methods such as the support-vector machines, which are used in [2], could be used to automatically classify the values accordingly and set a threshold.

Plot for 3.01 Room temperature vs. Air exhaust

Fig. 5-4 shows the influence that one of the air exhaust parameters has on the 3.01 room temperature from the results of the parameter identification. We can see that the main cloud of points is located on the left of the plot, most of the points having an $\mathbf{x_i}$ value of 0 or close.

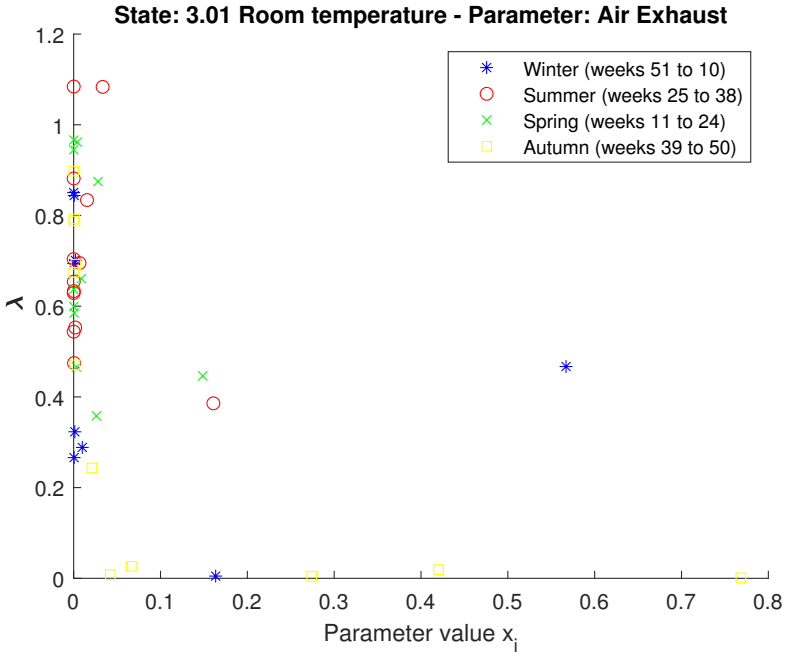

Figure 5-4: Influence of the air exhaust parameter on the room 3.01 temperature state

Further questions could rise, such as why some of the points stray away from the main cloud, and why there is a point that is very close the an $\mathbf{x_i}$ value of 0.8. Further analysis of the data on the server and how the BACS was working in that moment should be done in order to find out where these anomalies in the data come from.

Plot for 3.45 Power consumption vs. Occupancy

Fig. 5-5 shows the influence that the occupancy parameter has on the power consumption state of room 3.45 for the whole year. As stated in Table 3-1, the power data point measures the consumption of the lighting and the smart sensors, and the occupancy sensor is binary, according to the BACS operator.

Knowing this, we could expect that occupancy is related to the power consumption since lighting should be turned on and off depending on the fact that there are people in the room or not. However, Fig. 5-5 shows two main clouds of points: one on the right, composed by weeks that belong to summer and spring; and one on the left, composed mainly by weeks that belong to winter and autumn.

This inconsistency could be related to the behaviour of the data collection in the server during the year, which has been found to give values, during the year 2021, between 0 and 1 for summer and spring. According to the screen-shots taken directly from the dashboard in server (see Appendix A-7), we can see that for August (Fig. A-7), the occupancy and power profiles are more similar to each other; whereas for December (Fig. A-8), the occupancy shows actual binary values. Lower $\mathbf{x_i}$ values can be expected from parameters that use binary values as compared to the analog values of the State.

Moreover, while the summer and spring weeks form the right cloud, there is one week from spring that is located in the left cloud. This is week 16 of the year, and knowing this information, if we plot in the server the power and occupancy for this week, we obtain Fig A-9. We can see that most of the week both values stay constant, which could be the reason of the anomaly.

Further questions, such as why there is one week of winter (week 9) that stays away from the left cloud and has a negative λ value, can be asked. This week is very close to spring, but since the weather does not change drastically from the end of one season to the start of the other, such anomalies could appear in the plots. All of this requires a deeper analysis of all the related data, that can normally only be accessed by the BACS operator. However, these plots could already provide guidelines on questions such as where to start looking, and what to expect.

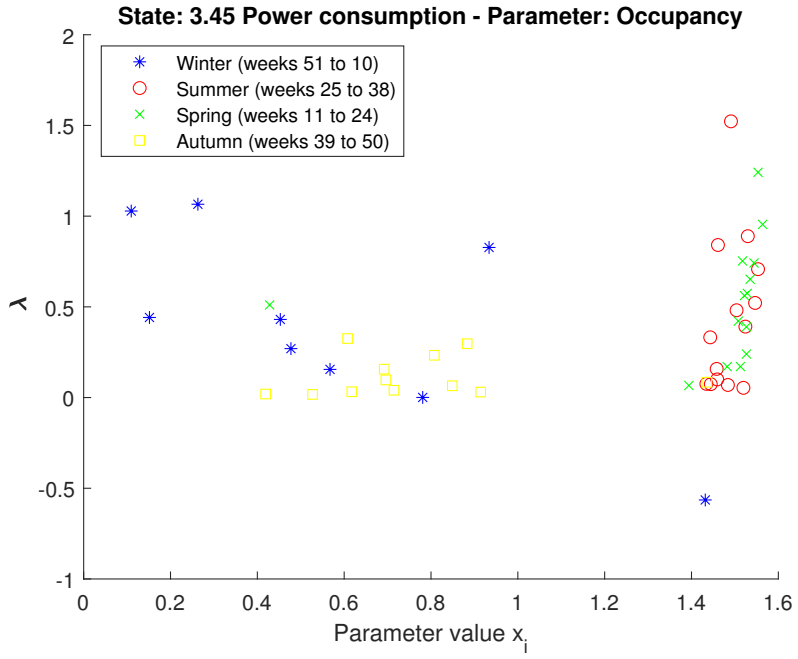

Figure 5-5: Influence of the occupancy parameter on the room 3.45 power consumption state

Chapter 6

Conclusion and outlook

A new automated process for multilinear parameter identification has been created. To do so, a new data grouping algorithm based on Regular Expressions has been built from scratch using the Perl programming language, which is used in conjunction with the already-existing low-rank parameter identification algorithm from the SONDE project, implemented with MATLAB. One only click is required in order to run both algorithms from MATLAB, one after the other. This automated process is useful for buildings that generate large amounts of data points due to the complexity of their BACS. Furthermore, the data from a smart building located in Hamburg, which is composed by around 17,000 data points, has provided a real case scenario to which the new process has been applied.

Regular Expressions have been successfully used for automatic translation of data point tags from the server to more standardized BUDO names, although not each of the 17,300 data points have been able to be translated due to high inconsistency and poor traceability of the labelling. Thanks to the BUDO tags, it was possible to group plenty of data points to have a first guess about which parameters affect each identified State, based on the zone/room where these belong.

Thanks to the data point grouping, the parameter identification algorithm has been run automatically for the case building that has thousands of data points. Some parameters have been identified that match the previous expectation of having an influence on specific States.

The results from the parameter identification are shown as plots that could be used for anomaly detection. Different values for each week of the whole year are spotted in these plots, but there is a tendency that makes these values form clouds of data points, which could be used to establish thresholds to identify anomalous values that stay out of these limits.

Finally, the use of real data from a building has demonstrated, not only the benefits, but also the necessity of a standardized data point naming scheme in order to ensure a correct functionality of the BACS that rely on big data to work. Regular Expressions have shown a big potential to make a good use of the standardized BUDO scheme thanks to the quality string information that BUDO tags can provide.

Further improvements could be done, starting with the automatic translation of data point labels from the server to a standardized scheme. A machine-learning program such as the mentioned AIKIDO tool could be more efficient to translate the tags, and thus offer more data points in the output file of the data point grouping algorithm.

Further deeper analysis of the data in the server, specially with a good knowledge about the building's BACS, could be done in order to find out why the detected anomalies occur. Moreover, classification methods like support-vector machines could be used to automatically identify the values according to their position in the plots, instead of doing this automatically.

The application of a standardized scheme to the data points of new buildings could make a difference and would increase the traceability of the data, thus completely removing the first step of translating the data point tags in the server to further analyse them. The ideal situation is that an international standard, published by an official institution like ISO, would be established.

Appendix A

Attached information to the document

A-1 Perl metacharacters

Perl programming language metacharacters for RegExps:

	PURPOSE	WHERE
\	Escape the next character	Always, except when escaped by another \
^	Match the beginning of the string (or line, if /m is used)	Not in []
^	Complement the [] class	At the beginning of []
.	Match any single character except newline (under /s, includes newline)	Not in []
$	Match the end of the string (or before newline at the end of the string; or before any newline if /m is used)	Not in [], but can mean interpolate a scalar
|	Alternation	Not in []
()	Grouping	Not in []
[	Start Bracketed Character class	Not in []
]	End Bracketed Character class	Only in [], and not first
*	Matches preceding element 0 or more times	Not in []
+	Matches preceding element 1 or more times	Not in []
?	Matches preceding element 0 or 1 times	Not in []
{	Starts a sequence that gives number(s) of times the preceding element can be matched	Not in []
{	when following certain escape sequences starts a modifier to the meaning of the sequence	
}	End sequence started by {	
-	Indicates a range	Only in [] interior
#	Beginning of comment, extends to line end	Only with /x modifier

A-2 Data points from rooms 3.01 and 3.45

Tables A-1, A-2 and A-3 show the main data points associated to the weather station and to rooms 3.01 and 3.45, together with the information they contain and their type.

Original data point ID	Data contained	Type
ISP_Level_02-102010_WS_Windgeschwindigkeit-AV34	Wind speed	Disturbance
ISP_Level_02-102010_WS_Windrichtung-AV35	Wind direction	Disturbance
ISP_Level_02-102010_WS_Temperatur-AV36	Outside temperature	Disturbance
ISP_Level_02-102010_WS_Globalstrahlung_N-AV43	Solar radiation N	Disturbance
ISP_Level_02-102010_WS_Globalstrahlung_O-AV44	Solar radiation E	Disturbance
ISP_Level_02-102010_WS_Globalstrahlung_S-AV45	Solar radiation S	Disturbance
ISP_Level_02-102010_WS_Globalstrahlung_W-AV46	Solar radiation W	Disturbance
ISP_Level_02-102010_WS_Globalstrahlung_Sky-AV47	Solar radiation Sky	Disturbance

Table A-1: Main data points from the Weather Station

Original data point ID	Data contained	Type
ISP_Level_03-103010_HBK_1a_03_L_RLT01_03_090_LQF01_MW_3-AI55	Room temperature	State
ISP_Level_03-103010_HBK_1a_03_L_RLT01_03_090_LQF01_MW_1-AI53	Air quality	State
Level0-smartdirector-D01-100040_1923402:3.01_HBK Ceiling Temperature-AI1923402	Ceiling temp.	State
Level0-smartdirector-D01-100040_1923403:3.01_HBK Power-AI1923403	Power (lighting and sensors)	State
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR13_RW_1-AI13	Air supply 1 [real]	Input
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR15_RW_1-AI15	Air supply 2 [real]	Input
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR14_RW_1-AI14	Air exhaust 1 [real]	Input
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR16_RW_1-AI16	Air exhaust 2 [real]	Input
Level0-smartdirector-D01-100040_1923405:3.01_HBK Occupancy-BI1923405	Occupancy	Disturbance
ISP_Level_03-103010_Raum_01_Licht-MSV15	Lighting	Set-point
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR13_ST_1-AO13	Air supply 1 [target]	Set-point
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR15_ST_1-AO15	Air supply 2 [target]	Set-point
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR14_ST_1-AO14	Air exhaust 1 [target]	Set-point
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR16_ST_1-AO16	Air exhaust 2 [target]	Set-point
ISP_Level_03-103010_Raum_01_Temp_Sollwertverstellung-AV6	Temp. set-point	Set-point
ISP_Level_03-103010_Raum_01_Lueftung-MSV1	Ventilation set-point	Set-point
ISP_Level_03-103010_Raum_01_LQ_Sollwert-AV34	Air quality set-point	Set-point
DAIKIN MasterStation III No 100050-100050_TempAdjust_039-AV9994	Temp. s-p adjust. 1	Set-point
DAIKIN MasterStation III No 100050-100050_TempAdjust_045-AV11530	Temp. s-p adjust. 2	Set-point

Table A-2: Main data points associated to room 3.01

Original data point ID	Data contained	Type
ISP_Level_03-103010_HBK_1a_03_L_RLT01_03_450_LQF02_MW_3-AI58	Room temperature	State
ISP_Level_03-103010_HBK_1a_03_L_RLT01_03_450_LQF02_MW_1-AI56	Air quality	State
Level0-smartdirector-D01-100040_1927902:3.45_Co Working Ceiling Temperature-AI1927902	Ceiling temp.	State
Level0-smartdirector-D01-100040_1927903:3.45_Co Working Power-AI1927903	Power (lighting and sensors)	State
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR17_RW_1-AI17	Air supply 1 [real]	Input
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR19_RW_1-AI19	Air supply 2 [real]	Input
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR18_RW_1-AI18	Air exhaust 1 [real]	Input
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR20_RW_1-AI20	Air exhaust 2 [real]	Input
Level0-smartdirector-D01-100040_1927905:3.45_Co Working Occupancy-BI1927905	Occupancy	Disturbance
ISP_Level_03-103010_Raum_45_Licht-MSV28	Lighting	Set-point
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR17_ST_1-AO17	Air supply 1 [target]	Set-point
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR19_ST_1-AO19	Air supply 2 [target]	Set-point
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR18_ST_1-AO18	Air exhaust 1 [target]	Set-point
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR20_ST_1-AO20	Air exhaust 2 [target]	Set-point
ISP_Level_03-103010_Raum_45_Temp_Sollwertverstellung-AV32	Temp. set-point	Set-point
ISP_Level_03-103010_Raum_45_Lueftung-MSV14	Ventilation set-point	Set-point
ISP_Level_03-103010_Raum_45_LQ_Sollwert-AV35	Air quality set-point	Set-point
DAIKIN MasterStation III No 100050-100050_TempAdjust_046-AV11786	Temp. s-p adjust. 1	Set-point
DAIKIN MasterStation III No 100050-100050_TempAdjust_047-AV12042	Temp. s-p adjust. 2	Set-point

Table A-3: Main data points associated to room 3.45

A-3 Application of BUDO scheme to data points

Tables A-4, A-5 and A-6 show the data points from the weather station and rooms 3.01 and 3.45, together with their translation into the BUDO scheme.

Original data point ID	BUDO tag
ISP_Level_02-102010_WS_Windgeschwindigkeit-AV34	B-HBK§WST_SEN+WIND.SPE
ISP_Level_02-102010_WS_Windrichtung-AV35	B-HBK§WST_SEN+WIND.DRCN
ISP_Level_02-102010_WS_Temperatur-AV36	B-HBK§WST_SEN+T
ISP_Level_02-102010_WS_Globalstrahlung_N-AV43	B-HBK§WST_SEN+SOL.RADI-N
ISP_Level_02-102010_WS_Globalstrahlung_O-AV44	B-HBK§WST_SEN+SOL.RADI-E
ISP_Level_02-102010_WS_Globalstrahlung_S-AV45	B-HBK§WST_SEN+SOL.RADI-S
ISP_Level_02-102010_WS_Globalstrahlung_W-AV46	B-HBK§WST_SEN+SOL.RADI-W
ISP_Level_02-102010_WS_Globalstrahlung_Sky-AV47	B-HBK§WST_SEN+SOL.RADI-SKY

Table A-4: Weather Station data points translated into BUDO tags

Original data point ID	BUDO tag
ISP_Level_03-103010_HBK_1a_03_L_RLT01_03_090_LQF01_MW_3-AI55	B-HBK_R+LIV.HALL-301§SEN+T-ROOM_MEA+T
ISP_Level_03-103010_HBK_1a_03_L_RLT01_03_090_LQF01_MW_1-AI53	B-HBK_R+LIV.HALL-301§SEN+AQ_MEA+CO2
Level0-smartdirector-D01-100040_1923402:3.01_HBK Ceiling Temperature-AI1923402	B-HBK_R+LIV.HALL-301§SEN+T-CEIL_MEA+T
Level0-smartdirector-D01-100040_1923403:3.01_HBK Power-AI1923403	B-HBK_R+LIV.HALL-301§MET+EL_MEA+POW
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR13_RW_1-AI13	B-HBK_R+LIV.HALL-301§CTRL.VF-13_MEA+VF
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR15_RW_1-AI15	B-HBK_R+LIV.HALL-301§CTRL.VF-15_MEA+VF
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR14_RW_1-AI14	B-HBK_R+LIV.HALL-301§CTRL.VF-14_MEA+VF
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR16_RW_1-AI16	B-HBK_R+LIV.HALL-301§CTRL.VF-16_MEA+VF
Level0-smartdirector-D01-100040_1923405:3.01_HBK Occupancy-BI1923405	B-HBK_R+LIV.HALL-301§SEN+PRES_MEA+PRES
ISP_Level_03-103010_Raum_01_Licht-MSV15	B-HBK_R+LIV.HALL-301§LI_COM.POS+LI.INT+7.STEP
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR13_ST_1-AO13	B-HBK_R+LIV.HALL-301§CTRL.VF-13_COM.POS+VF+SP
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR15_ST_1-AO15	B-HBK_R+LIV.HALL-301§CTRL.VF-15_COM.POS+VF+SP
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR14_ST_1-AO14	B-HBK_R+LIV.HALL-301§CTRL.VF-14_COM.POS+VF+SP
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_010_VVR16_ST_1-AO16	B-HBK_R+LIV.HALL-301§CTRL.VF-16_COM.POS+VF+SP
ISP_Level_03-103010_Raum_01_Temp_Sollwertverstellung-AV6	B-HBK_R+LIV.HALL-301§AHU_COM.POS+T+SP.ADJ
ISP_Level_03-103010_Raum_01_Lueftung-MSV1	B-HBK_R+LIV.HALL-301§AHU_COM.POS+VF+3.STEP
ISP_Level_03-103010_Raum_01_LQ_Sollwert-AV34	B-HBK_R+LIV.HALL-301§AHU_COM.POS+AQ+SP
DAIKIN MasterStation III No 100050-100050_TempAdjust_039-AV9994	B-HBK_R+LIV.HALL-301§AHU_COM.POS+T+SP+1
DAIKIN MasterStation III No 100050-100050_TempAdjust_045-AV11530	B-HBK_R+LIV.HALL-301§AHU_COM.POS+T+SP+2

Table A-5: Room 3.01 data points translated into BUDO tags

Original data point ID	BUDO tag
ISP_Level_03-103010_HBK_1a_03_L_RLT01_03_450_LQF02_MW_3-AI58	B-HBK_R+OFFI.OP-345§SEN+T-ROOM_MEA+T
ISP_Level_03-103010_HBK_1a_03_L_RLT01_03_450_LQF02_MW_1-AI56	B-HBK_R+OFFI.OP-345§SEN+AQ_MEA+CO2
Level0-smartdirector-D01-100040_1927902:3.45_Co Working Ceiling Temperature-AI1927902	B-HBK_R+OFFI.OP-345§SEN+T-CEIL_MEA+T
Level0-smartdirector-D01-100040_1927903:3.45_Co Working Power-AI1927903	B-HBK_R+OFFI.OP-345§MET+EL_MEA+POW
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR17_RW_1-AI17	B-HBK_R+OFFI.OP-345§CTRL.VF-17_MEA+VF
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR19_RW_1-AI19	B-HBK_R+OFFI.OP-345§CTRL.VF-19_MEA+VF
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR18_RW_1-AI18	B-HBK_R+OFFI.OP-345§CTRL.VF-18_MEA+VF
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR20_RW_1-AI20	B-HBK_R+OFFI.OP-345§CTRL.VF-20_MEA+VF
Level0-smartdirector-D01-100040_1927905:3.45_Co Working Occupancy-BI1927905	B-HBK_R+OFFI.OP-345§SEN+PRES_MEA+PRES
ISP_Level_03-103010_Raum_45_Licht-MSV28	B-HBK_R+OFFI.OP-345§LI_COM.POS+LI.INT+7.STEP
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR17_ST_1-AO17	B-HBK_R+OFFI.OP-345§CTRL.VF-17_COM.POS+VF+SP
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR19_ST_1-AO19	B-HBK_R+OFFI.OP-345§CTRL.VF-19_COM.POS+VF+SP
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR18_ST_1-AO18	B-HBK_R+OFFI.OP-345§CTRL.VF-18_COM.POS+VF+SP
ISP_Level_03-103010_HBK_1a_03_L_RLT03_03_450_VVR20_ST_1-AO20	B-HBK_R+OFFI.OP-345§CTRL.VF-20_COM.POS+VF+SP
ISP_Level_03-103010_Raum_45_Temp_Sollwertverstellung-AV32	B-HBK_R+OFFI.OP-345§AHU_COM.POS+T+SP.ADJ
ISP_Level_03-103010_Raum_45_Lueftung-MSV14	B-HBK_R+OFFI.OP-345§AHU_COM.POS+VF+3.STEP
ISP_Level_03-103010_Raum_45_LQ_Sollwert-AV35	B-HBK_R+OFFI.OP-345§AHU_COM.POS+AQ+SP
DAIKIN MasterStation III No 100050-100050_TempAdjust_046-AV11786	B-HBK_R+OFFI.OP-345§AHU_COM.POS+T+SP+1
DAIKIN MasterStation III No 100050-100050_TempAdjust_047-AV12042	B-HBK_R+OFFI.OP-345§AHU_COM.POS+T+SP+2

Table A-6: Room 3.45 data points translated into BUDO tags

A-4 Deep look into the structure of the data points

This Section of the Appendix serves the purpose to analyse the information contained within the tags presented in Table 4-1. This will show whether the presented data points have enough information to be caught with the first phase of the data grouping algorithm efficiently.

Type A data points

Fig. A-1 shows the general structure of the Type A data points, with the information that is included within.

Figure A-1: Type A data point general structure

This tag includes information about what the data point contains, but the code does not show any clue of which room it belongs to, it is a simple enumeration code. Because of the amount of data points with this structure, it would take a long time to figure out the zones to which each of these data points belong, so these are not considered in the data grouping algorithm.

Type B data points

Figures A-2, A-3 and A-4 show the general structure of the Type B data points (separated in types B.1, B.2 and B.3) with the information that is included within.

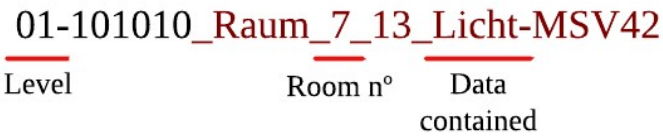

Figure A-2: Type B.1 data point general structure

Figure A-3: Type B.2 data point general structure

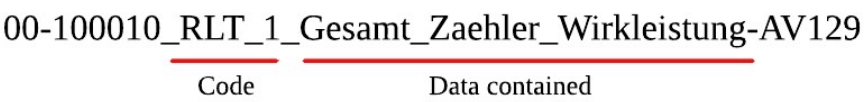

Figure A-4: Type B.3 data point general structure

Type B.1 and B.2 data points do show both room number and data contained, so these can be easily caught by the data grouping algorithm. In the example for Type B.1 we can see the string "Licht", which means "Light" in German; and the string "Raum", which stands for "Room". Moreover, as assumed by the information provided by the operator for rooms 3.01 and

3.45, "VVR" stands for "Variable Volume Ratio" and is used to define the data points related to the Variable Air Volume controller. Together with the following string "ST" for Set-points and "RW" for real Inputs, we can catch and define Type B.2 data points using BUDO tags.

Type B.3 data points do not show the room number, but they show the number of a specific Air Handling Unit, as shown by the German abbreviation RLT, which stands for "Raumlufttechnik", referring to the ventilation system. Each RLT has been chosen to be considered another zone to take into account in the parameter identification. Lastly, "Gesamt Zaehler Wirkleistung" stands for "total counter active power".

Type C data points

Fig. A-5 shows the general structure of the Type C data points, with the information that is included within. Type C data points clearly show the data contained within as well as the room number they belong to, so they can be caught efficiently by the data grouping algorithm.

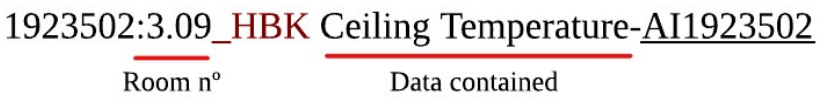

Figure A-5: Type C data point general structure

Type D data points

Fig. A-6 shows the general structure of the Type D data points, with the information that is included within.

Figure A-6: Type D data point general structure

Type D data points contain clear information of the data contained within, but do not show the room number where they belong. However, they include codes that are repeated several times and, according to the information in the server, represent the different RLTs in the building. The RLTs are also considered as zones in this thesis since it could be interesting to analyse them too. The codes and the associated RLT are shown below:

- Code 100060: RLT01
- Code 105010: RLT02
- Code 105020: RLT03
- Code 105030: RLT04
- Code 100080: RLT06
- Code 100090: RLT07
- Code 100100: RLT08

A-5 Data point grouping algorithm codes

In this Section of the Appendix, the tag-transformation (Code A.1) and data point-grouping (Code A.2) codes created using Perl programming language are displayed. These are further explained in Chapter 4.

Listing A.1: First phase: Perl code to transform tags

```
use warnings;
use strict;
use Data::Dump qw(dd);

my $input = 'HBK_all_datapoints.txt';              open(IN,'<',$input) or die $!;
my $output = 'HBK_datapoints_simplified.txt';      open(OUT,'>',$output) or die $!;

while (<IN>) {
    chomp;

   if ( my ($level,$code1,$code2,$room_num,$state,$tag,$id) = /0([0-9]+)\-([0-9]+)_([^_]+)_([0-9]+)_([^_]+)_([^_]+)\-([^_]+)/ ) {

     # Gets tags with structure like: ISP_Level_0//1-101010_Raum_09_Temp_Sollwert-AV19//

     $tag = "T_SP" if ( $state =~ /Temp/ && $tag =~ /Sollwert/ );
     $tag = "AQ_SP" if ( $state =~ /LQ/ && $tag =~ /Sollwert/ );

     print OUT "$level$room_num\§$tag\_$id";
    }

   elsif ( ($level,$id,$code2,$room_num,$tag,$id) = /0([0-9]+)\-([0-9]+)_([^_]+)_([0-9]+)_([^_]+)\-([^_]+)/ ) {

     # Gets tags with structure like: ISP_Level_0//1-101010_Raum_10_Lueftung-MSV7//

     $tag = "VF_SP" if ($tag =~ /Lueftung/);
     $tag = "LI_STEP" if ($tag =~ /Licht/);

     print OUT "$level$room_num\§$tag\_$id";
    }

    elsif ( ($level,$room_num,$tag) = /\:([0-9]+)\.([0-9]+)_([^_]+)/ ) {

     # Gets tags with structure like: Level0-smartdirector-D01-100040_1923403//:3.01_HBK Power//-AI1923403

     $tag = "SEN\+T_MEA\+T" if ($tag =~ /Ceiling Temperature/);
     $tag = "MET\+EL_MEA\+EL" if ($tag =~ /Power/);
```

```
    $tag = "MEA\+PRES_MEA\+PRES" if ($tag =~ /Occupancy/);

    print OUT "$level$room_num\§$tag";
  }

elsif ( ($level,$room_num,$tag,$id,$code1) = /0([0-9]+)_([0-9]+)0_([^_]+)_([^_]+)_1\-([^_]+)/ ) {

    # Gets tags with structure like: ISP_Level_03-103010_HBK_1a_03_L_RLT03_//03_010_VVR14_ST_1-AO14//

    $tag = "MEA\+VF" if ($tag =~ /VVR/ && $id =~ /RW/);
    $tag = "VF_SP" if ($tag =~ /VVR/ && $id =~ /ST/);

    print OUT "$level$room_num\§$tag\_$code1";
  }

elsif ( ($level,$code1,$tag,$id) = /pCOWeb([0-9]+)\-([0-9]+)_([^_]+)_([^_]+)/ ) {

    # Gets tags with structure like: //pCOWeb100090-100090_Temperature_Exhaust//_Air-AV4

    $tag = "RLT01§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /100060/ && $id =~ /Exhaust/);
    $tag = "RLT01§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /100060/ && $id =~ /Supply/);
    $tag = "RLT02§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /105010/ && $id =~ /Exhaust/);
    $tag = "RLT02§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /105010/ && $id =~ /Supply/);
    $tag = "RLT03§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /105020/ && $id =~ /Exhaust/);
    $tag = "RLT03§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /105020/ && $id =~ /Supply/);
    $tag = "RLT04§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /105030/ && $id =~ /Exhaust/);
    $tag = "RLT04§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /105030/ && $id =~ /Supply/);
    $tag = "RLT06§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /100080/ && $id =~ /Exhaust/);
    $tag = "RLT06§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /100080/ && $id =~ /Supply/);
    $tag = "RLT07§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /100090/ && $id =~ /Exhaust/);
    $tag = "RLT07§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /100090/ && $id =~ /Supply/);
    $tag = "RLT08§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /100100/ && $id =~ /Exhaust/);
    $tag = "RLT08§SEN\+T_MEA\+T" if ($tag =~ /Temperature/ && $level =~ /100100/ && $id =~ /Supply/);

    $tag = "RLT01§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /100060/ && $id =~ /Exhaust/);
    $tag = "RLT01§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /100060/ && $id =~ /Supply/);
    $tag = "RLT02§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /105010/ && $id =~ /Exhaust/);
```

```
    $tag = "RLT02§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /105010/ && $id =~ /Supply/);
    $tag = "RLT03§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /105020/ && $id =~ /Exhaust/);
    $tag = "RLT03§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /105020/ && $id =~ /Supply/);
    $tag = "RLT04§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /105030/ && $id =~ /Exhaust/);
    $tag = "RLT04§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /105030/ && $id =~ /Supply/);
    $tag = "RLT06§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /100080/ && $id =~ /Exhaust/);
    $tag = "RLT06§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /100080/ && $id =~ /Supply/);
    $tag = "RLT07§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /100090/ && $id =~ /Exhaust/);
    $tag = "RLT07§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /100090/ && $id =~ /Supply/);
    $tag = "RLT08§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /100100/ && $id =~ /Exhaust/);
    $tag = "RLT08§MEA\+VF" if ($tag =~ /Volume/ && $level =~ /100100/ && $id =~ /Supply/);

    print OUT "$tag\_$id";
}

elsif ( ($level,$code1,$tag,$id,$code2) = /pCOWeb([0-9]+)\-([0-9]+)_([^_]+)_([^_]+)_([^_]+)/ ) {

    # Gets tags with structure like: pCOWeb//105010-105010_Signal_Valve_Cooling//-AV29

    $tag = "RLT01§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /100060/ && $code2 =~ /Heating/);
    $tag = "RLT01§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /100060/ && $code2 =~ /Cooling/);
    $tag = "RLT02§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /105010/ && $code2 =~ /Heating/);
    $tag = "RLT02§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /105010/ && $code2 =~ /Cooling/);
    $tag = "RLT03§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /105020/ && $code2 =~ /Heating/);
    $tag = "RLT03§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /105020/ && $code2 =~ /Cooling/);
    $tag = "RLT04§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /105030/ && $code2 =~ /Heating/);
    $tag = "RLT04§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /105030/ && $code2 =~ /Cooling/);
    $tag = "RLT06§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /100080/ && $code2 =~ /Heating/);
    $tag = "RLT06§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /100080/ && $code2 =~ /Cooling/);
    $tag = "RLT07§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /100090/ && $code2 =~ /Heating/);
    $tag = "RLT07§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /100090/ && $code2 =~ /Cooling/);
    $tag = "RLT08§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /100100/ && $code2 =~ /Heating/);
    $tag = "RLT08§VAL_COM\.POS\+VF" if ($id =~ /Valve/ && $level =~ /100100/ && $code2 =~ /Cooling/);

    print OUT "$tag\_$code2";
}
```

```
   elsif ( ($code1,$tag) = /RLT_([0-9]+)_Gesamt_Zaehler_([^_]+)/ ) {

      # Gets tags with structure like: ISP_Level_00-100010_//RLT_1_Gesamt_Zaehler_Wirkleistung//-AV129

      $tag = "RLT01§MET\+EL_MEA\+EL" if ($code1 =~ /1/ && $tag =~ /Wirkleistung/);
      $tag = "RLT02§MET\+EL_MEA\+EL" if ($code1 =~ /2/ && $tag =~ /Wirkleistung/);
      $tag = "RLT03§MET\+EL_MEA\+EL" if ($code1 =~ /3/ && $tag =~ /Wirkleistung/);
      $tag = "RLT04§MET\+EL_MEA\+EL" if ($code1 =~ /4/ && $tag =~ /Wirkleistung/);

      print OUT "$tag";
   }

   else {
      print OUT $_;
   }
   print OUT "\n";
}

close(IN); close(OUT);
```

Listing A.2: Second phase: Perl code to group data points

```
use warnings;
use strict;
use Data::Dump qw(dd);

my $input = 'HBK_datapoints_simplified.txt';  open(IN,'<',$input) or die $!;
my $output = 'PI_input.txt';                  open(OUT,'>',$output) or die $!;

# ------------------------------1.GET TAGS FROM THE ORIGINAL DATA POINT FILE-----------------------------
my ( %states , %room , @wst );

while (<IN>) {
    chomp;
    if ( my ($room_num,$tag) = /([0-9]+)§([^_]+)/ ) {
      $room{$room_num}{$_} = $.;

      if( my $tag = /SEN\+T|SEN\+AQ|MET\+EL/ ) {
         $states{$room_num}{$_} = $.;
      }
    }

    if ( $_ = /_WST_/ ) {
      push @wst, $.;
   }
}

# -------------------------------------2.CREATE THE OUTPUT HASHES-------------------------------------
my ( %outformat , %inputs , %setpoints );

while( my($key, $value) = each %room ) {
   while( my($kii, $vii) = each %{ $states{$key} } ) {
      while( my($ki, $vi) = each %{ $room{$key} } ) {
         push @{ $outformat{$key}{$vii} }, $vi if ( $vi ne $vii );
         push @{ $inputs{$key}{$vii} }, $vi if ( ($vi ne $vii) && ($ki =~ /MEA/) );
         push @{ $setpoints{$key}{$vii} }, $vi if ( ($vi ne $vii) && ($ki =~ /SP|STEP|VAL/) );
      }
   }
```

```
}

# ------------------------3.PRINT INTO THE OUTPUT FILE WITH COMMA-SEPARATED VALUES-------------------------
foreach my $room_num (keys %outformat) {
   foreach my $state ( keys %{ $outformat{$room_num}} ) {
      if ( exists $setpoints{$room_num}{$state} && $inputs{$room_num}{$state} ) {
         print OUT ( join ',',
            $room_num , $state , @{$inputs{$room_num}{$state}} , @wst , @{$setpoints{$room_num}{$state}} );
         print OUT "\n";
      }

      elsif ( exists $inputs{$room_num}{$state} ) {
         print OUT ( join ',',
            $room_num , $state , @{$inputs{$room_num}{$state}} , @wst );
         print OUT "\n";
      }

      elsif ( exists $setpoints{$room_num}{$state} ) {
         print OUT ( join ',',
            $room_num , $state , @wst , @{$setpoints{$room_num}{$state}} );
         print OUT "\n";
      }

      else {
         print OUT ( join ',',
            $room_num , $state , @wst );
         print OUT "\n";
      }
   }
}

close(IN); close(OUT);
```

A-6 Row numbers of data points

In this Section of the Appendix, Tables A-7 and A-8 shows the data points from the weather station and 3.45 together with the row number they belong to in the original data point file. This Appendix supports the information of the plots in Section 5-4-1. The BUDO tags are used for more traceability to the data point functions. Row numbers 4764 and 4765 belong to temperature set-points that have been caught by the tag-transformation code.

Row nº	Data point (BUDO tag)
4422	B-HBK§WST_SEN+WIND.SPE
4423	B-HBK§WST_SEN+WIND.DRCN
4421	B-HBK§WST_SEN+T
4410	B-HBK§WST_SEN+SOL.RADI-N
4411	B-HBK§WST_SEN+SOL.RADI-E
4412	B-HBK§WST_SEN+SOL.RADI-S
4414	B-HBK§WST_SEN+SOL.RADI-W
4413	B-HBK§WST_SEN+SOL.RADI-SKY

Table A-7: Row numbers of data points from the weather station

Row nº	Data point (BUDO tag)
4500	B-HBK_R+OFFI.OP-345§SEN+T-ROOM_MEA+T
4498	B-HBK_R+OFFI.OP-345§SEN+AQ_MEA+CO2
6700	B-HBK_R+OFFI.OP-345§SEN+T-CEIL_MEA+T
6701	B-HBK_R+OFFI.OP-345§MET+EL_MEA+POW
4575	B-HBK_R+OFFI.OP-345§CTRL.VF-17_MEA+VF
4579	B-HBK_R+OFFI.OP-345§CTRL.VF-19_MEA+VF
4577	B-HBK_R+OFFI.OP-345§CTRL.VF-18_MEA+VF
4581	B-HBK_R+OFFI.OP-345§CTRL.VF-20_MEA+VF
6703	B-HBK_R+OFFI.OP-345§SEN+PRES_MEA+PRES
4759	B-HBK_R+OFFI.OP-345§LI_COM.POS+LI.INT+7.STEP
4576	B-HBK_R+OFFI.OP-345§CTRL.VF-17_COM.POS+VF+SP
4580	B-HBK_R+OFFI.OP-345§CTRL.VF-19_COM.POS+VF+SP
4578	B-HBK_R+OFFI.OP-345§CTRL.VF-18_COM.POS+VF+SP
4582	B-HBK_R+OFFI.OP-345§CTRL.VF-20_COM.POS+VF+SP
4766	B-HBK_R+OFFI.OP-345§AHU_COM.POS+T+SP.ADJ
4761	B-HBK_R+OFFI.OP-345§AHU_COM.POS+VF+3.STEP
4760	B-HBK_R+OFFI.OP-345§AHU_COM.POS+AQ+SP
4764	345§T_SP-AV14
4765	345§T_SP-AV33
3043	B-HBK_R+OFFI.OP-345§AHU_COM.POS+T+SP+1
3044	B-HBK_R+OFFI.OP-345§AHU_COM.POS+T+SP+2

Table A-8: Row numbers of data points from room 3.45

A-7 Plots from the server

This Appendix shows a series of plots that support the information of Section 5-4-2.

Figure A-7: Comparison of the 3.45 power state (blue) and the occupancy (green) graphically represented in the server for August 2021

Figure A-8: Comparison of the 3.45 power state (blue) and the occupancy (green) graphically represented in the server for December 2021

Figure A-9: Comparison of the 3.45 power state (blue) and the occupancy (green) graphically represented in the server for Week 16 of the year 2021

Appendix B

Attached digital files

Attached to the main document of this master thesis, a series of digital files supporting the information are included. These are the following:

1. HBK_all_datapoints.txt - Contains the mentioned original data point file, with all the 17,300 data points, row by row, from the studied building.
2. Datapoints_3.01and3.45.pdf - Contains the tables of the data points from rooms 3.01, 3.45 and weather station, provided by the operator with other useful information.
3. DataGrouping_algorithm.zip - Contains only the data point grouping algorithm implemented with Perl. To run it, Perl must be installed in the PC, and it must be run with the Perl programming language.
4. ParameterId_algorithm_bySONDEproject.zip - Contains the Matlab parameter identification algorithm. To run the algorithm, the file "mainPIinit.m" must be run. The data point grouping algorithm is integrated inside, in the folder named "Info", which means that this algorithm runs the whole process. After running the algorithm, the results can then be checked in MATLAB data file "Model.mat".
5. HamburgBuilding_data.zip - This compressed file contains 7 Excel spreadsheets with information from the operator related to the data points, which are useful to identify the function of each data point.
    - 200930_VSR_Gesamt_mit_AKS.xlsx - Information about data points related to volume airflow.
    - HBK_EDE-Files.xlsx - BACnet data from the server saved into the Engineering Data Exchange (EDE) file.
    - ISP_Level_00.xlsm to ISP_Level_04.xlsm - Provide lists of data points organized by type of values, together with some information that can be useful to identify the utility of the data points. One Excel spreadsheet per each level.

Bibliography

[1] F. M. of Education and Research, "Supervision and Optimization of New Buildings from Data Exploration - FaF Forschung an Fachhochschulen." [Online]. Available: https://www.forschung-fachhochschulen.de/fachhochschulen/shareddocs/projekte/en/fhprofunt/sonde.html

[2] L. Schnelle, G. Lichtenberg, and C. Warnecke, "Using low-rank multilinear parameter identification for anomaly detection of building systems," *11th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, 2022, status: accepted.

[3] L. Martirano and M. Mitolo, "Building automation and control systems (bacs): a review," in *2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I CPS Europe)*, 2020, pp. 1–8.

[4] D. Rigamonti, "Building automation and control system bacs - Designing Buildings, The Construction Wiki." [Online]. Available: https://www.designingbuildings.co.uk/wiki/Building_Automation_and_Control_System_BACS

[5] M. A. Niazi, "The layers of modern building automation system architecture - Control Automation." [Online]. Available: https://control.com/technical-articles/the-layers-of-modern-building-automation-system-architecture/

[6] N. Djuric and V. Novakovic, "Review of possibilities and necessities for building lifetime commissioning," *Renewable and Sustainable Energy Reviews*, vol. 13, no. 2, pp. 486–492, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032107001591

[7] G. Pangalos, A. Eichler, and G. Lichtenberg, "Tensor systems : multilinear modeling and applications," in *3rd International Conference on Simulation and Modeling Methodologies, Technologies and Applications, SIMULTECH, Reykjavik, Iceland, 29 - 31 July, 2013*, 2013, pp. 275–285. [Online]. Available: http://hdl.handle.net/11420/5869

[8] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009. [Online]. Available: https://doi.org/10.1137/07070111X

[9] K. Kruppa, "Comparison of Tensor Decomposition Methods for Simulation of Multilinear Time-Invariant Systems with the MTI Toolbox," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 5610–5615, Jul. 2017.

[10] F. Stinner, A. Kornas, M. Baranski, and D. Müller, "Structuring building monitoring and automation system data," in *The REHVA European HVAC Journal - August 2018*, ser. The REHVA European HVAC Journal, REHVA, Ed., 2018, pp. 10–15. [Online]. Available: https://www.rehva.eu/fileadmin/user_upload/10-15_RJ1804_WEB.pdf

[11] "Vocabulary of budo - RWTH University - EBC - Last edited by: Florian Stinner." [Online]. Available: https://github.com/RWTH-EBC/BUDO/wiki/Documentation-BUDO-vocabulary

[12] "List of considered standards for the BUDO scheme - RWTH University - EBC - Last edited by: Florian Stinner." [Online]. Available: https://github.com/RWTH-EBC/BUDO/wiki/Considered-standards

[13] J. F. Butler and R. Veelenturf, "Point naming standards - ASHRAE Journal, 2010." [Online]. Available: http://www.bacnet.org/Bibliography/BACnet-Today-10/Butler_2010.pdf

[14] F. Stinner, P. Neißer-Deiters, M. Baranski, and D. Müller, "Aikido: Structuring data point identifiers of technical building equipment by machine learning," *Journal of Physics: Conference Series*, vol. 1343, p. 012039, 2019. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1343/1/012039/pdf

[15] A. V. AHO, "Chapter 5 - algorithms for finding patterns in strings," in *Algorithms and Complexity*, ser. Handbook of Theoretical Computer Science, J. VAN LEEUWEN, Ed. Amsterdam: Elsevier, 1990, pp. 255–300. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444880710500102

[16] M. Sičák, "Higher order regular expressions," in *2015 13th International Conference on Engineering of Modern Electric Systems (EMES)*, 2015, pp. 1–4.

[17] J. E. Hopcroft, R. Motwani, and J. D. Ullman, in *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Ed., 1979.

[18] "Perl documentation - Perl.org official webpage." [Online]. Available: https://perldoc.perl.org

[19] "Perl programming language official webpage." [Online]. Available: https://www.perl.org/

Acronyms

BACS	Building Automation and Control System
BUDO	Building Unified Data point naming schema for Operation management
CP	Canonycal Polyadic
ISO	International Organization for Standardization
MTI	Multilinear Time-Invariant
RegExp	Regular Expression
SONDE	Supervision and Optimization of New Buildings with Data Exploration