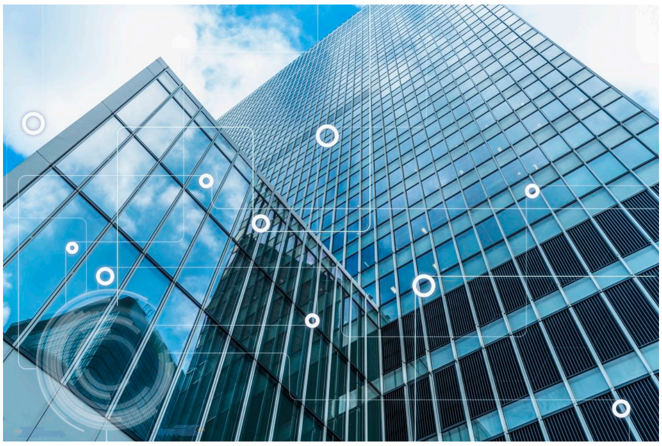
Evaluation of linear regression and neural network methods for estimating occupancy in office buildings using bGrid sensor data



Thesis

to obtain the degree of Master of Science
at Delft University of Technology,
to be defended publicly on October 7^{th}, 2020 at 14:00.

by

J.J. Kraaijeveld

Student number: 4089316

This thesis was supervised by

Prof. Dr. B. De Schutter
Dr. L.A. de Araujo Passos

Commitee:

Prof. Dr. B. De Schutter ,	Delft University of Technology
Dr. L.A. de Araujo Passos ,	Delft University of Technology
Dr. Ir. K. Batselier ,	Delft University of Technology

Keywords: office occupancy estimation, linear regression, artificial neural networks

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

Abstract

This thesis outlines the use of measured data collected using the bGrid system to estimate the number of people in two rooms in the Microsoft office at Schiphol. The main objective is to derive a correlation that transforms the data into a specific number of people.

The bGrid system consists of a network of sensor nodes that measure CO_2 concentration, movement intensity, relative humidity, ambient temperature, infrared object temperature and sound intensity. These nodes are strategically placed throughout the building and are interconnected through a gateway that is also part of the bGrid system. The specific offices used had an equal area of roughly 16 m^2, a capacity for eight people and contained two bGrid sensor nodes each. Data was collected on two days, separated by one day, yielding 1396 minutes of data. Ground truth data was collected using direct observation from outside the offices as to not interfere with the measurements. Since the offices were of equal area and capacity, were both used for the same purpose and each contained two sensor nodes, the rooms were deemed equivalent and could therefore be combined. On the first day of observation, only room 1 could be observed but on the second day both rooms could be observed. The data from room 1 was used to synthesise occupancy models while the data from room 2 was used for validation.

Two methods were used to model the occupancy. An approach based on Multiple Linear Regression used a combination of the movement intensity and CO_2 concentration to achieve a performance of 93.49%, with performance being defined as the times the model returns a number of people that is within one person of the observed occupancy expressed as a percentage of the total number of iterations. This was achieved by first splitting the data based on the observed occupancy to derive specific regression coefficients for those levels of occupancy. These were used in a Simulink model that actively chose which regression coefficient to use based on the input. Secondly a method using a three-layer, fully connected neural network with a combination of hyperbolic tangent and leaky ReLu activations functions used the ambient temperature, relative humidity, movement intensity and CO_2 data to achieve a performance of 93.86%. This was achieved using a network structure with three hidden layers with 19, 21 and 39 neurons respectively. These numbers were derived using the genetic algorithm to optimise for 43 iterations with the number of neurons per hidden layer and the learning rate as optimisation variables.

The final result yielded two models that were able to estimate the number of people in room 2 within one person of the observed number of people, respectively 93.49% and 93.86% of the time. The resulting models could, when verified further with additional data, be used to aid HVAC systems with ambient temperature control; having a reliable metric for occupancy allows people to be added to the energy balance of a room, which in turn allows for the creation of more accurate models of the indoor thermal climate. These models would allow for model based temperature controllers as opposed to PID type controllers that are currently used in most office thermostats.

Preface

Everyone who has ever produced a thesis or dissertation knows the preface is an optional addition that is quite often omitted. It can contain personal information about the writer that lead to the research that is presented and/or how the writer's background and experience relate to the subject that is being researched. It may also give some information about the intended audience. Knowing this, it is clear why it is often omitted in theses. The reasons this research topic was chosen can usually be summarised as "I had to find a graduation subject and this opportunity presented itself". The relationship between the writer's background and experience and the subject of research is usually also not too difficult to discern since the thesis subject will be closely related to the master program for which it is written. And finally, the intended audience usually does not need to be specified. A master thesis is not a novel that some passerby can grab, read the abstract and think "nope, this is not for me". Generally the only people reading master theses are the people intended to read those master theses. However, having already dedicated so many words to the questionable existence of the preface it would be a shame not to continue and actually finish this chapter. So here goes; following the standard approach, the following paragraphs will tell how I landed on the subject for this thesis, how my background and experience relate to the subject and a description of the intended audience.

Some time ago I was invited to talk to a representative from bGrid solutions at De Delftse Bedrijvendagen. She introduced me to Wouter Kok, executive at bGrid solutions, to talk about a possible thesis subject. bGrid solutions is a company specialising in smart building technology so it had to be possible to find research that was relevant for the company while also offering an academic standard that would enable me to graduate. It was a lengthy process with ups, downs, more ups and definitely more downs but in the end the subject that resulted in this thesis was found. A subject that I would not have chosen if I had known then what I know now but a subject that lead to a process that has taught me valuable lessons nonetheless. I can therefore look back without regret (if and only if I obtain the degree of Master of Science with this thesis) and am pleased with the results.

I find it difficult to describe how my personal background and experience has lead to this thesis. One thing that always interested me in the field of systems and control is how applicable the theory is once reality has been mathematically modelled. Once the world is transformed into a system of equations, reality dissolves into something that anyone with sufficient knowledge on system theory, mathematics and control theory should be able to understand. This realisation is what lead me to think that the area in which I performed my research should not matter, as long as I was able to transform it into something that I could understand. With the benefit of hindsight, I stand by this claim though it is a little more nuanced than I initially thought. Specific knowledge of the area that acts as the foundation for research always helps, even if the research itself is highly mathematical. This meant it took quite some time for this thesis to reach a point that I would call "fun", leading to delays that may not have occurred if a different path was chosen in the beginning.

This thesis was attempted to be written in a way that would make it accessible to anyone with a background in control engineering, mathematics and/or systems and control. Additionally the contents should be interesting for data analysts and people working with smart building technology. It should also be mildly enjoyable for anyone with a technical background or people who know me personally, though I assume for the latter the joy received from reading this will gradually decrease in the following chapters if the only reason for reading is their relationship to me. Regardless of who you are, my hope is that at the end of this thesis you will have a clear understanding of what I achieved and how I achieved it. If that is not the case, you can always say that this thesis described how I obtained the degree of Master of Science and with any luck, you will be correct.

Acknowledgements

Before diving into the research, some acknowledgements are in order. First and foremost this research would not have been possible without Luigi de Araujo Passos and Bart De Schutter from Delft University of Technology, who provided excellent supervision and made sure the thesis was held to an academic standard.

The same can be said for the help and assistance of Wouter Kok, Kees van Grieken and Sven Kraaijevanger from bGrid solutions. It goes without saying that the data bGrid provided was the backbone of this research, but additionally the people mentioned also offered support and guidance which was at least equally valuable.

A small word of gratitude goes out to the people at the Microsoft office at Schiphol for their support and tolerance during the gathering of ground truth data. The people there kindly opened their office, assisted in finding suitable rooms and allowed the observations to take place during normal office activity.

Finally special thanks to María García Fernández for various contributions and understanding.

Contents

1 Introduction

A revolutionary development in building technology is the introduction of intelligent buildings (also referred to as smart buildings). The Intelligent Building Institute (IBI) in the United States and the European Intelligent Building Group (EIBG) based in the United Kingdom propose the most accepted definitions for what constitutes an intelligent building. The IBI defines an intelligent building as *'one which provides a productive and cost-effective environment through optimization of its four basic elements including structures, systems, services and management and the interrelationships between them'*, while EIBG defines an intelligent building as *'one that creates an environment which maximizes the effectiveness of the building's occupants, while at the same time enabling efficient management of resources with minimum life-time costs of hardware and facilities'* [37]. Including smart building technology in a structure enables a deeper level of control than would be possible without this technology. Some examples are adaptive lighting that responds to real time occupancy, adaptive ventilation and ambient temperature control. But is this development forming just because people are unable to manage their environment themselves? Shouldn't people be able to adjust thermostats and ventilation controls when they're uncomfortable? The reason humans often fail in this aspect is because the level of thermal comfort is unlikely to be equal across multiple people because of all different factors that influence thermal comfort. People all have their own personal threshold for thermal comfort that depends not just on the ambient temperature. Humidity plays a big role as well, influencing how the ambient temperature is experienced. Additionally, humans are notoriously untrustworthy thermometers. If temperature changes occur gradually (less than 0.5°C per minute) people can remain unaware of a 4-5°C change in ambient temperature, if the skin remains within the neutral thermal region of 30-36°C [57].

For this and other reasons there is an increasing demand for a method to approach ambient temperature management in a more intelligent manner. Currently most buildings' ambient temperature control is set up such that the building is at the desired temperature at the start of the day but the Heating, Ventilation and Air-Conditioning (HVAC) system spends the rest of the day cooling to counteract the thermal energy added by the occupants and the outdoor environment. From a Systems and Control point of view the HVAC management side of the problem seems the most interesting, especially since buildings are responsible for roughly 38% of the total energy consumption globally [35, 59]. Granted, this includes both the commercial sector as the residential sector. Looking at the commercial sector alone the contribution remains big with heating, cooling and lighting being the largest contributors, taking the United states and Japan as an example, at 58% of a building's total energy in the United States and 69% in Japan [59]. Research has shown that energy-related occupant behaviour is both unpredictable and a big contributing factor to a building's energy consumption. Often occupant behaviour is represented using standardised schedules that can lead to simulation inaccuracies [21]. Since the impact of a building's occupants is so big and seemingly unpredictable this would be a more valuable area for research and is what this thesis hopes to cover. In the early stages of the literature review some time was spent researching the different methods that were previously used to design temperature controllers to maximise user comfort in a multi-sensor setting. Lacking knowledge of the occupancy of the inspected room, this idea was modified. These papers will be used as a basis for possible future work in chapter 7.

As mentioned the impact of occupants to the indoor climate of a building is significant. Research has shown that a single person can add up to 120W of thermal power to the indoor environment with light office work, a value close to the thermal output of a small office printer returning one page every minute [12, 19]. Additionally, from a building management point of view, knowing when and to what extent offices are occupied enables office managers to map out the exact utilisation of a building's resources on a day to day basis. A 2013 paper by Norm G. Miller showed bigger companies are transitioning to smaller office footprints to achieve higher utilisation rates [39]. The struggle is that the need for collaboration and innovation is working against this goal, forcing companies to retain meeting rooms and collaboration areas to facilitate this need. Because of this, it becomes vital that those retained spaces are used optimally, which would demand accurate knowledge of the occupation.

This, along with people's impact on the indoor climate are some of the reasons why it would be interesting to know exactly how many people occupy a specific room at any given time. The method that first comes to mind would be to use image recognition software and high resolution cameras to capture everything that happens inside the office. One can imagine the complications with respect to privacy however, so a method that uses measured data like motion detection, CO_2 concentration, humidity or temperature instead of images would be preferable. This leads to the following problem statement:

Is it possible to accurately estimate the occupancy of a room using measured data such as CO_2 and movement without using cameras?

The answer to the question stated above is not immediately evident. Based on the impact people have on the indoor environment, it is likely that the effects are measurable. If that is indeed the case, the effect of human presence and behaviour will show itself in measured data. The data is gathered with the help of bGrid Solutions, a company resulting from a joint venture between Evalan in Amsterdam and Deerns in Rijswijk. bGrid Solutions is an emerging company that specialises in delivering smart building technology to offices and other large buildings. Their goal is to equip buildings with integrated control systems for managing a variety of processes that are linked via a network, but additionally also have the ability to learn about the environment and the occupants to adapt the control accordingly. Their system, which will get an in-depth review in chapter 2, consists of a network of sensor nodes capable of capturing a wide variety of information about their environment.

Using the measured data, numerical methods will be used to find a correlation between the measurements and the ground truth data gathered through direct observation. The methods this thesis will outline are based on linear regression and artificial neural networks, both because these methods are widely used in data based modelling and because both have been successful in research that will be presented in chapter 3.

This thesis is structured as follows: after the introduction, the second chapter will cover the bGrid system and outline its different components and attributes. This chapter will also highlight the main merits and challenges that the bGrid system may bring. Chapter three covers literature on both the thermal comfort versus energy consumption problem and the occupancy estimation problem. Chapter four dives deeper into linear regression and artificial neural networks which, based on the literature, are thought to provide a solid foundation for this research. The fifth chapter covers the experimental analysis that starts with collecting ground truth data through direct observation and methods for managing the sensor data. It continues by presenting and performing the methods for estimating the occupancy and finishes by summarising the results. Chapter six will wrap things up by comparing the results from chapter three to the literature and the thesis closes with chapter seven where some suggestions for future work are made.

2

The bGrid System

In order to judge the merit of different research results in the context of what can be achieved with the bGrid sytem, a close look will have to be taken at its different attributes and components. In this chapter the bGrid system will be broken down and looked at to see what possibilities and challenges this particular system might bring when it comes to ambient temperature control and more importantly, occupancy estimation. Before the bGrid system is discussed, some background information about the company itself is in order. As mentioned in the introduction, bGrid Solutions (bGrid in short) is a company resulting from a joint venture between Evalan in Amsterdam and Deerns in Rijswijk that was founded in 2015. Their goal is to equip a building with the kind of technology that allows it to communicate with its users and vice versa. Their products all revolve around IoT (Internet of Things) which is the technology that connects seemingly unrelated devices to the same network, allowing for example a smartphone to connect to both a fridge, a dishwasher, a car and the interior lighting. bGrid Solutions does state that the Internet of Things can quickly turn into the Internet of Too Many Things, which is why a big part of their objective is pairing the hardware they develop with intuitive software. Their system is centred around gathering vast amounts of data that other processes and devices can use. Some preliminary examples include ambient temperature, relative humidity and CO_2 concentration measurements that are fed to the HVAC system and movement detection that is connected to the lights. The next section will dive deeper into the individual components.

2.1. System Overview

The bGrid system is modular and envelops five distinct parts [5, 6, 7, 8] as shown in Figure 2.1. The parts shown in blue are part of the bGrid system while the other components in the figure refer to third party software and/or hardware.

- **Sensor node:** the bGrid system revolves around the sensor nodes that measure the relevant parameters present in the building such as movement intensity, sound intensity, CO_2 concentration, ambient light, relative humidity and temperature. Note that the nodes are not equipped with all the measuring capabilities by default. Some features like measuring the CO_2 concentration require specialised parts that introduce extra costs while the demand might not be there and are thus optional. The nodes use Bluetooth Low Energy (BLE) advertisement messages to send the sensor data to a gateway that in turn sends it to the backend application. The sensor nodes are also suited to be connected to lamps from third-party manufacturers to enable lighting control. The nodes can be integrated into the ceiling, mounted on the ceiling or underneath desks. The sensor nodes can be seen as agents as described by E. Bonabeau in a 2002 paper [9]. Agents are capable of individually assessing their environment and make decisions based on a predefined set of rules . As of yet the nodes don't control the building's HVAC system directly. However, since they do detect movement most Building Management Systems (BMS) use the nodes as a switch for the ventilation system. Table 2.1 lists an overview of bGrid functions and the range and accuracy of the different sensors.

Function	Unit	Frequency	Performance	Comments
Temperature Measurement (T)	°C	Every minute	Range -20 to 60 °C Accuracy 0.5 °C	*The measurement may differ from the actual temperature in the room for nodes that are placed in the ceiling near the heating/cooling duct or the lamp*
Presence Detection (PIR)		Continuous	Average response time <1 second	*Presence detection using a passive infrared sensor (PIR sensor) is not included in all nodes. The exact number of PIR-nodes must be determined in consultation with the client.*
Light Intensity Measurement (LI)	LUX	Every minute	Range 0.1 to 20000 LUX Accuracy 1 LUX	*Used as a relative reference since the accuracy of light intensity measurements at default are highly dependent on surface underneath.*
Relative Humidity Measurement RH)	%	Every minute	Range 0% to 100% Accuracy 5%	
Sound Intensity Measurement (SI)	-	Every minute	Depends on sound frequency	*Measures the volume of sound through a human comfort directed algorithm*
Beacon Mode		iBeacon		*Unique identification number (UUID) of the beacon can be entered via API.*
Scanning Mode		Continuous	Average detection time <10 seconds	*Detects BLE devices that send a signal, or on-purpose BLE asset tags.*
DALI				*Node acts as DALI bus master, instructions to lamp on/off/intensity adjustment.*
Light On/Off Switch			<1 second	*Triggered by in-network control or external action received via API.*
BACnet				*Gateway passes sensor data to BMS via BACnet.*
CO2	ppm	Every minute	Range 0 to 2000 ppm Accuracy 50 ppm	*Automated calibration during the night*
Over-The-Air Software Updates				*Software updates can be installed on the nodes wirelessly.*

Table 2.1: Specification of bGrid functions [7]

- **Gateway:** The bGrid gateway collects the sensor data from a fixed group of sensor nodes via BLE and sends it through an Internet Protocol (IP) to the backend application. The gateway can be used for direct control in some use cases but it is mostly used as a hub to collect sensor data from up to 25 nodes and forward the commands it receives from the backend application. One of the use cases in which the gateway is used for direct control is the light management. Here the gateway can switch the lights of a predefined group of nodes connected to light units on and off directly, based on the nodes' presence detection. The gateways are mounted on the ceiling so that signals can travel uninterrupted.

- **Backend application:** The backend application gathers data from multiple gateways via an IP connection, storing all data it receives on both the cloud server and the on-site server. The backend application is the connecting element in the bGrid system since it connects the gateways and sends commands via IP. The routines that require the data to perform algorithms acquire this from the backend application via the Application Programming Interface (API). These routines can send their commands through the API to the cloud server. The API also grants third parties access to the data and enables third parties to send commands to the cloud server.

- **On-site server:** Each building where the bGrid system operates contains 2 on-site servers, one of which serves as a backup to the other and contains a copy of all the data. The backup server can be configured to take over all functions should the main server fail.

- **Cloud server:** The cloud server is where the API is stored. It is hosted in a secured industrial datacenter.
- **Routines:** The routines process all algorithms and are communicated to the cloud server via an API.

Additionally the bGrid system is able to communicate with a variety of third-party components (the white squares in Figure 2.1), excluding the third-party DALI light because it is not relevant to the subject:

- **Third-party server:** Similar to the bGrid servers the third-party server communicates with the cloud server to request data from the bGrid system. This communication occurs via API in a User Interface (UI) or via third-party devices and can contain commands to the cloud server to adjust relevant parameters.
- **Third-party UI:** Third-parties can implement their own UI to interact with the bGrid system. This could for example be a tool to visualise the data and send commands through the third-party server.
- **Third-party device:** Third-parties can implement devices that are connected to their server and can receive and send data through that connection. They can also send BLE messages to and receive BLE messages from the bGrid sensor nodes.
- **Third-party BMS:** The BMS (Building Management System) is responsible for the control of all connected components (heating, ventilation etc.) of a building. The bGrid system integrates with the BMS such that it can use the information collected by the bGrid sensor nodes through the gateway for control. The BMS uses a one-way BACnet protocol [36] via IP to communicate with the gateway.
- **Third-party control Unit:** Third-parties can connect units for the control of for instance air conditioning to their BMS. The bGrid system is connected to the BMS, allowing it to collect building parameters through the bGrid system and send commands to the control units.

2.2. Conclusion

In the context of this project one of the biggest assets of the bGrid system is the sensor node. These devices are strategically placed throughout a building and generate a vast amount of data. The amount of nodes (25 per gateway as mentioned before) can be especially useful to eliminate outliers without losing the amount of data required for accurate modelling and control. It is not uncommon for nodes to be placed in a less strategic place or to be blocked in a later stage of a building's development by third-parties. This could result in anomalous sensor data which could throw off modelling techniques and control strategies if there were only a few nodes in the area. At this point it has to be noted that the overlap between nodes is minimal. On the one hand, environmental data such as CO_2 concentration, temperature and relative humidity are not expected to vary much between nodes that are placed in the same room, though this does rely somewhat on the total room volume and the distance between the nodes. For those data types an adjacent node could serve as a backup in a case of packet drops or bad reception. Looking on the other hand at sound intensity and movement intensity, this is no longer the case. The data collected by one node can be very different than the data collected by an adjacent node because the sensors do not cover the same area. As said, there is some overlap but this is minimal. This only becomes an issue when the reception is bad and/or when a lot of data is lost during transmission and it is therefore not guaranteed to cause problems.

The measurement range and accuracy of the nodes is another benefit (range -20°C to 60 and accuracy 0.5 as can be seen in Table 2.1) . Humans are notoriously untrustworthy thermometers in that the perceived temperature is influenced by more than just the ambient temperature. If temperature changes occur gradually (less than 0.5°C per minute) people can remain unaware of a 4-5°C change in ambient temperature, if the skin remains within the neutral thermal region of 30-36°C [57]. The resolution of the sensor nodes is therefore sufficiently high to pick up on any changes in temperature that

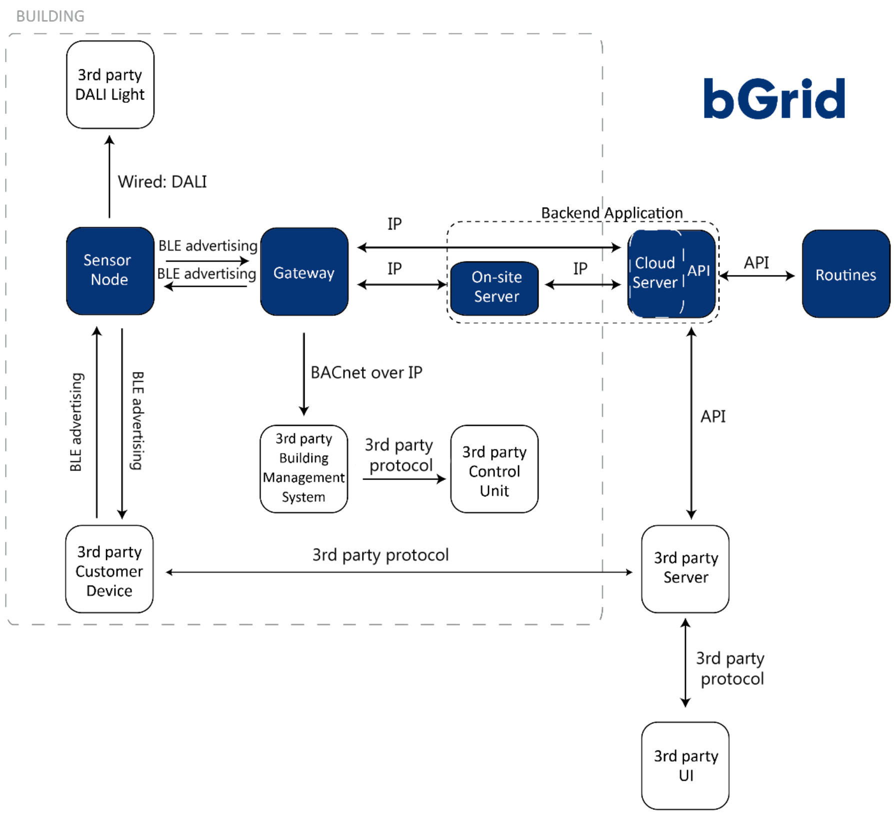


Figure 2.1: Deployment diagram of the bGrid system [7]

could affect user comfort.

An asset that creates a big challenge is the amount of data that the nodes are able to collect. A complete data set over a period of 24 hours can amount to 1GB of rough data. To give any algorithm a fighting chance of recognising patterns there has to be some periodicity in the data. For an office building most activity repeats daily but larger patterns emerge on a weekly basis (weekly meetings, weekly schedules etc.), meaning a full comprehensive data set could add up to 14GB. This size can be reduced when data that is not relevant for the purpose of this project is omitted and only the period between for instance 9:00 and 17:00 is looked at, but the amount of data may still pose a challenge. On the other side, acquiring sufficient ground truth data (on site observations) may be challenging. Not having access to camera footage means all observations will have to be done in person on location, making it a very time consuming and error prone activity. Another potential challenge comes with determining if a specific data point is valid or caused by noise. One can imagine that especially the sound and movement data is very susceptible to noise. The sound sensors not only pick up sound from the room where the node is located but also from neighbouring rooms or corridors. Movement sensors not only pick up movement from people who are currently occupying a room but also from people who just open the door and have a look inside. These things need to be taken into account as well.

3

Literature Study

This chapter summarises the literature review as follows: Section 3.1 will handle engineering solutions to the comfort management versus energy consumption minimisation problem. Section 3.2 will look into research concerning occupancy estimation and occupant behaviour modelling. For each piece of research there will be a brief summary; the main contributions are listed and the benefits and/or drawbacks in the context of replicating those results with the bGrid system are listed. Section 3.3 will compare the methods in terms of complexity and applicability. The chapter finishes with Section 3.4 where conclusions will be drawn.

3.1. Thermal Comfort Management

This project started out with the idea of designing an intelligent ambient temperature controller that could maximise user comfort while minimising energy consumption. The first thing that was researched is how to define thermal comfort, since this term is commonly used in all papers concerning comfort management. There are two distinct ways of looking at thermal comfort: the rational approach, where thermal comfort is assessed through heat balance calculations, and the psychological approach that assesses thermal comfort by measuring occupant satisfaction with the thermal environment through polls and inquires. Examples of the rational approach are the *ASHRAE 55* standard and the equivalent *EN 15251* standard for indoor thermal comfort [4, 12]. Both take into account a human's average metabolic rate for typical tasks, clothing insulation for typical ensembles, ambient air temperature, radiant temperature, air flow and humidity to illustrate environmental conditions that are acceptable to 80% or more of the users of a room. Besides the heat balance approach for thermal comfort, another method that both standards used for defining thermal comfort is the adaptive approach. One of the main contributions of this approach to thermal comfort is that it relates the indoor comfort temperature to the outdoor temperature. This method is fundamentally different from the heat balance approach as that relates the comfort temperature to environmental (as in indoor environment) factors and personal factors. A downside to this approach is that the correlation between comfort and outside temperature was intended for free-running or non air-conditioned buildings and is more complex and less stable for heated/cooled buildings [14, 15, 18, 23, 24, 25, 27, 44].

The trade-off between managing user comfort and minimising a building's energy consumption is a problem that has become more relevant in the recent years than it has ever been. Starting on the next page, this part covers papers centred around solving this issue.

Research team: Yang et al. [32, 47, 65]
Subject: Ambient temperature management using PSO (Particle Swarm Optimisation).
Contributions/conclusion:
The algorithm used PSO to maximise the overall user comfort level based on ambient temperature, illumination levels and CO_2 concentration. In simulation this method achieved maximum comfort while consuming less energy than conventional methods even though the energy consumption was not actively minimised. The comfort values were predefined by a group of users.
Benefits/drawbacks in context of available data:
The research done by Yang et al. provide a good foundation for further research. Their results showed that PSO is a valid solution for managing ambient temperature with a multi-agent sytem in a domestic environment. On the downside, this approach uses user defined values to determine the comfort levels. These are values that we do not have and would therefore be impractical to walk down this road.

Research team: Wang et al. [16, 17, 20, 60, 63, 64]
Subject: Ambient temperature management and power minimisation using MO-PSO (Multi-Objective Particle Swarm Optimisation).
Contributions/conclusion:
A continuation of the aforementioned research where the same research team extended the procedure to minimise the energy consumption while also maximising user comfort. To achieve this the PSO algorithm evolved into the MO-PSO algorithm which actively minimises the predefined user comfort and energy consumption cost functions. Additionally the research team was able to run this algorithm inside the central controller agents as opposed to previous research where the algorithm had to run on separate hardware.
Benefits/drawbacks in context of available data:
The solutions provided in the research done by Rui Yang and Linfeng Wang et al. relied on using the MOPSO algorithm to solve the conflict between maximising user comfort on the one hand and minimising energy consumption on the other. In the context of this project this approach has the benefit of being able to optimise two objectives at the same time and is thus guaranteed to save energy if implemented successfully. The downside is that it requires a level of control that the bGrid system is not always capable of. The system designed by Yang/Wang was intended for domestic use while the bGrid system operates in a corporate setting. In a domestic setting a system can be installed such that it has direct control over all climate influencing appliances. For office buildings and large buildings in general there is a Building Management System (BMS) with which any third party system needs to communicate, prohibiting or hindering the level of control any additional system would have. This is a barrier that would make a solution as proposed by Yang and Wang et al. less viable because it relies not only on having that level of control, but also access to all data necessary to make accurate estimations of energy consumption and personal comfort levels.

Research team: Ghahramani et al. [1, 2, 28, 29, 30]
Subject: A knowledge based approach for selecting energy-aware and comfort-driven HVAC temperature set points.
Contributions/conclusion:
Ali Ghahramani et al. developed a method in which thermal comfort preferences were learned online and then modelled as zone level personalised comfort profiles. The zone temperature set points were then selected through solving an optimisation problem for energy consumption with comfort, indoor air quality, and system performance constraints taken into consideration.
Benefits/drawbacks in context of available data:
The research conducted here showed similarities with that conducted by Yang et al. in that both methods required a multi agent set-up with control at a zone level. This 3-part approach relies heavily on user provided data, a functionality that the bGrid system does not yet utilise. Another thing Ghahramani et al. introduced was an estimation for energy consumption. The energy consumption for a specific zone was shown to be proportional to the airflow into that zone. However, this method relies on a specific type of HVAC system where all heating and cooling is exclusively handled by manipulating volumes of air (Variable Air Volume Air Handling Units). This is rare in most office buildings in the Netherlands and therefore not easily reproducible. The idea of scaling down the complexity is still a valuable contribution

and may be something to fall back on should the complexity exceed manageable limits.

Research team: Mozer et al. [13, 41, 42, 48]
Subject: Neural network aided optimal control
Contributions/conclusion:
Mozer et al. developed an adaptive controller that regulates indoor air temperature in a residence by switching the furnace on or off based on the results of an optimal control problem. The task considers both comfort and energy costs as part of the control objective. Because the consequences of control decisions are delayed in time, the algorithm had to anticipate heating demands with predictive models of occupancy patterns and the thermal response of the house and furnace. Occupancy pattern prediction was achieved by a hybrid neural network and look-up table combination.
Benefits/drawbacks in context of available data:
The study performed by Mozer et al. showed that no life is too irregular to be predicted, at least partly. Even highly non-deterministic schedules can serve as a basis for a predictive controller. A downside to the approach of Mozer et al. is that, just as the work done by Ghahramani et al. it assumes individual users to be able to override control decisions. In a domestic setting this approach will yield valuable but, more importantly, manageable data due to the small number of occupants. In a corporate setting the amount of occupants will be considerably higher making it unfeasible to use the same approach. Additionally, users of buildings equipped with the bGrid system don't have the degree of control that the occupants in the test setup of Mozer et al. had. The idea of using neural networks for pattern recognition in otherwise highly stochastic behaviour is nevertheless worth exploring, since occupancy in a corporate setting is not necessarily linked to specific individuals.

3.2. Occupancy Estimation

What became evident during the literature review concerning methods for thermal comfort management is the importance of human behaviour and accurate occupancy figures. All the methods summarised previously relied on knowledge of the occupancy in one way or another but none of the methods looked into accurately estimating the occupancy from sensor data. Dedicated research on this topic has been scarce and more focused on predicting occupant behaviour rather than determining the exact occupancy. A 2020 study by Salvatore Carlucci et al. reviewed approaches, methods and key findings in studies related to modelling occupants' presence and actions (OPA) in buildings [49]. They identified a total of 753 papers relating to the subject, of which 478 had to be disregarded due to them not being in English or the full text not being available. Of the remaining 278 only 53 were specifically about presence and activity, with the rest focusing on occupant activity such as window, shading and lighting operation. Of the remaining 53, only 4 papers researched determining a count for room-level occupancy.

Research team: Kjærgaard et al. [52]
Subject: Development of a fusion algorithm to determine the occupancy on a room-level using 3D camera footage
Contributions/conclusion:
The 2016 study by Kjærgaard et al. detailed the development of a fusion algorithm named *PLCount* for estimating occupancy using 3D camera footage. The algorithm builds on an existing counting methodology that uses cameras or thermal sensors to detect passing of so called count lines, a metric for occupancy. However, this method has its flaws and the error in occupancy adds up over time. The method detailed in this paper was able to minimise that error by op to 86% by using a dynamic programming approach to solve the count correction problem.
Benefits/drawbacks in context of available data:
The method described in this research is promising but the use of cameras makes it impossible to replicate. The issue of privacy was briefly mentioned in the study and was the reason the research team was not allowed to use footage from consecutive days for validation. In the Netherlands, if a single person objects to being filmed it is not allowed to collect any footage. Attempting a method similar to this research would require a controlled testing facility of some kind, which wasn't in the cards.

Research team: Kjærgaard et al. [33]
Subject: Designing a probabilistic algorithm to determine the occupancy on a room-level using CO_2 and movement data
Contributions/conclusion:
Continuing with a 2018 study, Kjærgaard et al. developed a method they called *DCount* that was able to achieve room-level counts with a documented low normalised RMSE of 0.93. They achieved this using a probabilistic algorithm that would assign a count to a room based on the measured CO_2 data and movement detection through PIR sensors. To evaluate the algorithm they obtained sensor data and building information data from a 8,000 m^2 office building, containing offices, classrooms and study areas with an average total daily occupancy of 1000 people. Multiple PC2 3D stereo-vision cameras were used to monitor transitions through the entrances and exits of the building. Analysing the video footage with their *PLCount* algorithm and collecting sensor data over a period of 30 days from September to October 2016 yielded a reference people count of 345,600 people, 3,499,200 PIR readings and 3,844,000 CO_2 measurements (both on a one minute basis). With this approach the research team was able to achieve considerably lower RMSE and reduce the estimation error up to 86% when compared to the raw people count.
Benefits/drawbacks in context of available data:
The method described in this research shows many similarities with what this thesis is trying to achieve. It was shown that CO_2 and movement data can paint a clear enough picture to accurately estimate the occupancy, given a large data set. The lack of video cameras will make it unfeasible to gather a similar amount of data but the bGrid system does collect more than just CO_2 and movement data which may compensate for the fewer number of samples.

Research team: Mora et al. [40]
Subject: Obtaining occupancy patterns via cluster analysis and logical flowcharts.
Contributions/conclusion:
In 2018 Mora et al. designed an experimental set-up in an office aimed to establish patterns in occupancy by monitoring occupancy state, relative humidity, CO_2 concentration, VOC (Volatile Organic Compounds), temperature, door and window opening and electricity usage. The correlation between all variables was looked into and both temperature and humidity did not show any clear correlation with the observed occupancy. Using cluster analysis and models based on logical flowcharts an error of 12% was achieved using one parameter. Using two parameters the error decreased to 10% but using more than three parameters didn't significantly improve the accuracy.
Benefits/drawbacks in context of available data:
The main takeaway in the context of this thesis is the work Mora et al. did to find the correlation between different data types. This may be used to a priori disregard temperature and humidity data because these did not show significant correlation with the occupancy. Instead, the focus can be on movement and CO_2 concentration which did show significant correlation with the occupancy.

Research team: Causone et al. [11]
Subject: A data driven approach to model occupancy in residential buildings using smart meters
Contributions/conclusion:
In 2019 a team lead by Francesco Causone developed a novel data-driven method that generates yearly occupancy and occupant-related electric load profiles. These were used to improve building energy modelling in terms of reliability and peak performance. The method starts by identifying occupant-related electric load profiles by reading out smart meters in the testing facility, a residential estate in Milan. It was impossible to identify the nature of the electrical appliances with the available smart meters and therefore the measured energy consumption covered all the appliances installed in each apartment. The resulting electric load profiles could be especially valuable to modellers who don't have access to ground truth data (direct observations).
Benefits/drawbacks in context of available data:
The research performed by Causone et al. does not contribute greatly in the context of this thesis. It is relevant because the results showed that knowledge of the use of electrical appliances could be used as a metric for determining occupancy at a room level, but to determine the exact number of people would require more sophisticated meters than what they had access to. Had the testing facility been equipped with smart wall outlets for example, the research may have looked different. In the context

of this thesis, no such data is available and can therefore not be used.

3.3. Comparison

Having discussed multiple papers concerning both comfort management and occupancy estimation, this section compares the individual research in terms of merits and drawbacks. The following tables rate and compare the previously discussed papers on the complexity of the methods used and the applicability in the context of this thesis or possible future work based on this thesis, since comfort management is beyond the scope of what this thesis hopes to achieve. The individual papers are specified by leading author and ranked on a five tier scale, indicated with '++','+', '0', '-' and '- -'.

	Yang et al.	**Wang et al.**	**Ghahramani et al.**	**Mozer et al.**
Complexity	-	- -	0	+
Applicability	- -	+	- -	0

Table 3.1: Comfort management method comparison

In Table 3.1, a '+' in complexity indicates that the specific method is in fact not excessively complex. Looking at the table, only the research by Yang et al. and Wang et al. were deemed to be overly complex compared to the others [60, 63, 64]. The reason for this is that these methods both ran an online optimisation method to manage user comfort, requiring more computing power than the (partly) offline or otherwise less involved approaches in the other papers. The research done by Ghahramani et al. for instance utilised a database containing user defined preferences and a scalar optimisation problem, requiring less computing power than the multi-objective optimisation problem as proposed by Yang et al. Similarly the research by Mozer et al. captured predictable patterns in occupant behaviour in a database while using an online reinforcement learning approach for the stochastic parts.

The applicability of the papers on comfort management is rated taking future work into account, on the premise that this thesis achieves what it aims to achieve. The most applicable research seems to be that of Wang at al. which is the only one that actively minimises the energy consumption. In this aspect having knowledge of the exact occupancy could be a great help. A common theme among all papers is the requirement of input that the bGrid system cannot deliver to manage thermal comfort. Features such as access to individual comfort preferences of the occupants or occupants having a degree of control that is very difficult to reproduce in a corporate setting and the bGrid system is therefore not (yet) equipped with. This could however be bypassed by making some assumptions based on for example the *ASHRAE 55* standard for indoor thermal comfort. Using that, maximising thermal comfort becomes a heat balance problem where knowledge of the number of occupants in a room at any given time is essential because of the contribution people have to the thermal environment [12, 19]. Looking past this fact however, there are parts of the research by Mozer et al. in particular that could be applied to the research in this thesis. Using a neural network for pattern recognition the way Mozer et al. did show similarities to what this thesis is trying to achieve and will therefore be looked into further in chapter 4.

Table 3.2 shows a similar comparison as before, using the occupancy estimation research. Again the papers have been scored on a five tier scale in terms of complexity and applicability in the context of this thesis. The only paper that was deemed overly complex is the paper by Mora et al. Where this thesis focuses on relating measured data to observed occupancy the research by Mora et al. tried to find a correlation between all measured data and observations through cluster analysis and logical flowcharts, which is beyond the scope of this thesis. Comparing that to the work done by Kjærgaard et al. in 2016 where they used 3D cameras as a tool for occupancy estimation, the latter is rather less involved. They continued their work in 2018 where they used the method developed in the first paper to gather ground truth data while designing a method based on measured data to estimate occupancy, increasing the complexity. This approach does make the 2018 research by Kjærgaard et al. very applicable, which will be discussed in the next paragraph.

	Kjærgaard et al. (2016)	**Kjærgaard et al. (2018)**	**Mora et al.**	**Causone et al.**
Complexity	++	+	-	0
Applicability	- -	++	- -	- -

Table 3.2: Occupancy estimation method comparison

Since these papers are centred around the same subject as this thesis, the work should be more applicable. However most methods that were discussed still use data that the bGrid system cannot deliver. The 2016 study by Kjærgaard et al. for example used data captured by 3D cameras as a basis for occupancy estimation while Caosone et al. used insight in the use of the electrical systems to achieve similar results. The 2018 study by Kjærgaard et al. however is highly applicable to this thesis, as mentioned before. Cameras were still used to collect ground truth data but this is not the only way in which ground truth data can be collected and can therefore be circumvented. The rest of the research showed that both movement data and CO_2 data can serve as a good foundation for accurate occupancy estimation. Since the bGrid data is able of providing exactly those data types, this is a valuable realisation and a good place to start from.

3.4. Conclusion

Having compared all methods in terms of complexity and applicability, some conclusions can be drawn. The focus will be on the methods for occupancy estimation as that is what is most in line with the direction of this thesis. Based on the research done by Mora et al. the data types that will be used a priori will be CO_2 and movement, since these showed significant correlation with the occupancy. This is supported by the work done by Kjærgaard et al. which also showed CO_2 and movement data resulted in accurate occupancy estimates. Of course the bGrid sytem does provide additional data types, one of which is the sound intensity. This data type in particular is something that sets the bGrid system apart from others and will be looked into. Additionally the work by Mozer et al. showed that even highly stochastic behaviour could be modelled by a artificial neural networks. Because of this and again supported by the work by Kjærgaard et al. the next chapter will dive deeper into neural networks while also looking at a linear modelling method.

4

Methods

Based on what has been achieved in previous research, this thesis aims to achieve similar or better results in terms of occupancy estimation. The one thing that separates previous work from what this research is trying to attempt is the amount of ground truth data available. The office that was used was not equipped with cameras and thus ground truth data had to be gathered through direct observation. This is a time consuming method of gathering data and the ground truth data set is therefore quite limited. However, if a strong relation between measured data and the observed occupancy does exist, even a relatively small data set could probe sufficient. The first method that will be looked into is a linear regression method due to its widespread usage in data based modelling. Secondly an artificial neural network approach will be discussed, partly based on the work in [33].

4.1. Linear Regression

As mentioned linear regression methods are widely used in data based modelling. The most standard variant is simple linear regression, using a single scalar input variable *x* and a single scalar output variable *y*. In most real-world problems however, the inputs and outputs are not scalar and thus a vector based method is required. Two such methods will be discussed below.

Multiple Linear Regression [62]

MLR is an elementary data driven modelling technique that uses a vector of regression coefficients $\vec{b}$ and a vector of residual errors $\vec{e}$ in the following relation:

$$\vec{y} = \mathbf{X}\,\vec{b} + \vec{e} \tag{4.1}$$

Where $\vec{y}$ is the output vector containing the *m* individual outputs y_i and $\mathbf{X}$ is the *m x n* data matrix containing the individual inputs $x_{i,j}$. Equation 4.1 has a unique solution for $\vec{b}$ if and only if $m = n$ and the data matrix is of full rank. This is rare however and thus an exact solution for $\vec{b}$ can often not be determined. Still, it is possible to reach a solution by minimising the residual error $\vec{e}$ using for example the least squares method. The objective function then becomes:

$$L = \sum_{i=1}^{m} e_i^2 = (\vec{y} - \mathbf{X}\,\vec{b})^T(\vec{y} - \mathbf{X}\,\vec{b}) \tag{4.2}$$

Resulting in a solution for $\vec{b}$ of the form:

$$\vec{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}\vec{y} \tag{4.3}$$

This method is well understood and widely used due to its simplicity and applicability. The only thing left to do is to select the inputs that this model will include. Usually, a trial and error based method is used to see which inputs achieve the best performance.

Principal Component Regression [38, 50, 67]

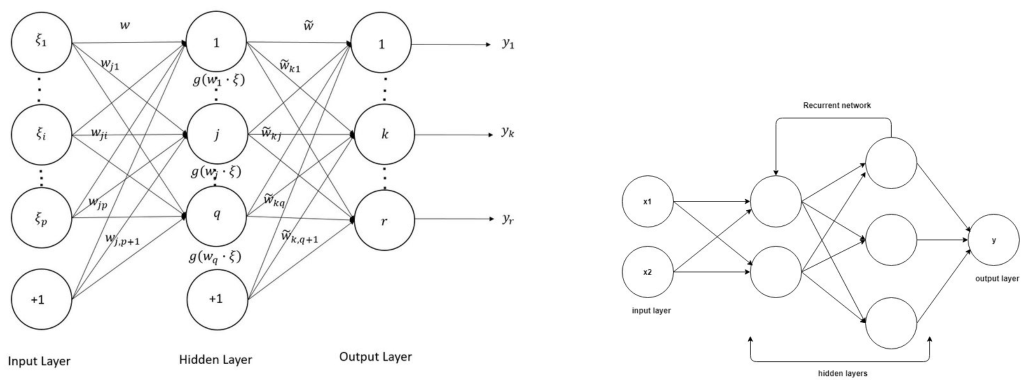
A problem with the MLR technique is possible collinearity between the supposedly independent variables. PCR is a variation of MLR that uses principal component analysis to construct an orthogonal basis from $\mathbf{X}$. From this basis a subset is selected which is then used to predict y. Determining the number of principal components to base a prediction on can be difficult. Using too many can result in extra noise while using too few could result in an incomplete weight description of $\mathbf{X}$. There are a multitude of methods to find the optimal number of principal components, of which the *cross validation* [61] and *average eigenvalue* [31] methods are some of the most well known. First, the matrix $\mathbf{X}$ is decomposed in a score matrix $\mathbf{T}$ and a loading matrix $\mathbf{P}$, consisting of the right singular values of $\mathbf{X}$ in the following way:

$$\mathbf{X} = \mathbf{T}\,\mathbf{P}^T + \tilde{\mathbf{X}} = \mathbf{T}\,\mathbf{P}^T + \tilde{\mathbf{T}}\tilde{\mathbf{P}}^T = [\mathbf{T} \quad \tilde{\mathbf{T}}][\mathbf{P} \quad \tilde{\mathbf{P}}]^T \tag{4.4}$$

Where the matrix $\tilde{\mathbf{X}} = \tilde{\mathbf{T}}\tilde{\mathbf{P}}^T$ contains the residual errors. The loading matrix $\mathbf{P}$ describes the projections of a unit vector along the principal components while the score matrix $\mathbf{T}$ contains the coordinates of the data points on the matching principal component line. After the transformation, MLR can be utilised to solve the original problem with a higher chance of success due to the eliminated collinearity and overall reduction of the problem size.

4.2. Artificial Neural Networks

It has long been known that ANN can be used to solve problems in system identification. [10, 54] Largely speaking, ANN can be classified in two categories: feedforward neural networks (*FFANN*) as shown in Figure 4.1a, and recurrent neural networks (*RANN*) as shown in Figure 4.1b.



(a) Example of feedforward neural network

(b) Example of recurrent neural network

Figure 4.1: Neural network types

FFANN are mostly used in pattern recognition problems, similar to the problem presented in this thesis. RANN are more powerful and can grow into vast complex systems. Outputs from neurons can be fed back into the network, creating a system with something that resembles a memory of past events. Unless stated otherwise the next part will concern FFANN.

In the case where an ANN is used to find a relation between past input-output data gathered in the vector $\phi(t)\ \epsilon\mathbb{R}^d$ and future output $y(t)$, the general function expansion looks like this:

$$\hat{y}(t|\theta) = g(\phi(t), \theta) = \sum_{k=1}^{n} \theta(k) g_k(\phi(t)) \tag{4.5}$$

Where $g_k(\phi(t))$ is a basis for a general function that maps $\mathbb{R}^d \rightarrow \mathbb{R}$, $\hat{y}(t|\theta)$ is used instead of $y(t)$ to indicate that $g(\phi(t), \theta)$ is an estimate for $y(t)$ given $\phi(t)$ and given a particular parameter value θ. The "optimal" value for θ can be obtained by solving the optimisation problem stated below:

$$\hat{\theta}_N = argmin \sum_{k=s}^{N} \left| y(t) - \hat{y}(t|\theta) \right|^2 \tag{4.6}$$

Where, once more, $\hat{\theta}$ is used to indicate that this is the best estimation and not the actual value. To go from the general function expansion in Equation 4.5 to an ANN with one hidden layer and one output neuron, some assumptions need to be made. If $g_k(\phi) = \alpha_k \sigma(\beta_k \phi \ + \ \gamma_k)$ is chosen as a basis, where β_k is a parameter vector with dimension matching ϕ and both γ_k and α_k are scalars, the following is obtained:

$$g(\phi) = \sum_{k=1}^{n} \alpha_k \sigma(\beta_k \phi \ + \ \gamma_k) + \alpha_0 \tag{4.7}$$

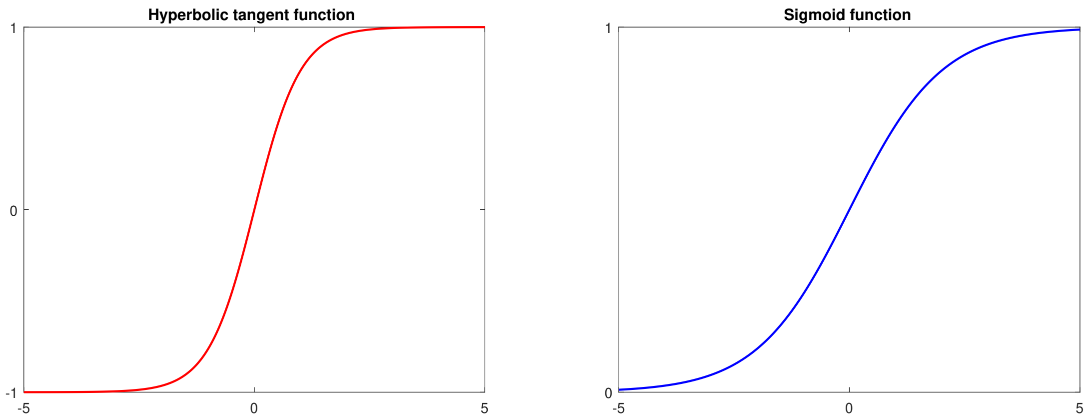
Where α_0 is an arbitrary parameter to adjust the mean [54]. $\sigma(\cdot)$ is the activation function and is usually the same for all neurons. The activation function determines what input values activate the specific neuron. A common choice for an activation function is the sigmoid or logistic function, which squeezes all real numbers between the values 0 and 1, of the form:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.8}$$

This specific sigmoid function is popular because it allows the use of gradient based parameter estimation methods such as back propagation and gradient descent. An alternative to the sigmoid function that offers the same functionality is the hyperbolic tangent function of the form:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.9}$$

Figure 4.2 shows both activation functions side by side.



(a) Hyperbolic tangent function

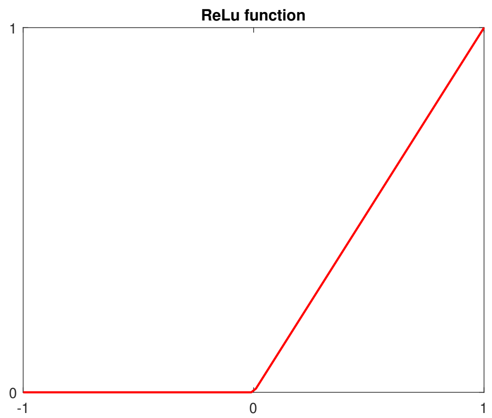
(b) Sigmoid or logistic function

Figure 4.2: Two sigmoidal activation functions

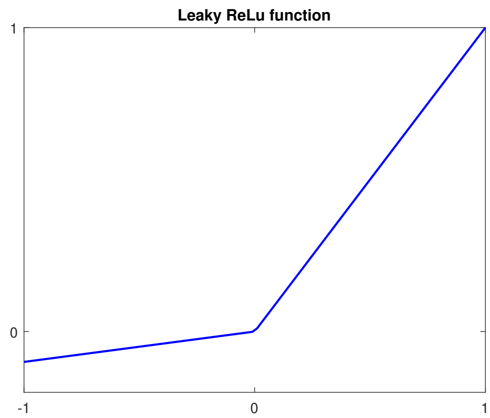
Contrary to the logistic function, the hyperbolic tangent function squeezes its inputs between -1 and +1. What sets the hyperbolic tangent function apart from the sigmoidal logistic function is that negative inputs will be mapped as strongly negative while zero inputs will remain at zero. A downside to sigmoidal functions like the logistic function and the hyperbolic tangent function is that, using back propagation for example, these functions can be saturated. The inputs are squeezed between the range limited by 0 and 1 in the case of the logistic function (-1 and +1 for the hyperbolic tangent) which means if the error of the neural network is stuck at a sufficiently high constant value during learning, the performance will no longer improve [66]. This is known to be caused by an inappropriate choice in initial weights which is why these will be randomised. In that way the risk of saturation due to inappropriate initial weights is minimised [45].

An alternative to sigmoidal activation functions are non-saturating activation functions. These functions don't squeeze the input and are consequentially not susceptible to saturation. An example are the Rectified Linear Units (ReLU) and Leaky Rectified Linear Units (Leaky ReLU) functions shown in Figure 4.3 and described by the following equations:

$$ReLU(x) = \begin{cases} 0 & for \quad x \leqslant 0 \\ x & for \quad x > 0 \end{cases} \qquad LeakyReLU(x) = \begin{cases} ax & for \quad \leqslant 0 \\ x & for \quad x > 0 \end{cases} \tag{4.10}$$



(a) ReLU function



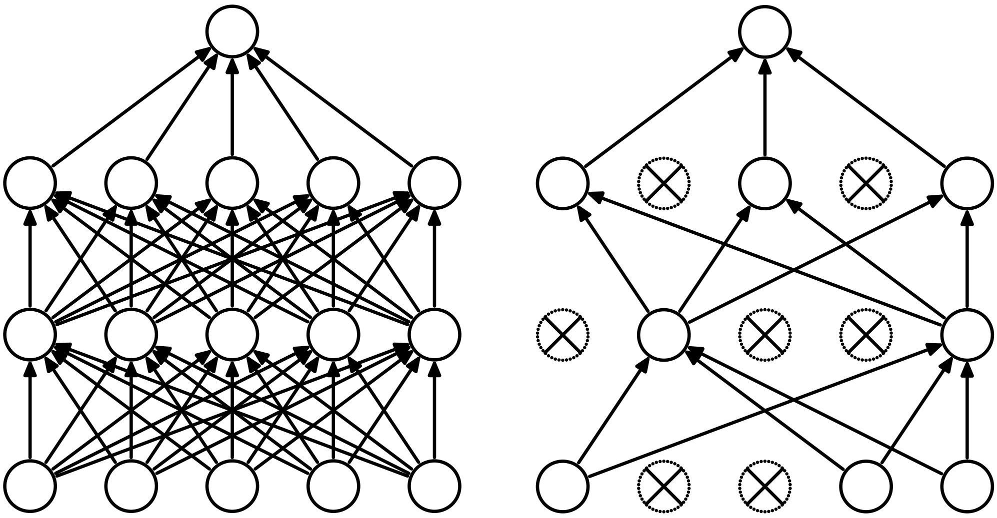
(b) Leaky ReLU function

Figure 4.3: Two non-saturating activation functions

Using a four-layer convolutional neural network with ReLU shaped activation functions on *CIFAR-10* (a collection of images commonly used to train machine learning algorithms) it was shown that a training error of 25% could be obtained six times faster than a similar network that used the hyperbolic tangent activation function [34, 43]. This speed boost is especially useful when using big data sets for training and validation and may therefore not achieve significantly better performance in this research. However, since combining multiple activation functions in one network is not uncommon, using a combination of sigmoidal and non-saturating activation functions is worth looking into. Research by MD Asaduzzaman showed using a combination of activation functions can increase the training speed and performance[3].

The choice of activation function influences how quickly a network converges to the desired error but it is not the only influence. The learning rate also influences convergence time and general performance. The learning rate is a variable that determines how heavily the error is fed back to the neurons. A high learning rate means the error is weighed heavily and the network reacts strongly to a slight error, making it unlikely the algorithm will converge to a local minimum. However it also means that the training algorithm is prone to overcorrect itself, making convergence difficult to achieve. On the other hand, a low learning rate means the error is not weighed as heavily and the network reacts to the error more gently. This makes it more likely the training algorithm will converge, although it getting stuck in a local minimum is a valid concern. Generally a learning rate smaller than 10e-2 but greater than 10e-6 is considered safe [55].

The final hyper-parameters that will be discussed are the number of hidden layers and the number of neurons per layer. These parameters determine the complexity of the network and influence the risk of under- or overfitting. Underfitting is where a network has not learned enough from the training data and does not produce a tight fit. Overfitting is the opposite, where a network has learned too much from the training data and is therefore not generalisable. These attributes can also be caused by a lack of variation in the training data but this is generally not something that can be easily changed [53, 55]. Avoiding both under- and overfitting is vital to the performance of a network and can be achieved by optimising the number of layers and neurons. A common optimisation method used for this purpose is the genetic algorithm as it allows for integer optimisation [55]. Another method specifically designed to prevent overfitting is called *dropout*. This method randomly drops a predefined number of neurons and their connections, characterised by the variable p. It can be applied to a single hidden layer or all input and hidden layers. The variable p has a value between zero and one and determines the fraction of neurons that are dropped (i.e. p=0.5 means half of the neurons and their connections are randomly selected and dropped).



(a) Standard Neural Network

(b) Neural Network with *dropout*

Figure 4.4: A standard neural network with two hidden layers (a) and a network produced by applying dropout to the network on the left (b). Crossed neurons have been dropped [22].

Since the method randomises which neurons get dropped, the presence of any specific neuron becomes unreliable. This breaks up any co-adaptations that are inherent to fully connected networks trained through back propagation [22]. Figure 4.4 shows an example of a simple neural network using *dropout*.

4.3. Comparison

Similar to the previous chapter, the discussed methods will be rated in terms of complexity and applicability. In this case all methods are applicable, otherwise they would not have made the cut. The score they receive for applicability will consequently be based more on how suited the method is given the relatively small data size and nature of the data set. Once again a five tiered rating system will be used, consisting of '++', '+', '0', '-' and '- -'.

	MLR	**PCR**	**FFANN**	**RANN**
Complexity	++	+	+	-
Applicability	0	-	0	-

Table 4.1: Modelling methods comparison

Table 4.1 compares the methods discussed previously. Looking at the linear methods first, MLR stands out as the least involved. PCR adds a layer of complexity because of the necessary transformation and therefore scores lower in that area. Looking at the applicability, both methods would be better suited for larger data sets. This is a bigger issue for PCR which reduces the data size even further. The relationship between the measured data and the occupancy is also expected to be non-linear, making it difficult for either linear method to achieve decent performance. The non-linearity could possibly be circumvented by sorting the measurements based on the observed number of people, more on this in subsection 5.2.1.

Looking next at the non-linear methods, the RANN can be disregarded immediately. The feedback element inherent to RANN adds unnecessary complexity to an already complex problem. The added complexity makes this type of network especially suited for speech recognition, a field in which these networks are widely used [51]. For the non-linear methods that were discussed, the one remaining is the FFANN. Looking at the complexity it was rated equal to PCR, even though non-linear methods are more involved by nature. The reason for this is that the basic design and architecture need not be very complex and can be gradually expanded and improved as desired. Looking at the applicability, the score was neutral. The reason for this is that a FFANN is perfectly suited to solve problems similar to what this thesis offers but does struggle to perform well when the training data set is relatively small.

4.4. Conclusion

This chapter outlined two linear methods and one non-linear method that can be used to model the occupancy from the measured data. Comparing the linear methods showed MLR was the best candidate. The added complexity and data size reduction that come with PCR contribute to the lower score in Table 4.1 and will therefore not be used in the experimental analysis. MLR is the first method that will be used, starting in subsection 5.2.1.

Secondly artificial neural networks were researched, based on the work by Kjærgaard et al.[33, 52] and Mozer et al.[41, 42] discussed in chapter 3. As discussed, the problem can be seen as a pattern recognition problem and so a feedforward neural network would be an appropriate choice to start with. The initial architecture should not be overly complex to avoid long computation times and/or overfitting. A good starting point would be a network with three hidden layers and an equal number of hidden neurons per layer, all using the same activation function. The hyperbolic tangent function is perfectly suitable in combination with back propagation as this function allows the network to respond equally well to positive and negative errors. Issues with activation function saturation can be dealt with if and when they present themselves, but a priori no counter methods have to be applied. Later on the dropout method can be applied if overfitting becomes an issue. The experimental analysis will start with structuring the data properly, followed by an approach centred around MLR and will finish with a neural network centred approach. The results for each approach will be presented as the methods progress.

5 Experimental Analysis

Based on the previous work and literature, this chapter will outline the approach to generate the occupancy from the measured data using both the MATLAB and Simulink programming environments by the MathWorks inc. The approach will consist of two parts, starting with observation and data management. The second part will consist of modelling using two distinct methods that were outlined in chapter 4 and the validation of these models.

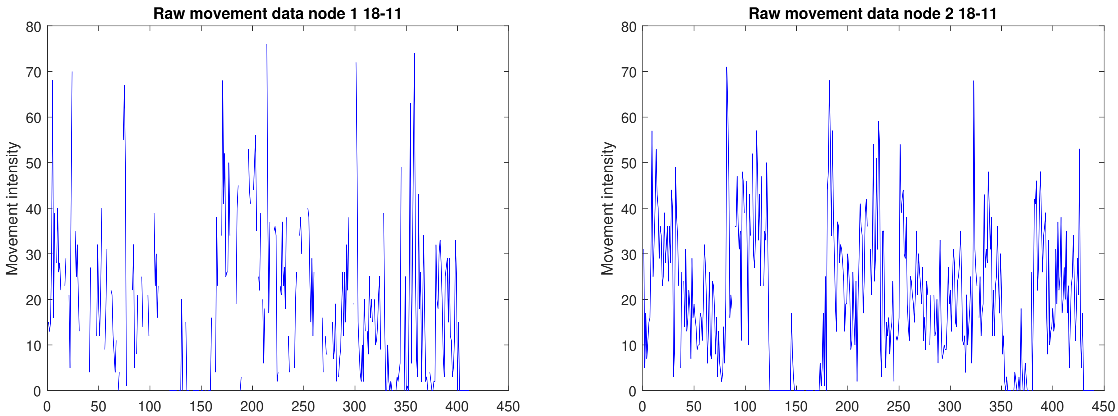
5.1. Observation and Data Management

The first step was to perform occupancy observations in a building equipped with the bGrid system. The Microsoft Office at Schiphol in Amsterdam was chosen for this due to the abundance of meeting rooms with glass walls so that the observations could be performed without interfering with office work or affect the data. An added benefit was easy accessibility since the office was located at Schiphol airport, which is easily accessed both by car and by train (and by airplane, obviously). Two similar meeting rooms of roughly 16m^2 were chosen with a rated maximum capacity of eight people, similar to the office shown in Figure 5.1.
The observations were performed over two days on 18 November and 20 November of 2019 on a one minute basis, keeping track of the occupancy and activity in the two offices. This resulted in a total of 1396 minutes of observation to use as a target, spread over 3 sets of data: Room 1 on day 1, room 1 on day 2 and room 2 on day 2.

Using the bGrid sensor nodes, data was collected over the two 2 days of observation. The bGrid nodes transfer the measurements wirelessly to a server in one-minute batches. Because of the wireless transfer, solid signal strength is essential to limit packet drops. This is something that can not always be ensured, resulting in gaps in the data that can cause problems when algorithms count on a steady data flow as input. As an example,Figure 5.2 shows actual movement data from day 1 taken from the two nodes situated in room 1 at the Microsoft Office. The most important difference between Figure 5.4a and Figure 5.4b is the number of missing data samples. The data is gathered in the time between 10:09 and 17:30 and thus should consist of 442 samples. In comparison, the data from node 2 contains 422 samples (95%) whereas the data from node 1 contains 304 samples (69%). Besides the obvious packet drops there is also a less obvious consequence of bad signal strength. Under normal conditions the nodes send their data every 60 seconds. In the case of node 1 specifically, this time is increased to roughly 64 seconds. The reason for this is that the algorithm responsible for the data transfer has a buffer that continues trying to send the data for a fixed time. Only when it fails to send the data within

Figure 5.1: Meeting room at Microsoft Office similar to the ones used for observations ©Microsoft

the set time is the data dropped. This results in less samples in a given window of time, even if no samples are dropped.



(a) Node 1

(b) Node 2

Figure 5.2: Preprocessed movement data taken on 18-11-2019 in room 3309 at the Microsoft office

Note that using node 2 as a backup to node 1 when a piece of data is dropped (and vice versa) is not an option. Even though both nodes are monitoring the same room, the overlapping area is minimal. This would mean that for instance movement that is picked up by node 1 may not be picked up by node 2. Note that this issue is more relevant when looking at the movement intensity. Environmental such as relative humidity, CO_2 concentration and temperature are continuous by nature, meaning there will be a correlation between consecutive samples measured by the same sensor. In both cases it is best to rely on older samples from the same node, rather than using data from another node and possibly introducing singularities.

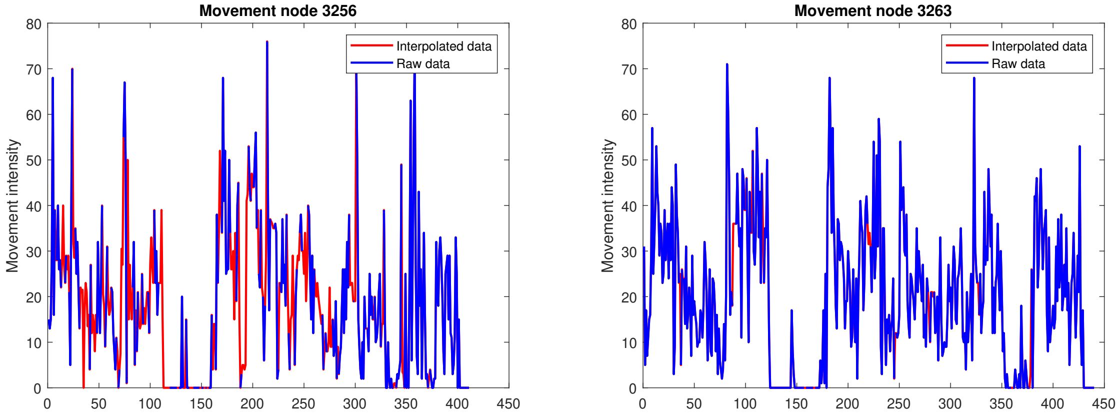
There are several methods to cope with the data loss, two of which will be discussed in this section. First, because of the nature of the project it may not be necessary to use a sample size of one minute.

Based on the observations, occupancy isn't particularly volatile and thus a sample size of five minutes may prove to be sufficient. The samples would be gathered in five minute blocks and averaged over the available samples. The main benefit is that, since the collected data never shows 5 consecutive sample drops, this method would ensure a data set with no gaps. The downside is that the number of samples is reduced by a factor of 1/5 which given that there are only two days of data is quite a steep price to pay. Secondly, the data can be resampled to the desired amount of samples using a variety of functions in MATLAB. Two functions were investigated, namely the *resample* function and the *fillmissing* functions. For the first one, the data was fed in unaltered, after which the algorithm uses the given sample rate to generate a new data set with the requested number of samples. The main benefit is the ease of use and the speed of the algorithm. The downside is that the algorithm does not know where samples are missing. The resulting data will have lost a lot of correlation with the original, as can be seen in Figure 5.3.

Figure 5.3: Movement data taken on 18-11-2019 in room 3309 at the Microsoft office, zoomed in

Again, the movement data from the same meeting room is used. Figure 5.3 show the resampled data in red versus the preprocessed raw data in blue, zoomed in at a piece of data that shows the shortcomings of the algorithm. Note that the data set with the most sample drops (node 1) was used. Figure 5.3 shows that the resampled data is clearly ahead of the original, something that occurs throughout the data set. The algorithm performs better when a more complete data set is used but given the nature of the collected data, this is not something that can be guaranteed. This method is therefore less than ideal.

The *fillmissing* function requires some preprocessing. The raw data doesn't show any gaps when it is in its raw form but by looking at the difference in the timestamps it is possible to determine if and where to introduce gaps in the data. Once the gaps are in place, the *fillmissing* function can go to work. Moving median was chosen over moving average as the interpolation method due to its inherent robustness and unaffectedness to outliers [26]. The window was chosen as 10 samples because this showed to produce less artefacts than taking the median of a smaller window. Filling the gaps in the data from node 1 and 2 used in the example increased the number of samples to 411 (93%) and 440 (99%) respectively as shown in Figure 5.4. The original data is shown in blue while the interpolated data is shown in red.



(a) Node 1

(b) Node 2

Figure 5.4: Results of the *fillmissing* function

5.2. Modelling

Now that the data is in a more manageable format, the modelling phase can begin. The goal is to derive a model that can transform the data into a certain number of people in real time. Throughout the modelling phase the data from room 1 on day 1 and 2 will be used for the design while the data from room 2 on day 2 will be used for validation.

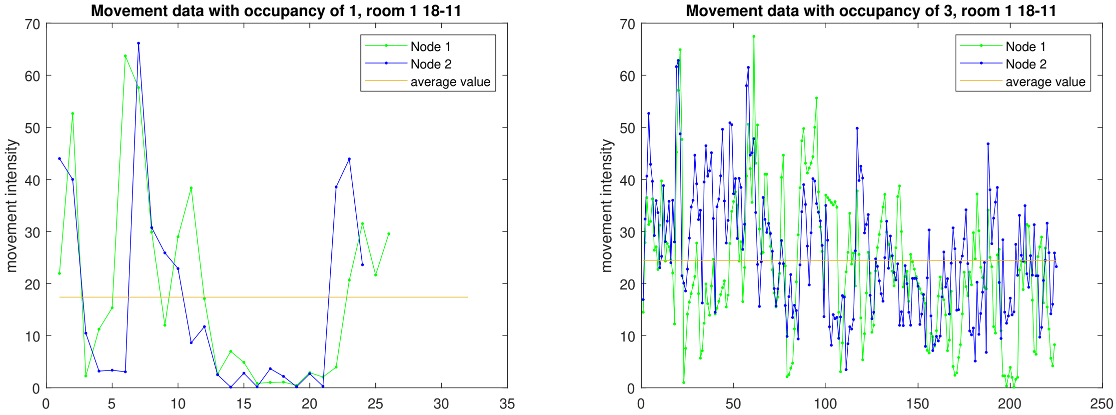
5.2.1. Linear Regression

Looking at the data, it is clear that linear methods used on the data as a whole will not yield positive results. However, using linear methods on certain parts of the data may prove quite potent. The linear modelling method is handled as follows:

1. Separate the data based on the observed occupancy.
2. Perform linear regression analysis on the separated data sets.
3. Create a switched system to switch between the different regression coefficients based on the input data.
4. Validate the results using the validation data.

To separate the data the observed occupancy is split into monotone parts, making sure the timestamps are preserved and unscrambled. Once the occupancy data is split, the measured data can be matched up and separated. This yields new sets of data that correspond to a certain constant occupancy. Figure 5.5 shows the results, displaying the movement intensity measured by both nodes in blue and green and the average movement intensity value in yellow. This last value is especially important because using that it is possible to say if a linear method has a chance of success. If the average value for a certain data type for a certain occupancy level is close to equal to the average value at a different occupancy level, the regression coefficients will be too close together to tell the difference between different levels of occupancy. Similarly, the average values should increase as the occupancy does. If this is not the case, using a linear method does not make sense. That being given, it would be interesting to look at the different average values for different occupancy levels. Table 5.1 shows exactly that. The CO_2 concentration and movement intensity data behave as expected, showing higher average values as the occupancy increases. The sound however appears counter intuitive, showing lower average values for higher occupancy. The average value was taken using both nodes with equal weight but the average values for individual nodes were also checked. This yielded similar results, indicating sound is an unreliable metric and as a result should be omitted for this research.

With the data separated, the linear regression method described in section 4.1 can be applied. Table 5.2 shows the regression coefficients for both the movement intensity data and the CO_2 concentration for all levels of occupancy. A few things stand out when looking at the values. While the



(a) Occupancy of 1 person

(b) Occupancy of 3 people

Figure 5.5: Separated movement data and average value

Occupancy	Movement(#)	$\mathbf{CO_2(PPM)}$	Sound (dB)
1	15.8711	730.1221	21.9042
2	21.5159	826.1981	23.7848
3	24.4610	851.1961	18.5432
4	25.7313	958.0088	14.1992
5	27.5050	1227.6129	27.2043

Table 5.1: Average values

previously shown average values for both data types increased as the occupancy increased, the same can not be said for the regression coefficients. In the case of the movement the regression coefficient for an occupancy of 4 and 5 people are smaller than the one for 3 people. This can be explained by the fact that the movement intensity at higher levels of occupancy can be significantly higher than at lower levels, meaning the regression coefficient needs to be smaller. Looking at the regression coefficient for the CO_2 concentration data, the ones for an occupancy of 4 and 5 are nearly identical. This does not mean that the CO_2 values are on average the same at those levels of occupancy, it simply means the step in CO_2 concentration from 4 people to 5 people is close to linear.

Occupancy	1	2	3	4	5
Movement	0.0546	0.1023	0.2446	0.1557	0.2217
$\mathbf{CO_2}$	0.0014	0.0024	0.0035	0.004	0.0041

Table 5.2: Linear regression coefficients

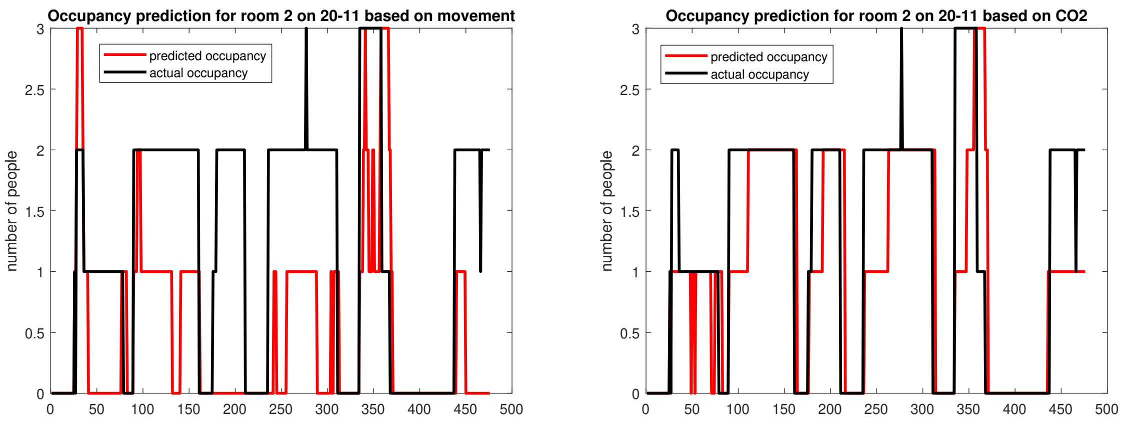
The regression coefficients shown in Table 5.2 can be used to estimate the occupancy based on the second day of data. As a starting point, the CO_2 concentration and movement intensity data will be used separately. Starting with the CO_2 concentration the first thing to fix, since the CO_2 concentration is slow to react to a change in occupancy, is that the occupancy has to be zero when no movement is detected. Movement intensity will always be the leading data type for detecting presence. The easiest way is to set the CO_2 concentration to zero when there is no movement, even though it is technically impossible for the CO_2 concentration in an office to be zero. The pseudocode in Equation 5.1 shows the process.

$$
\begin{aligned}
&for \quad i = 1 : finallength \\
&\quad if \quad movement(i) = 0 \\
&\quad\quad CO_2(i) = 0 \\
&\quad end \\
&end
\end{aligned}
\tag{5.1}
$$

This will ensure the CO_2 concentration doesn't lag when people leave the room. It should also ensure that occupancy is immediately measured as movement is detected. Next a method has to be developed to choose a specific regression coefficient based on the input data. This can be done using lower and upper bounds based on the average value of the training data shown in Table 5.1.

$$
\begin{aligned}
&for \quad i = 1 : finallength \\
&\quad if \quad data(i) = 0 \\
&\quad\quad occupancy(i) = 0 \\
&\quad elseif \quad data(i) > 0 \quad and \quad data(i) < upperbound_1 \\
&\quad\quad occupancy(i) = r_1 * data(i) \\
&\quad\quad . \\
&\quad\quad . \\
&\quad\quad . \\
&\quad elseif \quad data(i) > upperbound_4 \quad and \quad data(i) < upperbound_5 \\
&\quad\quad occupancy(i) = r_5 * data(i) \\
&\quad end \\
&end
\end{aligned}
\tag{5.2}
$$

The pseudocode shown in Equation 5.2 shows this process, where $data(i)$ can refer to either the movement intensity data or the CO_2 concentration data since the method is the same for both data types. The only difference between the two is the value of the lower and upper bounds. r_i are the regression coefficients from Table 5.2 and the values for $upperbound_i$ are based on the values in Table 5.1. Rounding the output to the closest integer yields the results shown in Figure 5.6, with Figure 5.6a using the movement intensity data and Figure 5.6b using the CO_2 concentration data.



(a) Predicted occupancy using movement intensity

(b) Predicted occupancy using CO_2 intensity

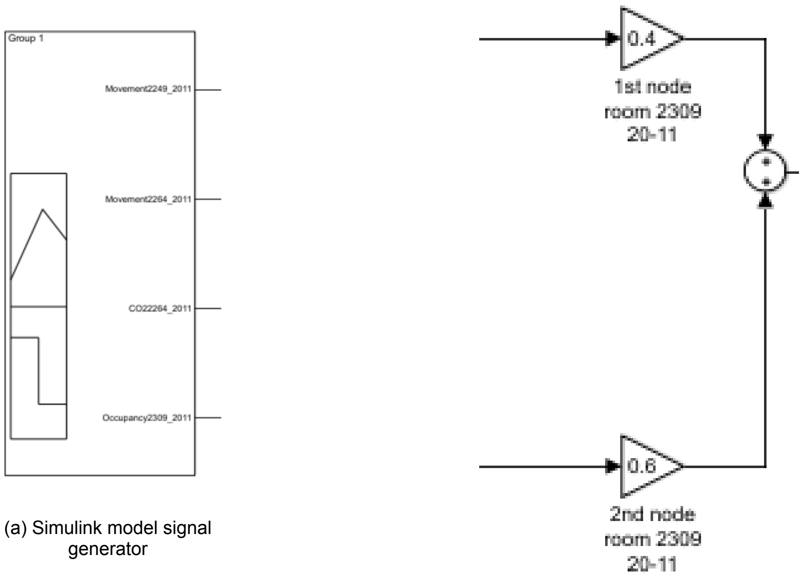
Figure 5.6: Results of linear regression method using movement intensity and CO_2 concentration separately

A measure for performance will be the number of times the system produces a correct result divided by the total number of iterations, multiplied by 100 to arrive at a percentage:

$$P = 100 * \frac{correctness}{i_{total}} \tag{5.3}$$

Using this, the performance of the method used in Figure 5.6a and Figure 5.6b was calculated to be 37.0% and 65.3% respectively, a result that does not seem too impressive. Looking at Figure 5.6a it is clear that the movement intensity data enables the algorithm to react quickly to changes at the cost of accuracy but looking at the CO_2 concentration based method in Figure 5.6b the opposite is true. A combination of both data types should hence yield better results.

Combining the CO_2 concentration and movement intensity seamlessly was easiest using a Simulink model. The full Simulink model can be found in Appendix A but the main parts are highlighted below. The first part, shown in Figure 5.7a is the signal generator that perfectly replicates the data that was loaded into MATLAB. It is therefore necessary to first run the data management and linear regression file to make sure the Simulink model has access to all the required parameters from the MATLAB workspace.

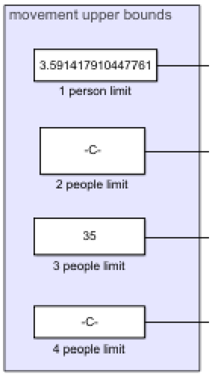


(a) Simulink model signal generator

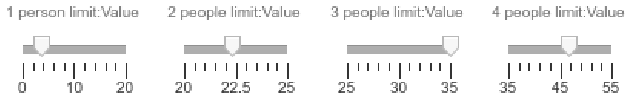
(b) Simulink model node weights

Figure 5.7: Zoomed in details for the Simulink model

In Simulink a few additions can be made to give additional control to the way the algorithm handles the data. The first addition was a modifier that changes the weight of an individual node, shown in Figure 5.7b. Movement intensity data is collected by both nodes in the room, each covering part of the office. As mentioned earlier, one is positioned near the door (node 1) while the other is located near the back wall (node 2). Since node 1 is picking up the movement at the entrance to the office, it also detects movement when somebody is only opening the door. Because of this, the data collected by node 1 is more susceptible to noise and is given a lower weight. Additionally, node 1 suffered more data loss than node 2. This was of course corrected but the fact remains that node 2 contains more original data than node 1, making it more reliable. Below, Figure 5.8a shows the block where the upper bounds for in this case the movement intensity data are set. These fixed values highly influence whether the algorithm performs well or not and hence should be adaptable. To serve that purpose, these bounds where connected to slider gains shown in Figure 5.8b to manually find which values yield the best performance.



(a) Simulink model movement intensity upper bounds



(b) Simulink model movement intensity slider gains for adjusting the upper bounds

Figure 5.8: Zoomed in details for the Simulink model

Figure 5.9 shows how the movement intensity is used as the leading data type for presence detection. As mentioned before, when no movement is detected the occupancy is zero. This means all data, for the purpose of occupancy estimation, should be zero as well. The system shown accomplishes exactly this.

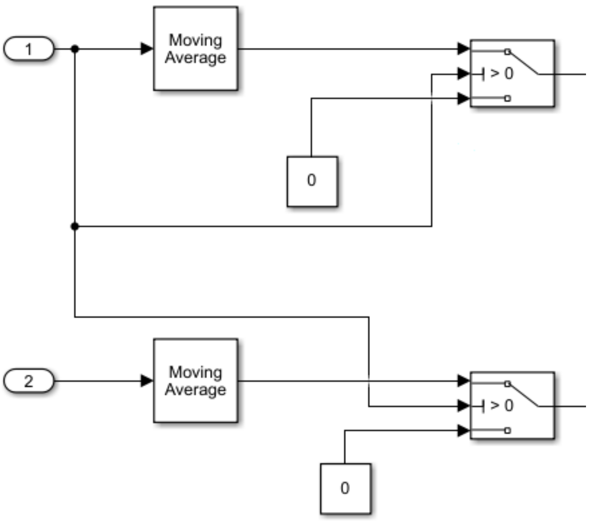


Figure 5.9: Simulink model movement moving average and system for setting data to zero when no movement is detected

Figure 5.10 shows the decision tree for the movement intensity data. The inputs 3 up to and including 6 are the bounds that are shown in Figure 5.8a. As mentioned, the subsystem labelled with 'Decision tree movement' is where the upper bounds determine which regression coefficient is used. Since there are five regression coefficients corresponding to the five different occupancy levels, the five signals need to be combined into one. This is not a problem since at any given time only one of the five will be nonzero and the signals can therefore be added together.

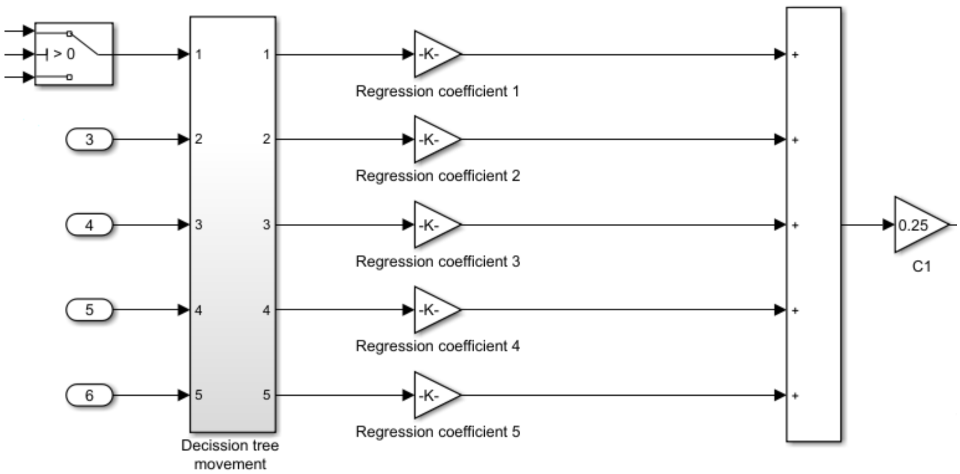


Figure 5.10: Simulink model movement intensity decision tree

Similar to Figure 5.7b, weights are given to both the movement intensity data and the CO_2 concentration data to make the algorithm able to rely more on one or the other. Based on earlier results, the movement intensity is more usable as a presence detection metric while the CO_2 concentration showed to be more accurate for estimating the occupancy. A benefit of using the movement intensity data however, is how quickly it responds to changes in occupancy. Using it in conjunction with the CO_2 concentration will thus enable the system to respond more quickly. However, since accuracy is the main objective the weights will be chosen to give the CO_2 concentration a higher influence in the output.

This method yielded the results shown in Figure 5.11 and was within one person of the target or spot on 93.49% of the time while being exactly correct 75.84% of the time. It may also be useful to express performance in terms of the Root Mean Square Error (RMSE), similar to the papers by Kjærgaard et al. that were discussed in section 3.2 [33, 52]. The method for describing performance chosen in this thesis was thought to be more insightful than the RMSE because it acts as a measure for how accurate the methods are in time. If the method is presented with an input, this performance measure will tell how likely the method is to get a correct result. RMSE on the other hand is an absolute measure for the fit to presented data, with lower values indicating a better fit. The RMSE can be calculated with the formula shown in Equation 5.4.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{5.4}$$

Equation 5.4 shows how the RMSE can be calculated, with $\hat{y}_i$ being the i'th element of the estimated occupancy, y_i being the i'th element of the observed occupancy and n being the total number of samples. Using this the RMSE is calculated to be RMSE=0.78.

The performance gain after combining both data types is impressive. Comparing Figure 5.11 to the previous attempts it is also visually evident that this method performs better. Using the movement intensity alongside the CO_2 concentration, the algorithm was able to react more quickly to changes in occupancy while the CO_2 concentration ensured the algorithm would not miss the target by much. The one caveat to this seemingly excellent performance is that the algorithm was actively adjusted to reach its best performance on this data set. Using the system in its current state on the data sets used to derive the regression coefficients yielded a maximum performance of 56%. A possible reason for why the performance using the validation set is so much higher is the lower complexity of the data set compared to the previous day. Day 1 showed higher occupancy levels in both rooms and more variation throughout the day. Nevertheless the results are acceptable for a linear method. In the next section a non linear method will be used to try and achieve similar or better performance.

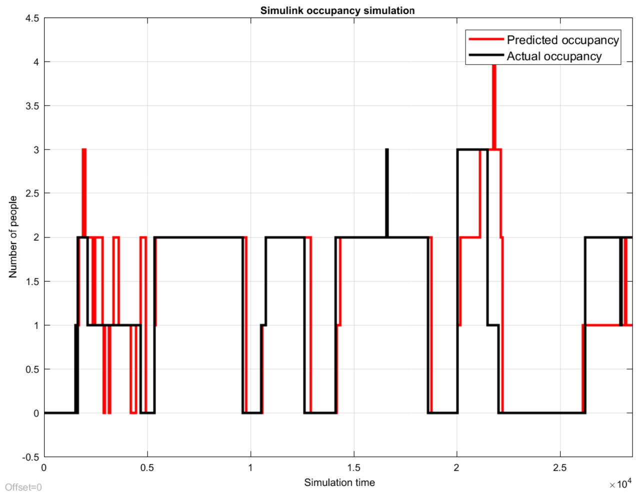


Figure 5.11: Occupancy estimation using the Simulink model

5.2.2. Artificial Neural Network

With the linear regression finished and showing decent performance, the time has come to look at the non-linear method discussed in section 4.2 to see if those results can be improved. The approach for designing an artificial neural network looks as follows:

1. Normalise the input data and construct the input matrix.
2. Generate the target vector using one-hot encoding.
3. Create an initial architecture for testing and debugging.
4. Experiment with different activation functions and/or dropout to improve the results.
5. Use the genetic algorithm to optimise the hyper parameters.
6. Validate the results using the validation data.

Contrary to the linear approach, when designing an artificial neural network normalising the input data is key. Normalising the input data can greatly improve the speed and accuracy of the resulting network [56]. There is a multitude of methods for data normalisation that will not all be discussed. The choice for a certain method over another depends on the specific data. With sufficient knowledge of the minimum and maximum values and low number of outliers in the data set the following pseudocode normalises the data in suitable way:

$$
\begin{aligned}
&for \quad i = 1 : finallength \\
&\qquad data_{normalised}(i) \quad = \quad \frac{data(i) - min(data)}{max(data) - min(data)} \\
&end
\end{aligned}
\tag{5.5}
$$

Where *data* is the specific data (movement intensity, CO_2 concentration e.g.) before normalisation, *max(data)* is the maximum value and *min(data)* is the minimum value.

After normalisation the input vector can be defined. As mentioned before, the training data consists of {room 1 day 1} and {room 1 day 2}, both of which contain temperature, movement intensity, relative humidity, CO_2 concentration and sound data but the sound data will again be omitted. The input vector X will be constructed so that it corresponds with the target vector Y, but before that the target needs to be constructed using *one-hot encoding*. This is because the problem at hand is categorised as a class identification problem. Each possible number of people will be a different class. This means there will be a total of six classes since in the training data there are six possible states for the occupancy.

The minimum is 0 while the maximum occupancy recorded was 5. An example of a six state one-hot encoded vector is shown in Equation 5.6

$$\begin{bmatrix}0\\1\\2\\3\\4\\5\end{bmatrix} \rightarrow \begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{bmatrix} \tag{5.6}$$

The target vector will be the stacked one-hot encoded occupancy vectors of {room 1 day 1} and {room 1 day 2}. Both the input matrix and target vector are shown in Equation 5.7

$$X = \begin{bmatrix} C_1 & M_{1,1} & M_{1,2} & T_{1,1} & T_{1,2} & H_{1,1} & H_{1,2} \\ C_2 & M_{2,1} & M_{2,2} & T_{2,1} & T_{2,2} & H_{2,1} & H_{2,2} \end{bmatrix} \quad , \quad Y = \begin{bmatrix} Y_{1,hot} \\ Y_{2,hot} \end{bmatrix} \tag{5.7}$$

where the subscript *i,j* on the input side represent the day and the node number respectively. The input matrix consists of the [nx1] and [mx1] column vectors CO_2 concentration (C), movement intensity (M), temperature (T) and relative humidity (H) with n and m being the total number of samples for each day. How the individual inputs are arranged is not important as long as it is consistent for the entire matrix. This means the data types from both days need to be structured in the same way, as shown in Equation 5.7. On the target side, the subscript *i,hot* indicates the one hot encoded occupancy vector of the i'th day.

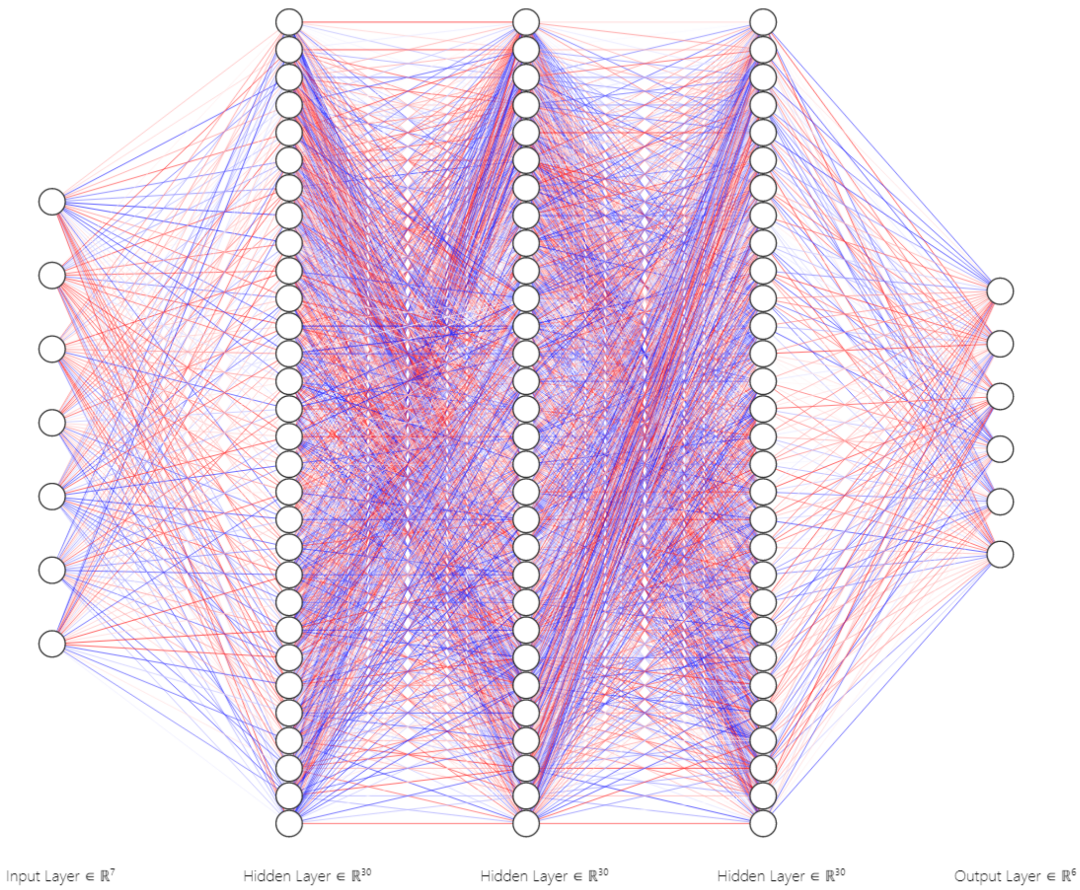
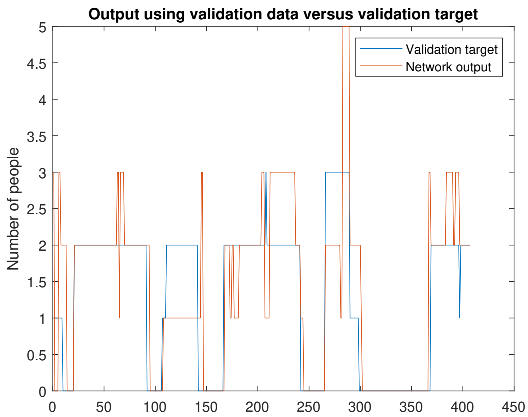


Figure 5.12: Initial architecture of ANN with hyperbolic tangent activation function, trained with back propagation

The initial architecture consisted of three hidden layers with an arbitrary 30 neurons each. That number was chosen as a starting point to already give the network some complexity. The input layer has seven neurons, one for each input. The output layer has six neurons, corresponding to the six possible states in the one-hot encoded vector. The output has to be translated back into a specific occupancy count so that it can be visualised. Back propagation is used as a training method and initially all neurons use the hyperbolic tangent activation function. The initial weights and biases are randomised as discussed previously to avoid activation function saturation. The resulting architecture is shown in Figure 5.12. The colours red and blue indicate the sign of the connection between the neurons while the brightness corresponds with the magnitude; a bright red connection between two neurons for example indicates a large positive weight and bias. This architecture was trained for 1000 epochs with a learning rate of 0.001 and no dropout.

This first architecture mainly served as a trial run to see if the algorithm was working correctly. It was not expected to achieve exceptional performance and in that regard it did not disappoint. The performance was calculated in the same way as before, as shown in Equation 5.3. Using that the initial architecture achieved a peak training performance of 87.84% and a validation performance of 62.16%. When the network misses the target it is usually not off by much. It was calculated to be a maximum of 1 person off the target a total of 7.49% of the time using the training data and 16.46% using the validation data. To calculate the performance, the output of the network first had to be converted to a vector containing the occupancy count. Since the network will never return ones and zeros, this was done by looking at the maximum value in each row. The results were then compared to the target as shown in Figure 5.13.

(a) Training output vs target

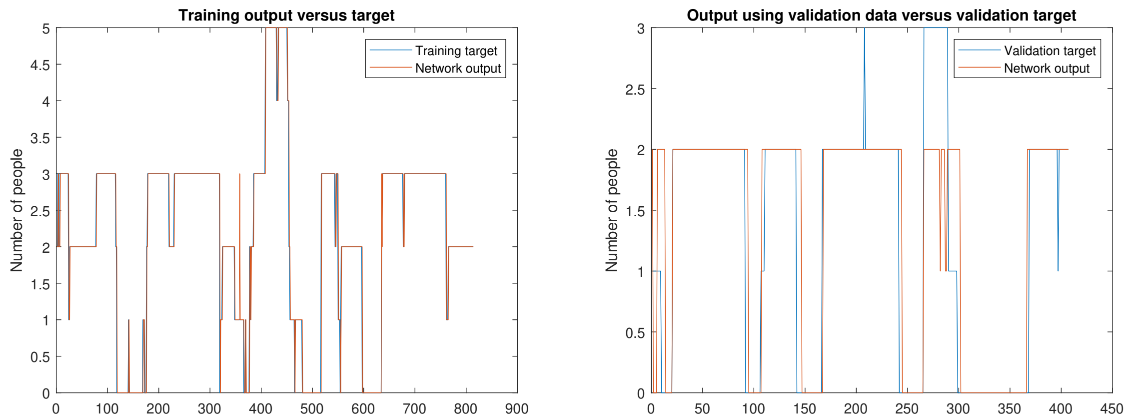


(b) Network output vs target using validation data

Figure 5.13: training results (a) and the validation results (b) of initial network architecture

Figure 5.13a nicely shows the output of the network trying to match the target. Based on the performance and supported by the visual it is fair to say overfitting is not yet an issue. In the next iteration the complexity can be increased and/or a different activation function can be used.

Starting with an increase in complexity in hopes of increasing performance, the number of neurons is increased from thirty to 100 for all three hidden layers. This more than doubles the computation time and looking at the results had an adverse effect. Figure 5.16a shows the network is currently overfitting to the training data. This results in a network that is less responsive than before when presented with new data, resulting in Figure 5.16b.
The performance on the validation set does not look bad at first glance, however it clearly has difficulty reaching occupancy levels other than 2. Calculating the performance resulted in a value of 83.05%, a value that continued to appear each time a network was overfitting to the training data. Other than the number of neurons per layer, everything was kept as it was in the first architecture. The poor performance is especially clear when the data that was used for training is presented to the network. The results are shown in Figure 5.15.



(a) Training output vs target

(b) Network output vs target using validation data

Figure 5.14: training results (a) and the validation results (b) of the more complex architecture, clearly showing overfitting

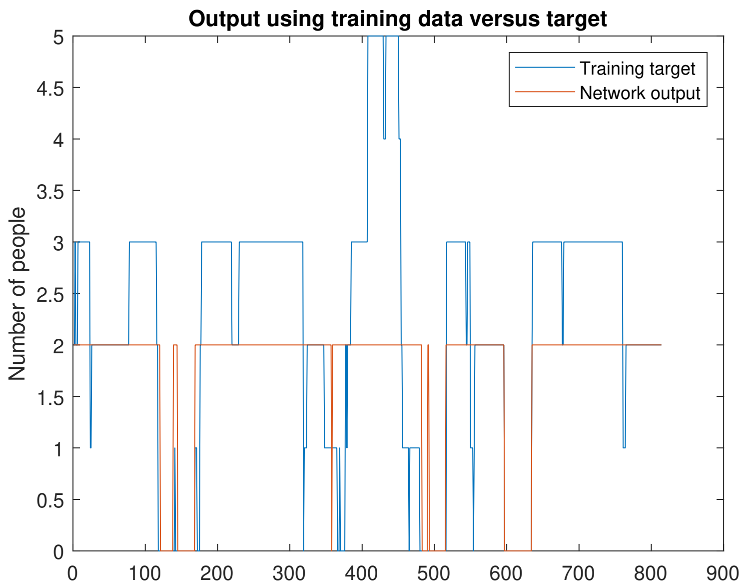


Figure 5.15: Results of using the training data on the trained more complex network, showing poor performance

Using the training data shows the same issues as with the validation data; the network is unable to accurately predict the non-zero occupancy. These are indicators that the current architecture is too complex for the problem at hand and something needs to change.

Before changing the number of neurons to decrease the complexity and with that the issue of overfitting, the *dropout* technique mentioned in section 4.2 is applied. To start with, a *p* value of 0.05 is used and applied to the first hidden layer. This means five of the 100 neurons are randomly dropped every iteration, making it more difficult for the network to overfit. However, this did not have the intended result. Instead of improving the performance, the algorithm produced exponentially increasing errors that eventually resulted in outputs so large Matlab could only display them as NaN. Even reducing the p value did not solve this issue and thus this method was disregarded. The issue is probably caused by the *exploding gradient problem*, a problem common to neural networks using a gradient based training method [46]. It occurs mostly in very deep networks but since randomly dropping neurons essentially deepens the network it is not uncommon to see this issue in more shallow networks. Solutions to the problem include using ResNets (residual neural networks) [58]. However, since this issue showed itself using dropout and it is not evident that dropout is in fact necessary, this does not yet have to be looked into, if at all. Dropout was tried to prevent overfitting but decreasing the network's complexity may also prove effective.

The goal of tweaking the hyper parameters (learning rate and number of neurons per hidden layer) to reach optimal performance is a problem well suited for optimisation. The *genetic algorithm* (GA) was selected as the optimisation method because of its ability to deal with integer variables. The number of neurons per hidden layer has to be an integer and though other methods can be modified to suit this need, problem may arise that can be circumvented by using the genetic algorithm. The downside to using GA is the time it takes for the algorithm to converge, especially when multiple variables are used. GA works by selecting the best properties of a generation through a process of selection, mutation and inheritance to create the next generation. Because of this it can be hard to replicate results and as mentioned before, it can take a long time for the algorithm to reach a global optimum. The parameters to be optimised will be the learning rate (bounded by 10e-6 and 10e-2) and the number of neurons for all 3 individual layers (bounded by 5 and 100). The integer constraint was naturally only applied to the number of neurons. In addition parallel programming was used to speed up the process. The initial optimisation was run using an architecture that used the hyperbolic tangent function for all neurons. The optimisation results are shown in Table 5.3.

Learning rate	0.002
Neurons HL1	20
Neurons HL2	50
Neurons HL3	30
Iterations	56
Performance$_{\textbf{trainingdata}}$	63.14%
Performance$_{\textbf{validationdata}}$	75.18%

Table 5.3: Optimisation results

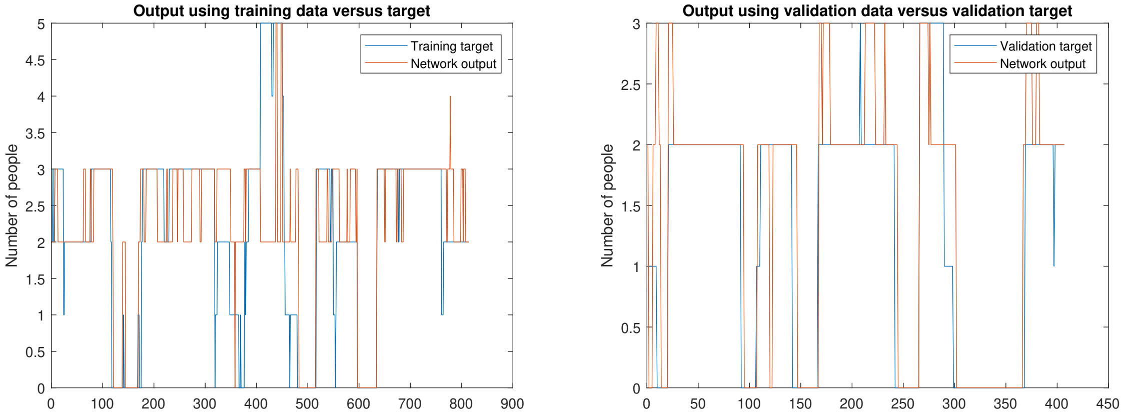
The results from the first optimisation are very comparable to the initial architecture, showing an increase in peak performance using the validation data. The performance on the training data should not be interpreted as training performance. At this point, training performance will be in the range of 90% because all architectures achieve a close fit with the training data during training. The performance of the trained network on the data it was trained with is valuable because it paints a better picture of the training effort. Note that the optimisation returns the optimal architecture which is does not necessarily always produce the highest performance all the time. This depends on the result of training that architecture and in turn on the initial weights and biases. Since these values are randomised to avoid activation function saturation, results may vary. At this point no other activation functions than the hyperbolic tangent were used.

Through an iterative procedure, the *Leaky ReLu* with an *a* value of 0.3 was chosen to be used on the output neurons while keeping the hyperbolic tangent function for the other neurons. The optimisation is ran again, keeping the lower and uppers bounds on the parameters equal to before. The results are shown in Table 5.4.

Learning rate	0.001
Neurons HL1	19
Neurons HL2	21
Neurons HL3	39
Iterations	43
Performance$_{\textbf{trainingdata}}$	61.55%
Performance$_{\textbf{validationdata}}$	76.17%

Table 5.4: Final optimisation results

The final optimisation showed a slight improvement in peak performance on the validation set while staying close to the performance on the training set. The main thing to look for besides raw performance



(a) Network output vs target using training data

(b) Network output vs target using validation data

Figure 5.16: Results of using the training data (a) and the validation data (b) to test the final network architecture

is the network's ability to reach different occupancy levels. Looking back at the overfitting architecture, the network struggled to reach all occupancy levels. Looking at Figure 5.16 the network is able to reach most levels op occupancy between the two data sets while achieving a higher or comparable peak performance when compared to the first architecture. Looking at the networks performance during training shown in Figure 5.17 it achieves a fit close to the example of overfitting, however the results clearly show that overfitting is not an issue with this architecture.

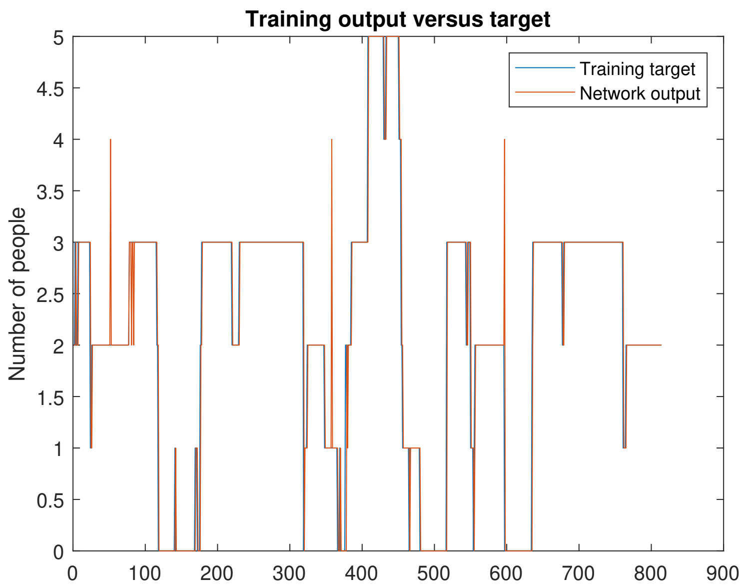


Figure 5.17: Training output vs target using the final network architecture

The final architecture also performed well when looking at near misses. The network missed the target by just 1 person 17.69% of the time using the validation data and 27.89% using the training data. Using Equation 5.4 again, the RMSE is calculated to be RMSE=0.67 for the validation data and RMSE=0.98 for the training data.

At this point the performance has reached the limits of the available data given the current architecture. It stands out that the performance difference between different successful versions of the network described in this section is minimal. This can be attributed to the rigorousness of the training method or the ”luck” that the initial values for the hyper parameters were already close to the values derived by the genetic algorithm. The final architecture is shown in Figure 5.18. Comparing the final architecture to the initial one shown in Figure 5.12, it is clear that the complexity has increased significantly. The increased complexity enables the network to learn smaller details in the training data, adding to the improved performance. However, as was previously shown, adding complexity is not a sure fire way to increase performance.

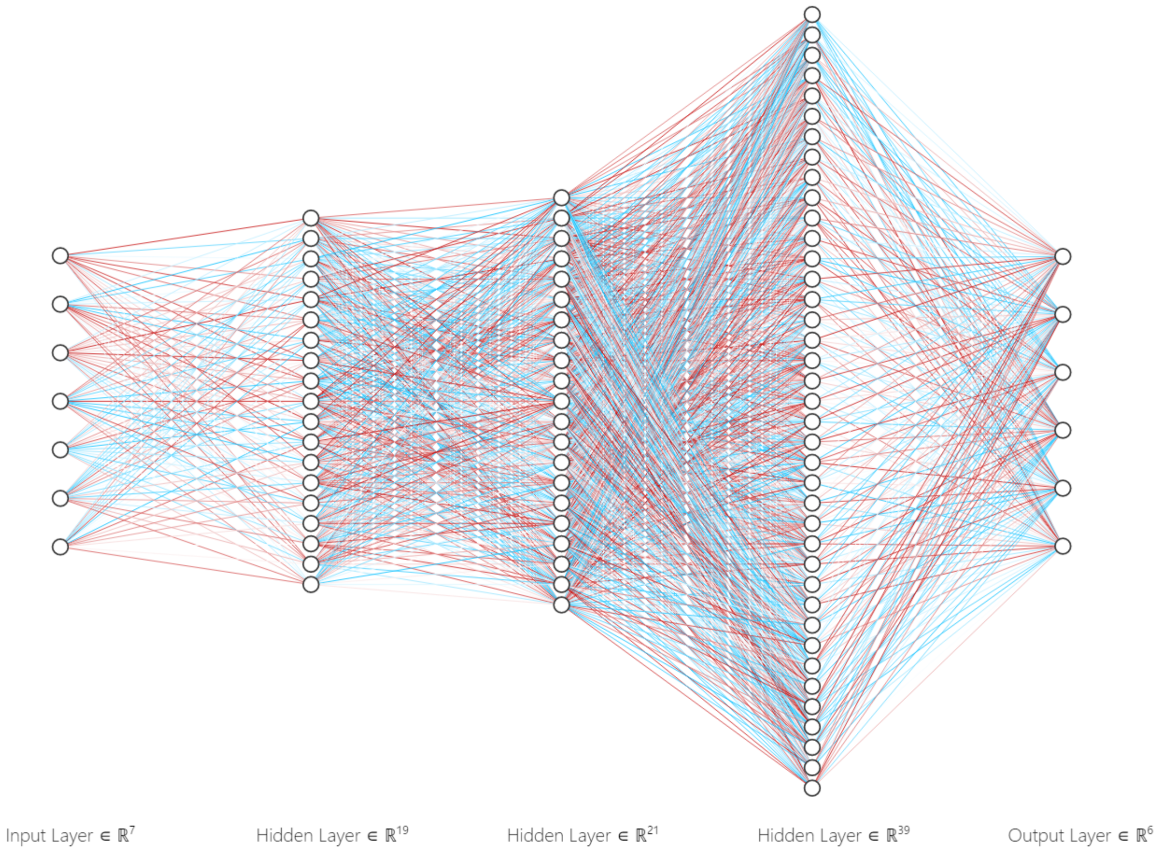


Figure 5.18: Final architecture with mixed *htan* and *Leaky ReLu* activation functions, trained using back propagation

5.3. Conclusion

With both the linear and non-linear methods completed, the results can be compared. A priori it was expected that a non-linear approach would yield better results due to the non-linear nature of the problem. Based on what could be observed there did not seem to be much linearity in the relationship between occupancy and the measured data. Nevertheless the linear approach proved reasonably successful. Table 5.5 compares the absolute performance of all methods, as well as the times the algorithms produced a result that was one more or less than the target and the times where the results were more than one removed from the target.

	$\mathbf{LR_{mov}}$	$\mathbf{LR_{CO_2}}$	$\mathbf{LR_{comb}}$	$\mathbf{ANN_{tanh}}$	$\mathbf{ANN_{mix}}$
Performance	40.50%	65.31%	75.84%	75.18	76.17%
Off by one	not calculated	not calculated	17.65%	15.72%	17.69%
Off by more	not calculated	not calculated	6.51%	9.10%	6.14%

Table 5.5: Performance comparison between the linear and non linear methods using the validation data

Where LR_{mov}, LR_{CO_2} and LR_{comb} are the linear regression methods that used movement, CO_2 and a combination of the two respectively. ANN_{tanh} is the neural network that exclusively used the hyperbolic tangent activation function and ANN_{mix} is the network that used a mix of hyperbolic tangent and *Leaky*

ReLu. For the linear regression methods that only used one data type anything other than absolute performance was not calculated. The reason for this was that the idea had always been to combine multiple data types and one metric for comparison was thought to be sufficient. As mentioned at the start of this section, it is surprising to see how well the best performing linear method compares to the neural network approach. The reason for this was briefly touched upon at the end of subsection 5.2.1, namely that the *Simulink* model based on linear regression was aimed hard at the validation data. Table 5.6 shows the performance drops quite a bit when another data set is used. Here the ANN show their versatility and clearly outperform the linear method.

	$\mathbf{LR_{mov}}$	$\mathbf{ANN_{tanh}}$	$\mathbf{ANN_{mix}}$
Performance	56.09%	63.14%	61.55%
Off by one	33.83%	24.94%	27.89%
Off by more	10.08%	11.92%	10.57%

Table 5.6: Performance comparison between the best linear and non linear methods using the training data

Looking at the values in both tables, it is not immediately clear which method is best. In Table 5.5 both the neural network with mixed activation functions and the linear regression method that combined CO_2 and movement produced very similar results, with the neural network winning only slightly. In Table 5.6 however, the linear regression method clearly showed sub par performance and thus the neural network method with mixed activation functions takes the cake.

6 Conclusion

Having finished the practical side of this thesis and gotten decent results for both methods, it is time to ask what went well and what could have gone better. This chapter will outline exactly that while also comparing the results from the methods described in chapter 5 to the ones discussed in chapter 3 to see if improvements were made and if not, why not. This thesis proposed two methods for estimating real time occupancy using sensor data gathered by the bGrid sensor nodes. After managing the measured data into a usable format and gathering ground truth data as a target, both a linear regression and neural network based approach yielded decent results.

The research that compares best to the work done in this thesis is the research by Kjærgaard et al described in [33]. Calculating the RMSE for the best performing method, the neural network with a mix of hyperbolic tangent and *Leaky ReLu* activation functions, yields an RMSE of RMSE=0.67 compared to RMSE=0.93 for Kjærgaard et al [33]. (lower is better). The achieved RMSE was calculated using the validation data. Using the training data, which produced overall less impressive results, still returned an RMSE of RMSE=0.98. In spite of the lower performance using the training data, this value was close to that found by Kjærgaard et al. It has to be noted at this point that the research mentioned in [33] was done on a much larger scale than the work presented in this thesis, making the results they achieved all the more impressive. The results of other methods discussed in chapter 3 are not directly comparable, except for previous work by Kjærgaard et al. that achieved higher RMSE values and are therefor not useful to compare. The problem statement asked if it would be possible to accurately estimate the occupancy of a room using measured data. Defining performance as the ability of the system to estimate the occupancy within one person of the actual number, the best result linear regression achieved was 93.49% for the validation data and 89.92% for the training data. For the neural network the best result added up to 93.86% for the validation data and 89.44% for the training data. A side note to these results is that, even though the results achieved with linear regression appear to be slightly better than the neural network approach, this only holds when looking at the ability of both models to be within one person of the actual occupancy. The neural network approach was consistently outperforming the linear regression model when only absolute correct predictions were taken into account. With this it can be concluded that the research was a success.

However, a critical look will always find something that can be improved. The main thing with regard to this thesis is the scale of the research. The limiting factor in improving the results is the collected amount of ground truth data, the direct observations. In other research cameras were used to acquire an amount of ground truth data that would be impossible to match when the observations need to be done in person. This also limited the validation that could be done. Ideally an entirely new data set would have been collected to further validate the models, giving them a more solid foundation. Having access to more data would not only affect the validation of the models but the synthesis as well. In the data set used for model synthesis, there was a clear lack of variation in occupancy. Both rooms that were observed did not exceed an occupancy of five people even though both had the capacity for eight people. Additionally the available data for an occupancy of five and four people was very limited as the room was mostly occupied by two or three people. A larger data set would in turn enable the use of methods such as PCR instead of MLR and a possibly deeper neural network, which could improve the performance significantly. In the end, this thesis proposes an approach that can be used as a foundation for future research. Both methods that were discussed can be easily expanded and tweaked if and when more data is collected. The next chapter will outline some possible areas for future research.

7 Future Work

This chapter will outline fields of research where the results from this thesis could prove useful. Recommendations for future work will include both suggestions for the short term or mid term that are achievable within a few months to half a year, and for the long term that will take years to set up and/or complete.

The literature review in chapter 3 already outlined recent studies that did not directly apply to this thesis, but that could act as a basis for future work based on the results found here. As mentioned in the introduction, there is an increasing demand for adaptive temperature controllers. Given the large influence people have on the thermal environment, having accurate knowledge of the real time occupancy could be an excellent metric for adaptive control. Especially when the controller not only tries to maximise user comfort but also conserve energy, having accurate knowledge of the occupancy is essential. In the long term the results of this thesis could therefore be put to great use. The work done by Wang et al. for example, where a system was designed to maximise user comfort while minimising energy consumption, did not use occupancy estimation in their models. Adding the thermal output of occupant to the heat balance can improve the performance of the energy consumption minimisation [60]. Improving the results of the method outlined in the paper by Wang et al. would be a feasible goal for the short to mid term.

However, this relies heavily on the accuracy that an occupancy prediction model could deliver, meaning the models derived in this thesis would first need to be improved by gathering more data. The results in this thesis were based on two days of data and could be improved by increasing that to one or two weeks worth of data. It would however be unrealistic to assume that two weeks of ground truth data would be collected through direct observations. Alternate methods to gather ground truth data could be used to circumvent this issue. Taking privacy into account, and taking into account that it is still desirable to expand this research with similar data which would exclude a controlled setup, installing cameras would not be an option. A method that may be worth looking into is the use of desk sensors in desk-heavy offices. An example of a room mostly filled with desks would be a silent study area at a university or a so called office garden. People occupying such rooms would mostly be at their desk, meaning a desk sensor would accurately measure the occupancy most of the time without causing privacy issues. These could then be used to gather ground truth data to enable the expansion of this research in the short term.

When the models have been improved by using a larger data set, in the long term the models could also be used for office management to better allocate human resources. With accurate knowledge of the occupancy at a room level, predictive models could be generated that would give office managers the tools to streamline processes like cleaning and overall improve the efficiency of the office area. Underused areas could be repurposed while overly crowded areas could be restructured without waiting for occupants to issue complaints. It is usually rare to receive complaints about the under use of office space, meaning parts of an office could go underused for long periods of time without consequence. As mentioned in the introduction of this thesis, a 2013 paper by Norm G. Miller showed bigger companies are transitioning to smaller office footprints to achieve higher utilisation rates [39]. This is a good example of a subject where accurate occupancy estimation is invaluable. Of course this is all speculative but it does show that the possibilities of having accurate knowledge of the occupancy can sprout research in the field of control engineering on one side of the spectrum, as well as human resource management on the other side of the spectrum.

Appendices

A

Simulink model

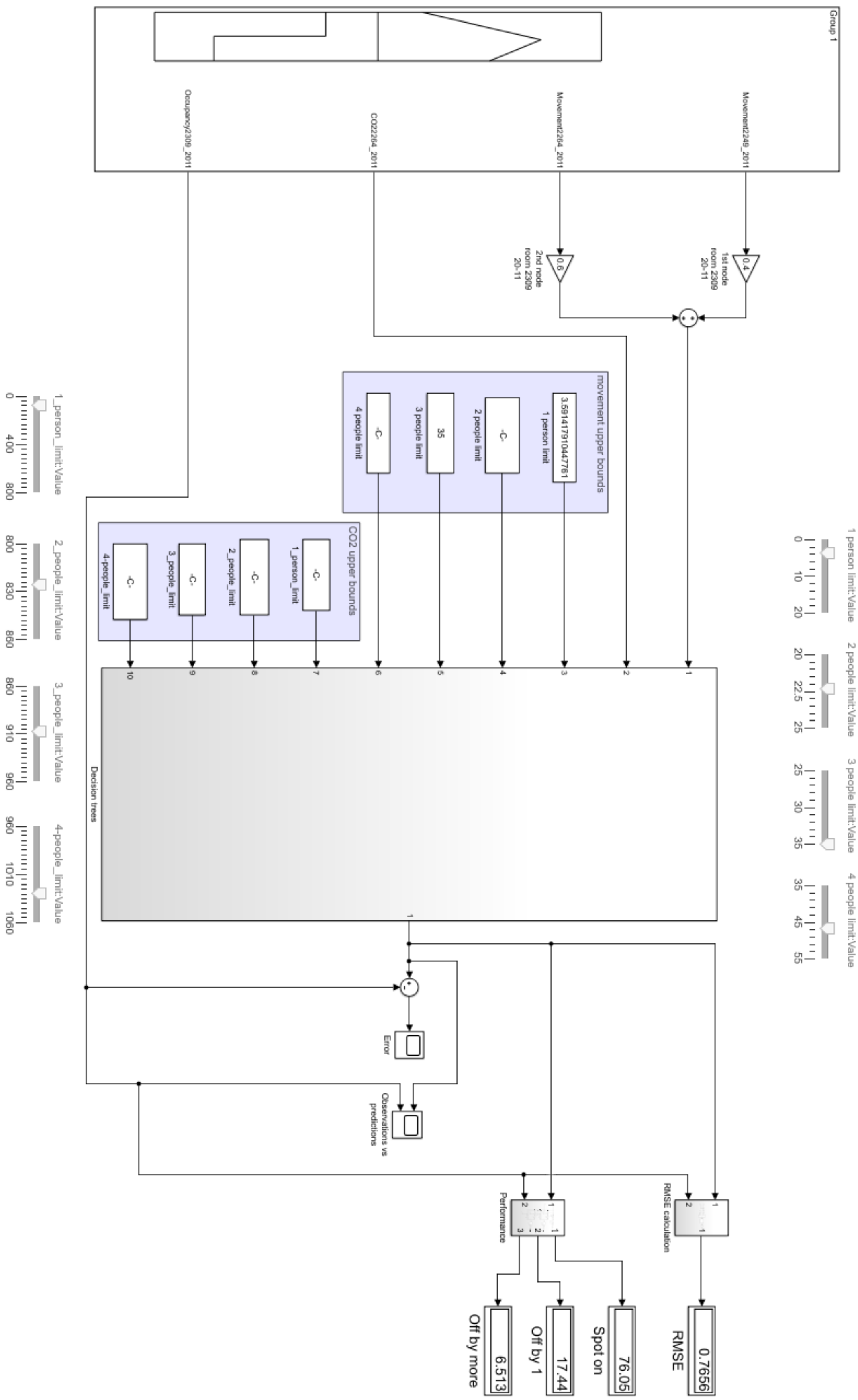


Figure A.1: Main Simulink model

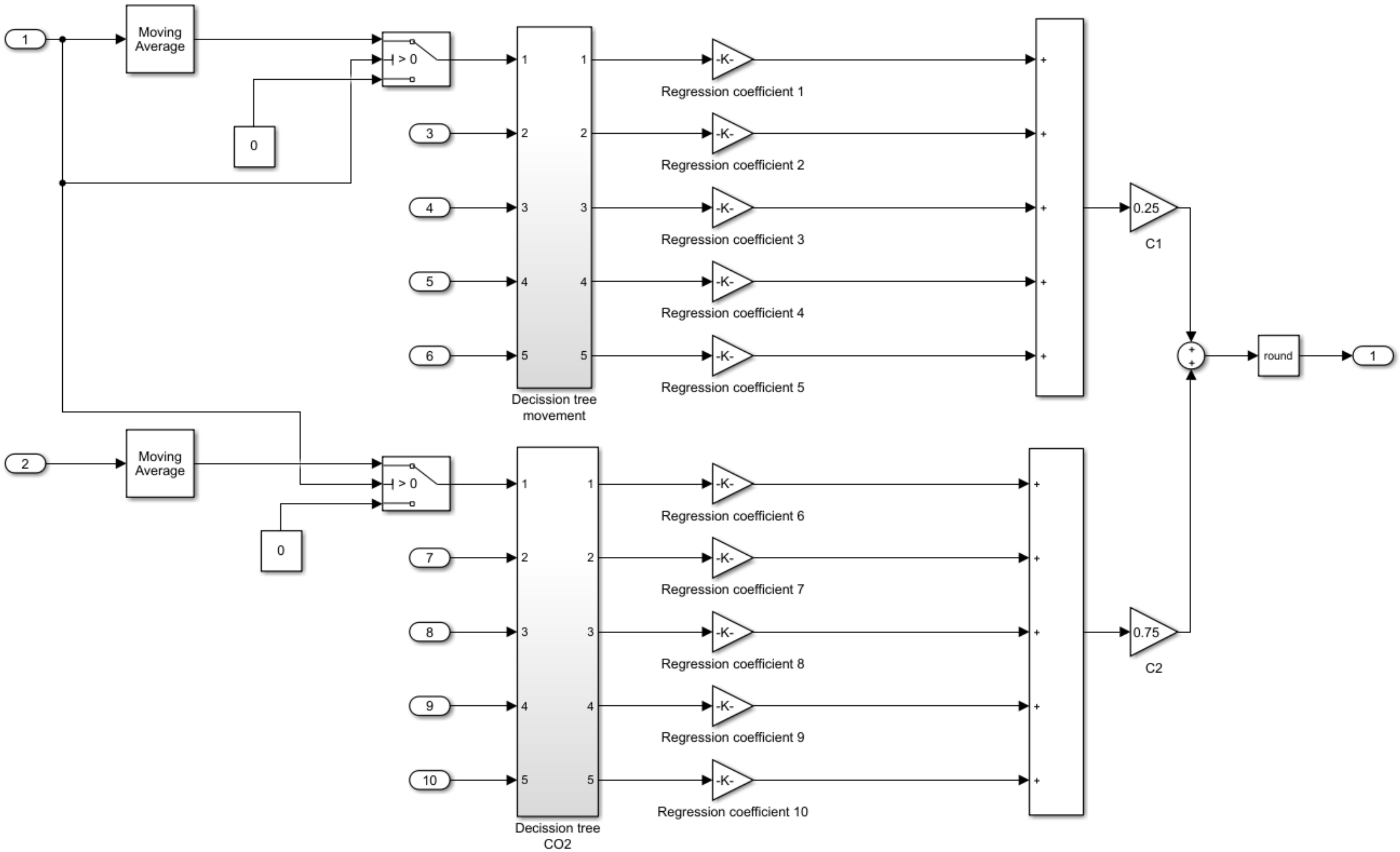


Figure A.2: Combinatory subsytem

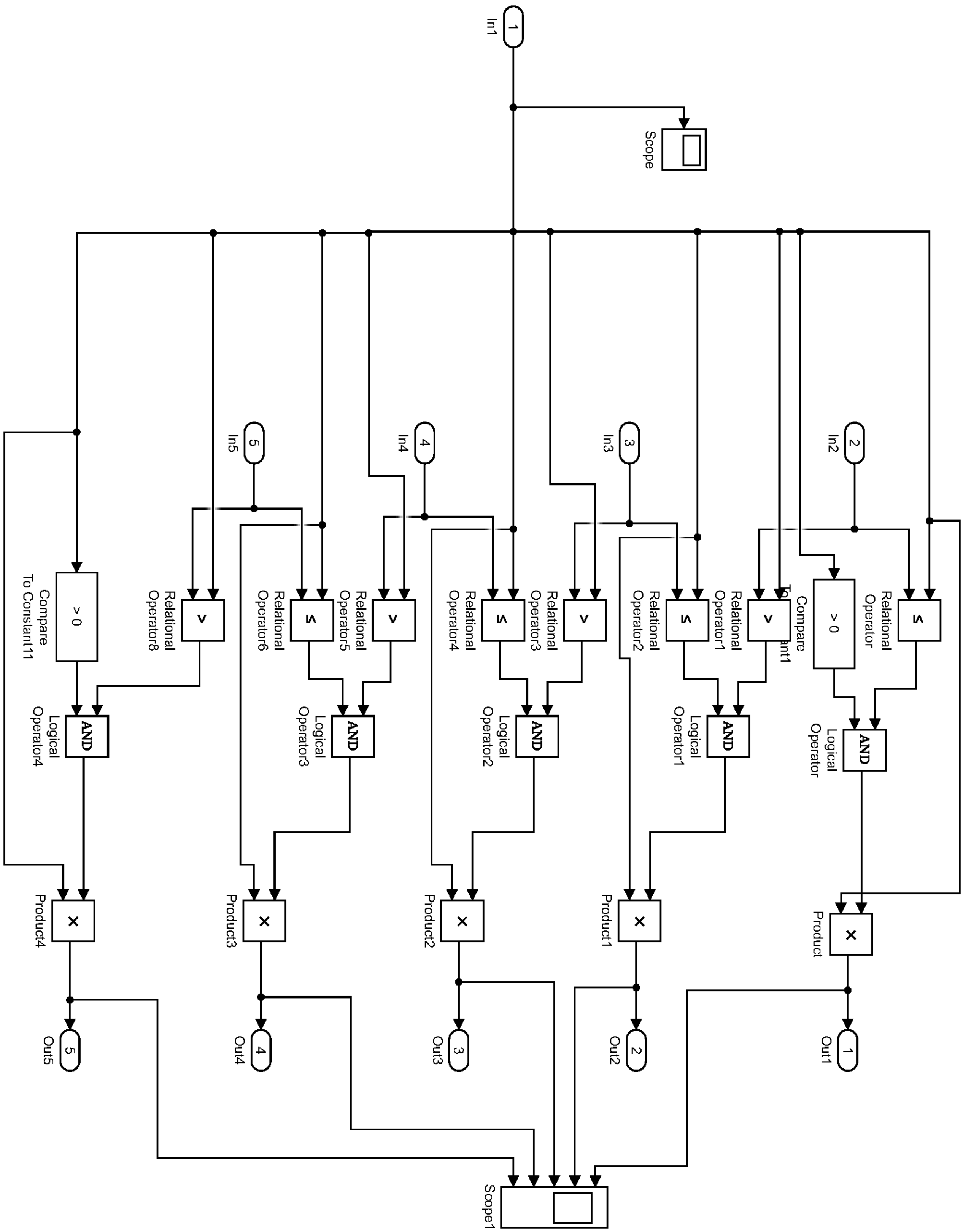


Figure A.3: Movement decision tree

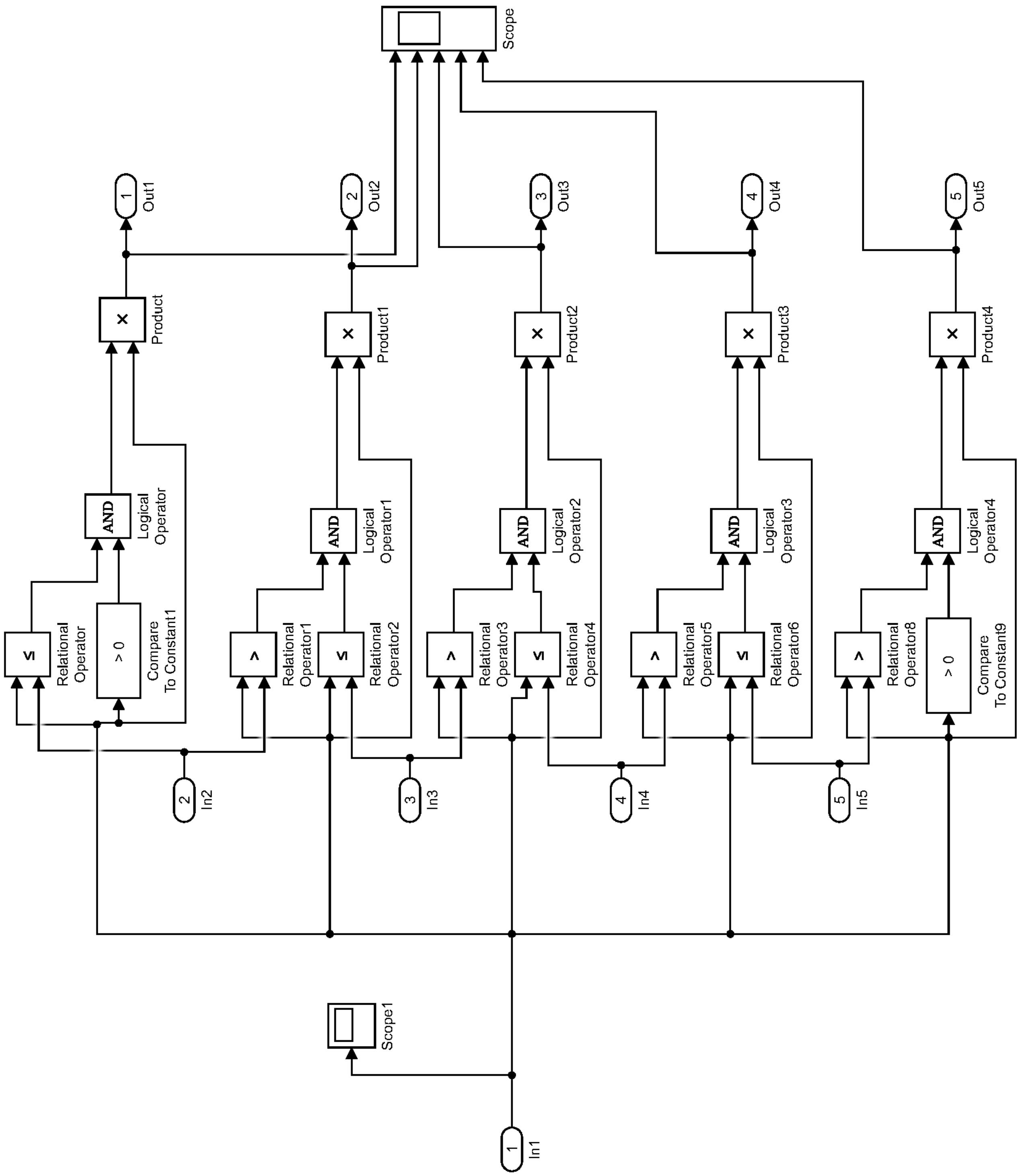


Figure A.4: CO_2 decision tree

B

MATLAB mfiles

Data management and MLR

```
clc
clear all
close all hidden
% This is the main file that starts with loading all data, running data
% management functions and ends with linear regression on the movement
% data and CO2 data separately. The calculated regression coefficients were
% then used for the design of the simulink model.
sound3256_1811=importdata('C:\Users\Badass\Google ...
    Drive\Systems&Control\Thesis\bGrid\Microsoft ...
    data\sound\Node3256_2019-11-18-R3309_sound15.xlsx');
sound3256_1811=sound3256_1811.data.Sheet1;
sound3256_1811(:,4)=sound3256_1811(:,4)- 43787*ones(length(sound3256_1811),1);
sound3256_1811_raw=sound3256_1811;
sound3263_1811=importdata('C:\Users\Badass\Google ...
    Drive\Systems&Control\Thesis\bGrid\Microsoft ...
    data\sound\Node3263_2019-11-18-R3309_sound15.xlsx');
sound3263_1811=sound3263_1811.data.Sheet1;
sound3263_1811(:,4)=sound3263_1811(:,4)- 43787*ones(length(sound3263_1811),1);
sound3263_1811_raw=sound3263_1811;

% Movement
movement3256_1811=importdata('C:\Users\Badass\Google ...
    Drive\Systems&Control\Thesis\bGrid\Microsoft ...
    data\movement\Node3256_2019-11-18-R3309_movement.xlsx');
movement3256_1811=movement3256_1811.data.Sheet1;
movement3256_1811_raw=movement3256_1811;
movement3263_1811=importdata('C:\Users\Badass\Google ...
    Drive\Systems&Control\Thesis\bGrid\Microsoft ...
    data\movement\Node3263_2019-11-18-R3309_movement.xlsx');
movement3263_1811=movement3263_1811.data.Sheet1;
movement3263_1811_raw=movement3263_1811;
% CO2
CO23263_1811=importdata('C:\Users\Badass\Google ...
    Drive\Systems&Control\Thesis\bGrid\Microsoft ...
    data\CO2\Node3263_2019-11-18-R3309_CO2.xlsx');
CO23263_1811=CO23263_1811.data.Sheet1;
CO23263_1811(:,4)=CO23263_1811(:,4)- 43787*ones(length(CO23263_1811),1);
```

```
% Occupancy
occupancy_1811=importdata('C:\Users\Badass\Google ...
      Drive\Systems&Control\Thesis\bGrid\Observations_Microsoft_Office_18-11-2019_R3309.xlsm');
occupancy_1811=occupancy_1811.data.Sheet1;
occupancy_1811=occupancy_1811(2:end,1:2);

% 20-11-2019
% Sound
sound3256_2011=importdata('C:\Users\Badass\Google ...
      Drive\Systems&Control\Thesis\bGrid\Microsoft ...
      data\sound\Node3256_2019-11-20-R3309_sound15.xlsx');
sound3256_2011=sound3256_2011.data.Sheet1;
sound3263_2011=importdata('C:\Users\Badass\Google ...
      Drive\Systems&Control\Thesis\bGrid\Microsoft ...
      data\sound\Node3263_2019-11-20-R3309_sound15.xlsx');
sound3263_2011=sound3263_2011.data.Sheet1;
sound2249_2011=importdata('C:\Users\Badass\Google ...
      Drive\Systems&Control\Thesis\bGrid\Microsoft ...
      data\sound\Node2249_2019-11-20-R2309_sound15.xlsx');
sound2249_2011=sound2249_2011.data.Sheet1;
sound2264_2011=importdata('C:\Users\Badass\Google ...
      Drive\Systems&Control\Thesis\bGrid\Microsoft ...
      data\sound\Node2264_2019-11-20-R2309_sound15.xlsx');
sound2264_2011=sound2264_2011.data.Sheet1;

% Movement
movement3256_2011=importdata('C:\Users\Badass\Google ...
      Drive\Systems&Control\Thesis\bGrid\Microsoft ...
      data\movement\Node3256_2019-11-20-R3309_movement_2.xlsx');
movement3256_2011=movement3256_2011.data.Sheet1;
movement3263_2011=importdata('C:\Users\Badass\Google ...
      Drive\Systems&Control\Thesis\bGrid\Microsoft ...
      data\movement\Node3263_2019-11-20-R3309_movement_2.xlsx');
movement3263_2011=movement3263_2011.data.Sheet1;
movement2249_2011=importdata('C:\Users\Badass\Google ...
      Drive\Systems&Control\Thesis\bGrid\Microsoft ...
      data\movement\Node2249_2019-11-20-R2309_movement_2.xlsx');
movement2249_2011=movement2249_2011.data.Sheet1;
movement2264_2011=importdata('C:\Users\Badass\Google ...
      Drive\Systems&Control\Thesis\bGrid\Microsoft ...
      data\movement\Node2264_2019-11-20-R2309_movement_2.xlsx');
movement2264_2011=movement2264_2011.data.Sheet1;

% CO2
CO23263_2011=importdata('C:\Users\Badass\Google ...
      Drive\Systems&Control\Thesis\bGrid\Microsoft ...
      data\CO2\Node3263_2019-11-20-R3309_CO2.xlsx');
CO23263_2011=CO23263_2011.data.Sheet1;
CO22264_2011=importdata('C:\Users\Badass\Google ...
      Drive\Systems&Control\Thesis\bGrid\Microsoft ...
      data\CO2\Node2264_2019-11-20-R2309_CO2.xlsx');
CO22264_2011=CO22264_2011.data.Sheet1;

% Occupancy
occupancy3309_2011=importdata('C:\Users\Badass\Google ...
      Drive\Systems&Control\Thesis\bGrid\Microsoft ...
      data\Occupancy\Observations_Microsoft_Office_20-11-2019_R3309.xlsm');
occupancy3309_2011=occupancy3309_2011.data.Sheet1;
occupancy3309_2011=occupancy3309_2011(2:end,1:2);
occupancy2309_2011=importdata('C:\Users\Badass\Google ...
      Drive\Systems&Control\Thesis\bGrid\Microsoft ...
      data\Occupancy\Observations_Microsoft_Office_20-11-2019_R2309.xlsx');
occupancy2309_2011=occupancy2309_2011.data.Sheet1;
occupancy2309_2011=occupancy2309_2011(2:end,1:2);

%% Data management
% I've condensed all steps to handle the missing pieces of data,
% introducing NAN's in those places and interpolating to fill the gaps into
% a data management function to clean up the file.

% 18-11
```

```
movement3256_1811t=datamanagement(movement3256_1811);
movement3263_1811t=datamanagement(movement3263_1811);
sound3256_1811t=datamanagement(sound3256_1811);
sound3263_1811t=datamanagement(sound3263_1811);
CO23263_1811t=datamanagement(CO23263_1811);

% 20-11
movement3256_2011t=datamanagement(movement3256_2011);
movement3263_2011t=datamanagement(movement3263_2011);
sound3256_2011t=datamanagement(sound3256_2011);
sound3263_2011t=datamanagement(sound3263_2011);
CO23263_2011t=datamanagement(CO23263_2011);

movement2249_2011t=datamanagement(movement2249_2011);
movement2264_2011t=datamanagement(movement2264_2011);
sound2249_2011t=datamanagement(sound2249_2011);
sound2264_2011t=datamanagement(sound2264_2011);
CO22264_2011t=datamanagement(CO22264_2011);

%% Resampling instead of interpolating and visual comparison
close all
figure
sound3263_1811_RS=resample(sound3263_1811_raw(:,3),round(length(occupancy_1811)),
length(sound3263_1811_raw));
x1=linspace(1,length(sound3263_1811_RS),length(sound3263_1811));
plot(sound3263_1811_RS,'r'); hold on; plot(x1,sound3263_1811(:,3),'b');
title('Sound node 3263 interpolating vs resampling')
legend('Resampled','Interpolated')

% figure
sound3256_1811_RS=resample(sound3256_1811_raw(:,3),round(length(occupancy_1811)),
length(sound3256_1811_raw));
x2=linspace(1,length(sound3256_1811_RS),length(sound3256_1811));
plot(sound3256_1811_RS,'r'); hold on; plot(x2,sound3256_1811(:,3),'b');
title('Sound node 3256 interpolating vs resampling')
legend('Resampled','Interpolated')

figure
movement3263_1811_RS=resample(movement3263_1811_raw(:,3),round(length(occupancy_1811)),
length(movement3263_1811_raw));
x3=linspace(1,length(movement3263_1811_RS),length(movement3263_1811));
plot(movement3263_1811_RS,'r'); hold on; plot(x3,movement3263_1811(:,3),'b');
title('Movement node 3263 interpolating vs resampling')


figure
movement3256_1811_RS=resample(movement3256_1811_raw(:,3),round(length(data)),
length(movement3256_1811_raw(:,3)));
plot(movement3256_1811_RS,'r'); hold on; plot(data(:,3),'b');
title('Movement node 3263 resampling vs raw')
legend('Resampled','raw')
%% raw data images
close all
figure
plot(data(:,3),'b')
title('Raw movement data node 1 18-11')

figure
plot(data2(:,3),'b')
title('Raw movement data node 2 18-11')
% Clearly resampling the data to get rid of the gaps is an inferior
% solution to interpolating. Not only does resampling introduce
% oscillations around points where the data is zero, there are also
% overshoots introduced, generating an overall messier dataset.

%% Matching data points
% It's clear that the points don't exactly line up on the horizontal axis.
% The occupancy is sampled on the minute exactly with 60 second in between.
% The measurements on the other hand tend to vary from node to node as a
% result of differences in firmware. My plan is to look at the dataset with
% the lowest number of samples to take as a baseline (on the 18th that
```

```
% would be the 407 samples in sound3256 and on the 20th the 452 samples in ...
    movement3256). The first step is to
% use the interp1 function that takes the timestamps of the observations as
% x_q, the timestamps of the measurements as x and the values of the
% measurements as y to return y_q, the resampled values matching with x_q.
% 'pchip' outperforms the 'linear' interpolation method in retaining most
% of the shape of the original data. It does mess up the start of the data
% but that isn't unexpected since the observations contain more samples.
% this is something that needs to change, however it's not as easy as just
% cutting the beginning and end off, as the data needs to match between
% themselves. The first step is to look at the beginning of all the data
% and look for the latest timestamp. This is the starting point all other
% data should match their first sample with since it's the limiting factor.
% The same goes for the end of the data where we're looking for the
% earliest timestamp that ends a dataset. This is the timestamp that the
% other datasets should end close to as well.

% 18-11
i=1;
% Occupancy
while movement3256_1811(1,4)-occupancy_1811(i,1)>3.703704000000418e-04
    occupancy_1811(i,:)=[];
end
i=length(occupancy_1811);
while abs(movement3256_1811(end,4)-occupancy_1811(i,1))>3.703704000000418e-04
   occupancy_1811(i,:)=[];
   i=length(occupancy_1811);
end

% Movement
i=1;
while movement3256_1811(1,4)-movement3263_1811(i,4)>3.703704000000418e-04
    movement3263_1811(i,:)=[];
end
i=length(movement3263_1811);
while abs(movement3256_1811(end,4)-movement3263_1811(i,4))>3.703704000000418e-04
   movement3263_1811(i,:)=[];
   i=length(movement3263_1811);
end

% Sound
i=1;
while movement3256_1811(1,4)-sound3256_1811(i,4)>3.703704000000418e-04
    sound3256_1811(i,:)=[];
end
i=length(sound3256_1811);
while abs(movement3256_1811(end,4)-sound3256_1811(i,4))>3.703704000000418e-04
   sound3256_1811(i,:)=[];
   i=length(sound3256_1811);
end
i=1;
while movement3256_1811(1,4)-sound3263_1811(i,4)>3.703704000000418e-04
    sound3263_1811(i,:)=[];
end
i=length(sound3263_1811);
while abs(movement3256_1811(end,4)-sound3263_1811(i,4))>3.703704000000418e-04
   sound3263_1811(i,:)=[];
   i=length(sound3263_1811);
end

% CO2
i=1;
while movement3256_1811(1,4)-CO23263_1811(i,4)>3.703704000000418e-04
    CO23263_1811(i,:)=[];
end
i=length(CO23263_1811);
while abs(movement3256_1811(end,4)-CO23263_1811(i,4))>3.703704000000418e-04
   CO23263_1811(i,:)=[];
   i=length(CO23263_1811);
end
```

```
% 20-11
% Occupancy
while movement2249_2011(1,4)-occupancy2309_2011(i,1)>3.703704000000418e-04
    occupancy2309_2011(i,:)=[];
end
i=length(occupancy2309_2011);
while abs(movement2249_2011(end,4)-occupancy2309_2011(i,1))>3.703704000000418e-04
   occupancy2309_2011(i,:)=[];
   i=length(occupancy2309_2011);
end

while movement3256_2011(1,4)-occupancy3309_2011(i,1)>3.703704000000418e-04
    occupancy3309_2011(i,:)=[];
end
i=length(occupancy3309_2011);
while abs(movement3256_2011(end,4)-occupancy3309_2011(i,1))>7.703704000000418e-04
   occupancy3309_2011(i,:)=[];
   i=length(occupancy3309_2011);
end

% Movement
i=1;
while occupancy3309_2011(1,1)-movement3263_2011(i,4)>3.703704000000418e-04
    movement3263_2011(i,:)=[];
end
i=length(movement3263_2011);
while abs(occupancy3309_2011(end,1)-movement3263_2011(i,4))>3.703704000000418e-04
   movement3263_2011(i,:)=[];
   i=length(movement3263_2011);
end

i=1;
while occupancy2309_2011(1,1)-movement2264_2011(i,4)>3.703704000000418e-04
    movement2264_2011(i,:)=[];
end
i=length(movement2264_2011);
while abs(occupancy2309_2011(end,1)-movement2264_2011(i,4))>3.703704000000418e-04
   movement2264_2011(i,:)=[];
   i=length(movement2264_2011);
end

% Sound
% R3309
i=1;
while occupancy3309_2011(1,1)-sound3256_2011(i,4)>3.703704000000418e-04
    sound3256_2011(i,:)=[];
end
i=length(sound3256_2011);
while abs(occupancy3309_2011(end,1)-sound3256_2011(i,4))>3.703704000000418e-04
   sound3256_2011(i,:)=[];
   i=length(sound3256_2011);
end
i=1;
while occupancy3309_2011(1,1)-sound3263_2011(i,4)>3.703704000000418e-04
    sound3263_2011(i,:)=[];
end
i=length(sound3263_2011);
while abs(occupancy3309_2011(end,1)-sound3263_2011(i,4))>3.703704000000418e-04
   sound3263_2011(i,:)=[];
   i=length(sound3263_2011);
end

% R2309
i=1;
while occupancy2309_2011(1,1)-sound2249_2011(i,4)>3.703704000000418e-04
    sound2249_2011(i,:)=[];
end
i=length(sound2249_2011);
while abs(occupancy2309_2011(end,1)-sound2249_2011(i,4))>3.703704000000418e-04
   sound2249_2011(i,:)=[];
   i=length(sound2249_2011);
```

```
end
i=1;
while occupancy2309_2011(1,1)-sound2264_2011(i,4)>3.703704000000418e-04
    sound2264_2011(i,:)=[];
end
i=length(sound2264_2011);
while abs(occupancy2309_2011(end,1)-sound2264_2011(i,4))>3.703704000000418e-04
   sound2264_2011(i,:)=[];
   i=length(sound2264_2011);
end

i=1;
while occupancy3309_2011(1,1)-movement3263_2011(i,4)>3.703704000000418e-04
    movement3263_2011(i,:)=[];
end
i=length(movement3263_2011);
while abs(occupancy3309_2011(end,1)-movement3263_2011(i,4))>3.703704000000418e-04
   movement3263_2011(i,:)=[];
   i=length(movement3263_2011);
end

% CO2
i=1;
while occupancy3309_2011(1,1)-CO23263_2011(i,4)>3.703704000000418e-04
    CO23263_2011(i,:)=[];
end
i=length(CO23263_2011);
while abs(occupancy3309_2011(end,1)-CO23263_2011(i,4))>3.703704000000418e-04
   CO23263_2011(i,:)=[];
   i=length(CO23263_2011);
end

i=1;
while occupancy2309_2011(1,1)-CO22264_2011(i,4)>3.703704000000418e-04
    CO22264_2011(i,:)=[];
end
i=length(CO22264_2011);
while abs(occupancy2309_2011(end,1)-CO22264_2011(i,4))>3.703704000000418e-04
   CO22264_2011(i,:)=[];
   i=length(CO22264_2011);
end

%% Interpolating all data to match the timestamps

% Since the occupancy was observed on the minute exactly, this is the
% timestamp that will be used for all data.

% 18-11
% Movement
movement3256_1811=movement3256_1811(:,3:4);
y1=interp1(movement3256_1811(:,2),movement3256_1811(:,1),occupancy_1811(:,1),'linear');
movement3256_1811=[y1 occupancy_1811(:,1)];
movement3263_1811=movement3263_1811(:,3:4);
y1=interp1(movement3263_1811(:,2),movement3263_1811(:,1),occupancy_1811(:,1),'linear');
movement3263_1811=[y1 occupancy_1811(:,1)];

% Sound
sound3256_1811=sound3256_1811(:,3:4);
y1=interp1(sound3256_1811(:,2),sound3256_1811(:,1),occupancy_1811(:,1),'linear');
sound3256_1811=[y1 occupancy_1811(:,1)];
sound3263_1811=sound3263_1811(:,3:4);
y1=interp1(sound3263_1811(:,2),sound3263_1811(:,1),occupancy_1811(:,1),'linear');
sound3263_1811=[y1 occupancy_1811(:,1)];

% CO2
CO23263_1811=CO23263_1811(:,3:4);
y1=interp1(CO23263_1811(:,2),CO23263_1811(:,1),occupancy_1811(:,1),'linear');
CO23263_1811=[y1 occupancy_1811(:,1)];

% 20-11
% R3309
```

```
% Movement
movement3256_2011=movement3256_2011(:,3:4);
y1=interp1(movement3256_2011(:,2),movement3256_2011(:,1),occupancy3309_2011(:,1),'linear');
movement3256_2011=[y1 occupancy3309_2011(:,1)];
movement3263_2011=movement3263_2011(:,3:4);
y1=interp1(movement3263_2011(:,2),movement3263_2011(:,1),occupancy3309_2011(:,1),'linear');
movement3263_2011=[y1 occupancy3309_2011(:,1)];

% Sound
sound3256_2011=sound3256_2011(:,3:4);
y1=interp1(sound3256_2011(:,2),sound3256_2011(:,1),occupancy3309_2011(:,1),'linear');
sound3256_2011=[y1 occupancy3309_2011(:,1)];
sound3263_2011=sound3263_2011(:,3:4);
y1=interp1(sound3263_2011(:,2),sound3263_2011(:,1),occupancy3309_2011(:,1),'linear');
sound3263_2011=[y1 occupancy3309_2011(:,1)];

% CO2
CO23263_2011=CO23263_2011(:,3:4);
y1=interp1(CO23263_2011(:,2),CO23263_2011(:,1),occupancy3309_2011(:,1),'linear');
CO23263_2011=[y1 occupancy3309_2011(:,1)];

% R2309
% Movement
movement2249_2011=movement2249_2011(:,3:4);
x=movement2249_2011(:,2);
y=movement2249_2011(:,1);
xi=occupancy2309_2011(:,1);
[x, index] = unique(x);
y1=interp1(x,y(index),xi,'linear');
movement2249_2011=[y1 xi];
movement2264_2011=movement2264_2011(:,3:4);
x=movement2264_2011(:,2);
y=movement2264_2011(:,1);
xi=occupancy2309_2011(:,1);
[x, index] = unique(x);
y1=interp1(x,y(index),xi,'linear');
movement2264_2011=[y1 xi];

% Sound
sound2249_2011=sound2249_2011(:,3:4);
x=sound2249_2011(:,2);
y=sound2249_2011(:,1);
xi=occupancy2309_2011(:,1);
[x, index] = unique(x);
y1=interp1(x,y(index),xi,'linear');
sound2249_2011=[y1 xi];
sound2264_2011=sound2264_2011(:,3:4);
x=sound2264_2011(:,2);
y=sound2264_2011(:,1);
xi=occupancy2309_2011(:,1);
[x, index] = unique(x);
y1=interp1(x,y(index),xi,'linear');
sound2264_2011=[y1 xi];

% CO2
CO22264_2011=CO22264_2011(:,3:4);
x=CO22264_2011(:,2);
y=CO22264_2011(:,1);
xi=occupancy2309_2011(:,1);
[x, index] = unique(x);
y1=interp1(x,y(index),xi,'linear');
CO22264_2011=[y1 xi];

%% Deal with NaN's

movement2249_2011(isnan(movement2249_2011))=nanmean(movement2249_2011(:,1));
movement2264_2011(isnan(movement2264_2011))=nanmean(movement2264_2011(:,1));
movement3256_1811(isnan(movement3256_1811))=nanmean(movement3256_1811(:,1));
movement3256_2011(isnan(movement3256_2011))=nanmean(movement3256_2011(:,1));
movement3263_1811(isnan(movement3263_1811))=nanmean(movement3263_1811(:,1));
movement3263_2011(isnan(movement3263_2011))=nanmean(movement3263_2011(:,1));
```

```
sound2249_2011(isnan(sound2249_2011))=nanmean(sound2249_2011(:,1));
sound2264_2011(isnan(sound2264_2011))=nanmean(sound2264_2011(:,1));
sound3256_1811(isnan(sound3256_1811))=nanmean(sound3256_1811(:,1));
sound3256_2011(isnan(sound3256_2011))=nanmean(sound3256_2011(:,1));
sound3263_1811(isnan(sound3263_1811))=nanmean(sound3263_1811(:,1));
sound3263_2011(isnan(sound3263_2011))=nanmean(sound3263_2011(:,1));

CO22264_2011(isnan(CO22264_2011))=nanmean(CO22264_2011(:,1));
CO23263_1811(isnan(CO23263_1811))=nanmean(CO23263_1811(:,1));
CO23263_2011(isnan(CO23263_2011))=nanmean(CO23263_2011(:,1));


%% Making movement the principal data type
% When no movement is detected, all other data might as well be zero
% because the room is no longer occupied. Hence it would make sense to let
% all other data actually be zero when both nodes detect no movement.

% 18-11
% Sound
for i=1:length(movement3256_1811)
    if movement3256_1811(i,1)==0 && movement3263_1811(i,1)==0
        sound3256_1811(i,1)=0;
        sound3263_1811(i,1)=0;
    end
end
% CO2
for i=1:length(movement3256_1811)
    if movement3256_1811(i,1)==0 && movement3263_1811(i,1)==0
        CO23263_1811(i,1)=0;
    end
end

% 20-11
% Sound
for i=1:length(movement3256_2011)
    if movement3256_2011(i,1)==0 && movement3263_2011(i,1)==0
        sound3256_2011(i,1)=0;
        sound3263_2011(i,1)=0;
    end
end
for i=1:length(movement2249_2011)
    if movement2249_2011(i,1)==0 && movement2249_2011(i,1)==0
        sound2249_2011(i,1)=0;
        sound2264_2011(i,1)=0;
    end
end
% CO2
for i=1:length(movement3256_2011)
    if movement3256_2011(i,1)==0 && movement3263_2011(i,1)==0
        CO23263_2011(i,1)=0;
    end
end
for i=1:length(movement2249_2011)
    if movement2249_2011(i,1)==0 && movement2264_2011(i,1)==0
        CO22264_2011(i,1)=0;
    end
end

%% Splitting data based on occupancy
% To give MCR and PCR its best chance, I'll split the data based on the
% occupancy. This means all data corresponding with an occupancy of a
% certain amount of people will be grouped together. Of course, this will
% not be the case in real life scenarios but if these methods are
% successful in finding consistent relations between data and a fixed
% occupancy, it may end up working.

% 18-11
for i=1:length(occupancy_1811)
    if occupancy_1811(i,2)==1
        occ_1811_1(i,1)=occupancy_1811(i,2);
```

```
            CO23263_1811_1(i,1)=CO23263_1811(i,1);
            mov3256_1811_1(i,1)=movement3256_1811(i,1);
            mov3263_1811_1(i,1)=movement3263_1811(i,1);
            sou3256_1811_1(i,1)=sound3256_1811(i,1);
            sou3263_1811_1(i,1)=sound3263_1811(i,1);
        elseif occupancy_1811(i,2)==2
            occ_1811_2(i,1)=occupancy_1811(i,2);
            CO23263_1811_2(i,1)=CO23263_1811(i,1);
            mov3256_1811_2(i,1)=movement3256_1811(i,1);
            mov3263_1811_2(i,1)=movement3263_1811(i,1);
            sou3256_1811_2(i,1)=sound3256_1811(i,1);
            sou3263_1811_2(i,1)=sound3263_1811(i,1);
        elseif occupancy_1811(i,2)==3
            occ_1811_3(i,1)=occupancy_1811(i,2);
            CO23263_1811_3(i,1)=CO23263_1811(i,1);
            mov3256_1811_3(i,1)=movement3256_1811(i,1);
            mov3263_1811_3(i,1)=movement3263_1811(i,1);
            sou3256_1811_3(i,1)=sound3256_1811(i,1);
            sou3263_1811_3(i,1)=sound3263_1811(i,1);
         elseif occupancy_1811(i,2)==4
            occ_1811_4(i,1)=occupancy_1811(i,2);
            CO23263_1811_4(i,1)=CO23263_1811(i,1);
            mov3256_1811_4(i,1)=movement3256_1811(i,1);
            mov3263_1811_4(i,1)=movement3263_1811(i,1);
            sou3256_1811_4(i,1)=sound3256_1811(i,1);
            sou3263_1811_4(i,1)=sound3263_1811(i,1);
        end
end

occ_1811_1=occ_1811_1(occ_1811_1≠0);
occ_1811_2=occ_1811_2(occ_1811_2≠0);
occ_1811_3=occ_1811_3(occ_1811_3≠0);

CO23263_1811_1= CO23263_1811_1( CO23263_1811_1≠0);
CO23263_1811_2= CO23263_1811_2( CO23263_1811_2≠0);
CO23263_1811_3= CO23263_1811_3( CO23263_1811_3≠0);

mov3256_1811_1=mov3256_1811_1(mov3256_1811_1≠0);
mov3256_1811_2=mov3256_1811_2(mov3256_1811_2≠0);
mov3256_1811_3=mov3256_1811_3(mov3256_1811_3≠0);

mov3263_1811_1=mov3263_1811_1(mov3263_1811_1≠0);
mov3263_1811_2=mov3263_1811_2(mov3263_1811_2≠0);
mov3263_1811_3=mov3263_1811_3(mov3263_1811_3≠0);

sou3256_1811_1=sou3256_1811_1(sou3256_1811_1≠0);
sou3256_1811_2=sou3256_1811_2(sou3256_1811_2≠0);
sou3256_1811_3=sou3256_1811_3(sou3256_1811_3≠0);

sou3263_1811_1=sou3263_1811_1(sou3263_1811_1≠0);
sou3263_1811_2=sou3263_1811_2(sou3263_1811_2≠0);
sou3263_1811_3=sou3263_1811_3(sou3263_1811_3≠0);

% 20-11
for i=1:length(occupancy3309_2011)
    if occupancy3309_2011(i,2)==1
            occ3309_2011_1(i,1)=occupancy3309_2011(i,2);
            CO23263_2011_1(i,1)=CO23263_2011(i,1);
            mov3256_2011_1(i,1)=movement3256_2011(i,1);
            mov3263_2011_1(i,1)=movement3263_2011(i,1);
            sou3256_2011_1(i,1)=sound3256_2011(i,1);
            sou3263_2011_1(i,1)=sound3263_2011(i,1);
        elseif occupancy3309_2011(i,2)==2
            occ3309_2011_2(i,1)=occupancy3309_2011(i,2);
            CO23263_2011_2(i,1)=CO23263_2011(i,1);
            mov3256_2011_2(i,1)=movement3256_2011(i,1);
            mov3263_2011_2(i,1)=movement3263_2011(i,1);
            sou3256_2011_2(i,1)=sound3256_2011(i,1);
            sou3263_2011_2(i,1)=sound3263_2011(i,1);
        elseif occupancy3309_2011(i,2)==3
            occ3309_2011_3(i,1)=occupancy3309_2011(i,2);
```

```
        CO23263_2011_3(i,1)=CO23263_2011(i,1);
        mov3256_2011_3(i,1)=movement3256_2011(i,1);
        mov3263_2011_3(i,1)=movement3263_2011(i,1);
        sou3256_2011_3(i,1)=sound3256_2011(i,1);
        sou3263_2011_3(i,1)=sound3263_2011(i,1);
    elseif occupancy3309_2011(i,2)==4
        occ3309_2011_4(i,1)=occupancy3309_2011(i,2);
        CO23263_2011_4(i,1)=CO23263_2011(i,1);
        mov3256_2011_4(i,1)=movement3256_2011(i,1);
        mov3263_2011_4(i,1)=movement3263_2011(i,1);
        sou3256_2011_4(i,1)=sound3256_2011(i,1);
        sou3263_2011_4(i,1)=sound3263_2011(i,1);
    elseif occupancy3309_2011(i,2)==5
        occ3309_2011_5(i,1)=occupancy3309_2011(i,2);
        CO23263_2011_5(i,1)=CO23263_2011(i,1);
        mov3256_2011_5(i,1)=movement3256_2011(i,1);
        mov3263_2011_5(i,1)=movement3263_2011(i,1);
        sou3256_2011_5(i,1)=sound3256_2011(i,1);
        sou3263_2011_5(i,1)=sound3263_2011(i,1);
    end
end

occ3309_2011_1=occ3309_2011_1(occ3309_2011_1≠0);
occ3309_2011_2=occ3309_2011_2(occ3309_2011_2≠0);
occ3309_2011_3=occ3309_2011_3(occ3309_2011_3≠0);
occ3309_2011_4=occ3309_2011_4(occ3309_2011_4≠0);
occ3309_2011_5=occ3309_2011_5(occ3309_2011_5≠0);

CO23263_2011_1= CO23263_2011_1( CO23263_2011_1≠0);
CO23263_2011_2= CO23263_2011_2( CO23263_2011_2≠0);
CO23263_2011_3= CO23263_2011_3( CO23263_2011_3≠0);
CO23263_2011_4= CO23263_2011_4( CO23263_2011_4≠0);
CO23263_2011_5= CO23263_2011_5( CO23263_2011_5≠0);

mov3256_2011_1=mov3256_2011_1(mov3256_2011_1≠0);
mov3256_2011_2=mov3256_2011_2(mov3256_2011_2≠0);
mov3256_2011_3=mov3256_2011_3(mov3256_2011_3≠0);
mov3256_2011_4=mov3256_2011_4(mov3256_2011_4≠0);
mov3256_2011_5=mov3256_2011_5(mov3256_2011_5≠0);

mov3263_2011_1=mov3263_2011_1(mov3263_2011_1≠0);
mov3263_2011_2=mov3263_2011_2(mov3263_2011_2≠0);
mov3263_2011_3=mov3263_2011_3(mov3263_2011_3≠0);
mov3263_2011_4=mov3263_2011_4(mov3263_2011_4≠0);
mov3263_2011_5=mov3263_2011_5(mov3263_2011_5≠0);

sou3256_2011_1=sou3256_2011_1(sou3256_2011_1≠0);
sou3256_2011_2=sou3256_2011_2(sou3256_2011_2≠0);
sou3256_2011_3=sou3256_2011_3(sou3256_2011_3≠0);
sou3256_2011_4=sou3256_2011_4(sou3256_2011_4≠0);
sou3256_2011_5=sou3256_2011_5(sou3256_2011_5≠0);

sou3263_2011_1=sou3263_2011_1(sou3263_2011_1≠0);
sou3263_2011_2=sou3263_2011_2(sou3263_2011_2≠0);
sou3263_2011_3=sou3263_2011_3(sou3263_2011_3≠0);
sou3263_2011_4=sou3263_2011_4(sou3263_2011_4≠0);
sou3263_2011_5=sou3263_2011_5(sou3263_2011_5≠0);

for i=1:length(occupancy2309_2011)
    if occupancy2309_2011(i,2)==1
        occ2309_2011_1(i,1)=occupancy2309_2011(i,2);
        CO22264_2011_1(i,1)=CO22264_2011(i,1);
        mov2249_2011_1(i,1)=movement2249_2011(i,1);
        mov2264_2011_1(i,1)=movement2264_2011(i,1);
        sou2249_2011_1(i,1)=sound2249_2011(i,1);
        sou2264_2011_1(i,1)=sound2264_2011(i,1);
    elseif occupancy2309_2011(i,2)==2
        occ2309_2011_2(i,1)=occupancy2309_2011(i,2);
        CO22264_2011_2(i,1)=CO22264_2011(i,1);
        mov2249_2011_2(i,1)=movement2249_2011(i,1);
        mov2264_2011_2(i,1)=movement2264_2011(i,1);
```

```
        sou2249_2011_2(i,1)=sound2249_2011(i,1);
        sou2264_2011_2(i,1)=sound2264_2011(i,1);
    elseif occupancy2309_2011(i,2)==3
        occ2309_2011_3(i,1)=occupancy2309_2011(i,2);
        CO22264_2011_3(i,1)=CO22264_2011(i,1);
        mov2249_2011_3(i,1)=movement2249_2011(i,1);
        mov2264_2011_3(i,1)=movement2264_2011(i,1);
        sou2249_2011_3(i,1)=sound2249_2011(i,1);
        sou2264_2011_3(i,1)=sound2264_2011(i,1);
     elseif occupancy2309_2011(i,2)==4
        occ2309_2011_4(i,1)=occupancy2309_2011(i,2);
        CO22264_2011_4(i,1)=CO22264_2011(i,1);
        mov2249_2011_4(i,1)=movement2249_2011(i,1);
        mov2264_2011_4(i,1)=movement2264_2011(i,1);
        sou2249_2011_4(i,1)=sound2249_2011(i,1);
        sou2264_2011_4(i,1)=sound2264_2011(i,1);
     elseif occupancy2309_2011(i,2)==5
        occ2309_2011_5(i,1)=occupancy2309_2011(i,2);
        CO22264_2011_5(i,1)=CO22264_2011(i,1);
        mov2249_2011_5(i,1)=movement2249_2011(i,1);
        mov2264_2011_5(i,1)=movement2264_2011(i,1);
        sou2249_2011_5(i,1)=sound2249_2011(i,1);
        sou2264_2011_5(i,1)=sound2264_2011(i,1);
    end
end

occ2309_2011_1=occ2309_2011_1(occ2309_2011_1≠0);
occ2309_2011_2=occ2309_2011_2(occ2309_2011_2≠0);
occ2309_2011_3=occ2309_2011_3(occ2309_2011_3≠0);

CO22264_2011_1= CO22264_2011_1( CO22264_2011_1≠0);
CO22264_2011_2= CO22264_2011_2( CO22264_2011_2≠0);
CO22264_2011_3= CO22264_2011_3( CO22264_2011_3≠0);

mov2249_2011_1=mov2249_2011_1(mov2249_2011_1≠0);
mov2249_2011_2=mov2249_2011_2(mov2249_2011_2≠0);
mov2249_2011_3=mov2249_2011_3(mov2249_2011_3≠0);

mov2264_2011_1=mov2264_2011_1(mov2264_2011_1≠0);
mov2264_2011_2=mov2264_2011_2(mov2264_2011_2≠0);
mov2264_2011_3=mov2264_2011_3(mov2264_2011_3≠0);

sou2249_2011_1=sou2249_2011_1(sou2249_2011_1≠0);
sou2249_2011_2=sou2249_2011_2(sou2249_2011_2≠0);
sou2249_2011_3=sou2249_2011_3(sou2249_2011_3≠0);

sou2264_2011_1=sou2264_2011_1(sou2264_2011_1≠0);
sou2264_2011_2=sou2264_2011_2(sou2264_2011_2≠0);
sou2264_2011_3=sou2264_2011_3(sou2264_2011_3≠0);

%% Plots, data split by occupancy
close all
x1=linspace(1,length(occ_1811_1),length(occ_1811_1));
x2=linspace(1,length(occ_1811_2),length(occ_1811_2));
x3=linspace(1,length(occ_1811_3),length(occ_1811_3));
meanmov_1=(mean(mov3256_1811_1)+mean(mov3263_1811_1))/2
meanmov_2=(mean(mov3256_1811_2)+mean(mov3263_1811_2))/2
meanmov_3=(mean(mov3256_1811_3)+mean(mov3263_1811_3))/2

% 18-11
% Movement
figure
plot(mov3256_1811_1(:,1),'g.-')
hold on
plot(mov3263_1811_1(:,1),'b.-')
hold on
plot(x1,ones(length(x1),1)*meanmov_1)
title('Movement data with occupancy of 1, room 1 18-11')
legend('Node 1','Node 2', 'average value')
ylabel('movement intensity')
```

```
figure
yyaxis right
plot(mov3256_1811_2(:,1),'g.-')
hold on
plot(mov3263_1811_2(:,1),'b.-')
hold on
plot(x2,ones(length(x2),1)*meanmov_2)
hold on
yyaxis left
plot(occ_1811_2(:,1),'k','linewidth', 2)
title('Movement data vs occupancy of 2, room 1 18-11')
legend('Occupancy','Node 1','Node 2', 'average value')
yyaxis left
ylim([0 5])
ylabel('number of people')
yyaxis right
ylabel('movement intensity')

figure
plot(mov3256_1811_3(:,1),'g.-')
hold on
plot(mov3263_1811_3(:,1),'b.-')
hold on
plot(x3,ones(length(x3),1)*meanmov_3)
title('Movement data with occupancy of 3, room 1 18-11')
legend('Node 1','Node 2', 'average value')
ylabel('movement intensity')

% CO2
movmeanCO2_1=movmean(CO23263_1811_1,5);
movmeanCO2_2=movmean(CO23263_1811_2,5);
movmeanCO2_3=movmean(CO23263_1811_3,5);
meanCO2_1=mean(movmeanCO2_1)
meanCO2_2=mean(movmeanCO2_2)
meanCO2_3=mean(movmeanCO2_3)

figure
yyaxis right
plot(CO23263_1811_1(:,1),'b.-')
hold on
plot(movmeanCO2_1)
hold on
yyaxis left
plot(occ_1811_1(:,1),'k','linewidth', 2)
title('CO2 data vs occupancy of 1, room 1 18-11')
legend('Occupancy','Node 1', 'Moving average window 10')
yyaxis left
ylim([0 5])
ylabel('number of people')
yyaxis right
ylabel('CO2 concentration')

figure
yyaxis right
plot(CO23263_1811_2(:,1),'b.-')
hold on
plot(movmeanCO2_2)
hold on
yyaxis left
plot(occ_1811_2(:,1),'k','linewidth', 2)
title('CO2 data vs occupancy of 2, room 1 18-11')
legend('Occupancy','Node 1', 'Moving average window 10')
yyaxis left
ylim([0 5])
ylabel('number of people')
yyaxis right
ylabel('CO2 concentration')

figure
yyaxis right
plot(CO23263_1811_3(:,1),'b.-')
```

```
hold on
plot(movmeanCO2_3)
hold on
yyaxis left
plot(occ_1811_3(:,1),'k','linewidth', 2)
title('CO2 data vs occupancy of 3, room 1 18-11')
legend('Occupancy','Node 1', 'Moving average window 10')
yyaxis left
ylim([0 5])
ylabel('number of people')
yyaxis right
ylabel('CO2 concentration')

% sound
meansou_1=(mean(sou3256_1811_1)+mean(sou3263_1811_1))/2
meansou_2=(mean(sou3256_1811_2)+mean(sou3263_1811_2))/2
meansou_3=(mean(sou3256_1811_3)+mean(sou3263_1811_3))/2
figure
yyaxis right
plot(sou3256_1811_1(:,1),'g.-')
hold on
plot(sou3263_1811_1(:,1),'b.-')
hold on
plot(x1,ones(length(x1),1)*meansou_1)
hold on
yyaxis left
plot(occ_1811_1(:,1),'k','linewidth', 2)
title('Sound data vs occupancy of 1, room 1 18-11')
legend('Occupancy','Node 1','Node 2', 'average value')
yyaxis left
ylim([0 5])
ylabel('number of people')
yyaxis right
ylabel('sound intensity')

figure
yyaxis right
plot(sou3256_1811_2(:,1),'g.-')
hold on
plot(sou3263_1811_2(:,1),'b.-')
hold on
plot(x2,ones(length(x2),1)*meansou_2)
hold on
yyaxis left
plot(occ_1811_2(:,1),'k','linewidth', 2)
title('Sound data vs occupancy of 2, room 1 18-11')
legend('Occupancy','Node 1','Node 2', 'average value')
yyaxis left
ylim([0 5])
ylabel('number of people')
yyaxis right
ylabel('sound intensity')

figure
yyaxis right
plot(sou3256_1811_3(:,1),'g.-')
hold on
plot(sou3263_1811_3(:,1),'b.-')
hold on
plot(x3,ones(length(x3),1)*meansou_3)
hold on
yyaxis left
plot(occ_1811_3(:,1),'k','linewidth', 2)
title('Sound data vs occupancy of 3, room 1 18-11')
legend('Occupancy','Node 1','Node 2', 'average value')
yyaxis left
ylim([0 5])
ylabel('number of people')
yyaxis right
ylabel('sound intensity')
```

```
% It's clear that the sound shows little correlation with the occupancy,
% looking at the average values. Looking at the average values for movement
% and CO2 the correlation is more clear. For CO2 taking the moving average
% with a larger window seemed to increase the difference in the overall
% averages between different occupancies. Of course, this is given a
% constant occupancy. Based on this I can try to use a bigger window for
% calculating the moving average on the full data set while using the
% linear regression coefficients calculated from the current data set.

%% Linear regression analysis on CO2
% y=X*b+e with y nx1 column vector containing the occupancy, X an nxm
% matrix containing the inputs, b an mx1 row vecor containing regression
% coefficients and e a nx1 column vector containing the residual errors.
% The objective is to minimise this error, yielding the objective function
% L= min_b    (y-Xb)'(y-Xb)

close all
% occupancy 1
y=occ_1811_1;
X=movmeanCO2_1;
b_CO2_1=inv((X'*X))*X'*y;

% occupancy 2
y=occ_1811_2;
X=movmeanCO2_2;
b_CO2_2=inv((X'*X))*X'*y;

% occupancy 3
y=occ_1811_3;
X=movmeanCO2_3;
b_CO2_3=inv((X'*X))*X'*y


%% Testing linear regression coefficients on validation data
% Now that I've used linear regression to get the coefficients that relate
% the input (CO2) to the output (occupancy), I can create a loop that uses
% these coefficients to generate a predicted occupancy based on CO2 data
% from the second day and/or the other room and compare that to the actual
% occupancy. Using a moving average with a bigger window on the input
% seemed to show a higher correlation with the input but yielded the exact
% same regression coefficients as a moving average with a smaller window so
% the smallest window of 5 samples (5 minutes) was selected. Using the
% moving average offers some noise reduction but also decreases the
% resolution of the data so care must be taken in not using a larger window
% than necessary. Note that I've also made the measured movement the
% principal data type. This means that all data should be zero if both
% nodes in a room detect zero movement.
close all

CO22264_2011MA=movmean(CO22264_2011(:,1),10);
perf_co2=0;
for i=1:length(CO22264_2011)
  if movement2249_2011(i,1)==0 && movement2264_2011(i,1)==0
         CO22264_2011MA(i)=0;
  end
end
Occ_pred2309=zeros(length(CO22264_2011MA),1);
for i=1:length(CO22264_2011MA)
    if CO22264_2011MA(i)==0
         Occ_pred2309(i)=0;
    elseif CO22264_2011MA(i)<780
         Occ_pred2309(i,1)=round(b_CO2_1*CO22264_2011MA(i),0);
             Occ_pred2309(i,1)=1;
    elseif CO22264_2011MA(i)<835
         Occ_pred2309(i,1)=round(b_CO2_2*CO22264_2011MA(i),0);
             Occ_pred2309(i,1)=2;
    else
         Occ_pred2309(i,1)=round(b_CO2_3*CO22264_2011MA(i),0);
             Occ_pred2309(i,1)=3;
    end
```

```
    if Occ_pred2309(i)==occupancy2309_2011(i,2)
        perf_co2=perf_co2+1;
    end
end
figure
plot(Occ_pred2309,'r','linewidth',2)
hold on
plot(occupancy2309_2011(:,2),'k','linewidth',2)
title('Occupancy prediction for room 2 on 20-11 based on CO2')
legend('predicted occupancy','actual occupancy')
ylabel('number of people')

100*perf_co2/i
% This early test with the results from the linear regression on the CO2
% alone already showed some promissing results. At this point no
% optimisation is utilised to improve the results, this method's sole
% purpose was to show that there is indeed a clear correlation between
% the measured data and occupancy. Using linear regression in combination
% with an optimisation method resembles what a ANN would do, which is why
% that should be the thing to look into. First I'll do a linear regression
% on the movement data to see if that yields similar results.

%% Linear Regression on movement data.
%close all
movmeanmov3256_1=movmean(mov3256_1811_1,5);
movmeanmov3256_2=movmean(mov3256_1811_2,5);
movmeanmov3256_3=movmean(mov3256_1811_3,5);
movmeanmov3263_1=movmean(mov3263_1811_1,5);
movmeanmov3263_2=movmean(mov3263_1811_2,5);
movmeanmov3263_3=movmean(mov3263_1811_3,5);
% occupancy 1
X1=movmeanmov3256_1;
X2=movmeanmov3263_1;
y1=occ_1811_1(1:length(X1));
y2=occ_1811_1(1:length(X2));
b1_mov_1=inv((X1'*X1))*X1'*y1;
b2_mov_1=inv((X2'*X2))*X2'*y2;

% occupancy 2
X1=movmeanmov3256_2;
X2=movmeanmov3263_2;
y1=occ_1811_2(1:length(X1));
y2=occ_1811_2(1:length(X2));
b1_mov_2=inv((X1'*X1))*X1'*y1;
b2_mov_2=inv((X2'*X2))*X2'*y2;

% occupancy 3
X1=movmeanmov3256_3;
X2=movmeanmov3263_3;
y1=occ_1811_3(1:length(X1));
y2=occ_1811_3(1:length(X2));
b1_mov_3=inv((X1'*X1))*X1'*y1;
b2_mov_3=inv((X2'*X2))*X2'*y2;


% Looking at the calculated regression coefficients data from both nodes
% seem to produce similar values. Similar to the CO2 data the actual values
% of the regression coefficient for different occupancies are sufficiently
% different to say that they may yield similar results as the CO2 data.

%% Testing on validation data using movement data
close all
perf_mov=0;
mov2264_2011MA=movmean(movement2264_2011(:,1),10);
for i=1:length(movement2264_2011)
   if movement2249_2011(i,1)==0 && movement2264_2011(i,1)==0
          mov2264_2011MA(i)=0;
   end
end
Occ_pred2309=zeros(length(mov2264_2011MA),1);
for i=1:length(mov2264_2011MA)
```

```
    if mov2264_2011MA(i)==0
        Occ_pred2309(i)=0;
    elseif mov2264_2011MA(i)<(mean(movmeanmov3256_1)+2)
        Occ_pred2309(i,1)=round(b2_mov_1*mov2264_2011MA(i),0);
    elseif mov2264_2011MA(i)<(mean(movmeanmov3256_2)+2)
        Occ_pred2309(i,1)=round(b2_mov_2*mov2264_2011MA(i),0);
    else
        Occ_pred2309(i,1)=round(b2_mov_3*mov2264_2011MA(i),0);
    end
    if Occ_pred2309(i)==occupancy2309_2011(i,2)
        perf_mov=perf_mov+1;
    end
end
figure
plot(Occ_pred2309,'r','linewidth',2)
hold on
plot(occupancy2309_2011(:,2),'k','linewidth',2)
title('Occupancy prediction for room 2 on 20-11 based on movement')
legend('predicted occupancy','actual occupancy')
ylabel('number of people')

100*perf_mov/i

% The results are less promising which may be due to the limitations of
% the data used for regression. Try using the data from room 1 on 20-11 for
% regression as that data has values for occupancy of 4 and 5 people while
% the data that is currently used caps off at an occupancy of 3 people.
% This would yield additional regression coefficients that may improve the
% results in the higher end. An additional reason the movement data proved
% less successful may be that the average values for different occupancies
% where less separated. This makes it hard to set a clear boundary and
% makes it easier to miss the mark. Next step: combine results.

%% Linear regression using data from 20-11
%close all
movmeanmov3256_1=movmean(mov3256_2011_1,5);
movmeanmov3256_2=movmean(mov3256_2011_2,5);
movmeanmov3256_3=movmean(mov3256_2011_3,5);
movmeanmov3256_4=movmean(mov3256_2011_4,5);
movmeanmov3256_5=movmean(mov3256_2011_5,5);
movmeanmov3263_1=movmean(mov3263_2011_1,5);
movmeanmov3263_2=movmean(mov3263_2011_2,5);
movmeanmov3263_3=movmean(mov3263_2011_3,5);
movmeanmov3263_4=movmean(mov3263_2011_4,5);
movmeanmov3263_5=movmean(mov3263_2011_5,5);
% occupancy 1
X1=movmeanmov3256_1;
X2=movmeanmov3263_1;
y1=occ3309_2011_1(1:length(X1));
y2=occ3309_2011_1(1:length(X2));
b1_mov_1=inv((X1'*X1))*X1'*y1
b2_mov_1=inv((X2'*X2))*X2'*y2

figure
plot(b1_mov_1*mov3256_2011_1(:,1),'g')
hold on
plot(b1_mov_1*X1,'b.-')
hold on
plot(b2_mov_1*mov3263_2011_1(:,1),'y')
hold on
plot(b2_mov_1*X2,'m.-')
hold on
plot(occ3309_2011_1,'k','linewidth', 2)
title('Regression coefficients on moving average compared with raw movement data with ...
    occupancy of 1')
legend('regressed raw 3256','regressed moving average 3256','regressed raw ...
    3263','regressed moving average 3263','occupancy')
ylabel('number of people')

% occupancy 2
X1=movmeanmov3256_2;
```

```
X2=movmeanmov3263_2;
y1=occ3309_2011_2(1:length(X1));
y2=occ3309_2011_2(1:length(X2));
b1_mov_2=inv((X1'*X1))*X1'*y1
b2_mov_2=inv((X2'*X2))*X2'*y2

figure
plot(b1_mov_2*mov3256_2011_2(:,1),'g')
hold on
plot(b1_mov_2*X1,'b.-')
hold on
plot(b2_mov_2*mov3263_2011_2(:,1),'y')
hold on
plot(b2_mov_2*X2,'m.-')
hold on
plot(occ3309_2011_2,'k','linewidth', 2)
title('Regression coefficients on moving average compared with raw movement data with ...
    occupancy of 2')
legend('regressed raw 3256','regressed moving average 3256','regressed raw ...
    3263','regressed moving average 3263','occupancy')
ylabel('number of people')

% occupancy 3
X1=movmeanmov3256_3;
X2=movmeanmov3263_3;
y1=occ3309_2011_3(1:length(X1));
y2=occ3309_2011_3(1:length(X2));
b1_mov_3=inv((X1'*X1))*X1'*y1
b2_mov_3=inv((X2'*X2))*X2'*y2

figure
plot(b1_mov_3*mov3256_2011_3(:,1),'g')
hold on
plot(b1_mov_3*X1,'b.-')
hold on
plot(b2_mov_3*mov3263_2011_3(:,1),'y')
hold on
plot(b2_mov_3*X2,'m.-')
hold on
plot(occ3309_2011_3,'k','linewidth', 2)
title('Regression coefficients on moving average compared with raw movement data with ...
    occupancy of 3')
legend('regressed raw 3256','regressed moving average 3256','regressed raw ...
    3263','regressed moving average 3263','occupancy')
ylabel('number of people')

% occupancy 4
X1=movmeanmov3256_4;
X2=movmeanmov3263_4;
y1=occ3309_2011_4(1:length(X1));
y2=occ3309_2011_4(1:length(X2));
b1_mov_4=inv((X1'*X1))*X1'*y1
b2_mov_4=inv((X2'*X2))*X2'*y2

figure
plot(b1_mov_4*mov3256_2011_4(:,1),'g')
hold on
plot(b1_mov_4*X1,'b.-')
hold on
plot(b2_mov_4*mov3263_2011_4(:,1),'y')
hold on
plot(b2_mov_4*X2,'m.-')
hold on
plot(occ3309_2011_4,'k','linewidth', 2)
title('Regression coefficients on moving average compared with raw movement data with ...
    occupancy of 3')
legend('regressed raw 3256','regressed moving average 3256','regressed raw ...
    3263','regressed moving average 3263','occupancy')
ylabel('number of people')

% occupancy 5
```

```
X1=movmeanmov3256_5;
X2=movmeanmov3263_5;
y1=occ3309_2011_5(1:length(X1));
y2=occ3309_2011_5(1:length(X2));
b1_mov_5=inv((X1'*X1))*X1'*y1
b2_mov_5=inv((X2'*X2))*X2'*y2

figure
plot(b1_mov_5*mov3256_2011_5(:,1),'g')
hold on
plot(b1_mov_5*X1,'b.-')
hold on
plot(b2_mov_5*mov3263_2011_5(:,1),'y')
hold on
plot(b2_mov_5*X2,'m.-')
hold on
plot(y1,'k','linewidth', 2)
title('Regression coefficients on moving average compared with raw movement data with ...
    occupancy of 3')
legend('regressed raw 3256','regressed moving average 3256','regressed raw ...
    3263','regressed moving average 3263','occupancy')
ylabel('number of people')

%% Testing on other data
%close all
mov3256_2011MA=movmean(movement3256_2011(:,1),5);
for i=1:length(movement3256_2011)
  if movement3256_2011(i,1)==0 && movement3256_2011(i,1)==0
      mov3256_2011MA(i)=0;
  end
end
Occ_pred3309=zeros(length(mov3256_2011MA),1);
for i=1:length(mov3256_2011MA)
    if mov3256_2011MA(i)==0
        Occ_pred3309(i)=0;
        elseif mov3256_2011MA(i)<(mean(movmeanmov3256_1)+2)
        Occ_pred3309(i,1)=round(b2_mov_1*mov3256_2011MA(i),0);
    elseif mov3256_2011MA(i)<(mean(movmeanmov3256_2)+2)
        Occ_pred3309(i,1)=round(b2_mov_2*mov3256_2011MA(i),0);
    else
        Occ_pred3309(i,1)=round(b2_mov_3*mov3256_2011MA(i),0);
    end
end
figure
plot(Occ_pred3309,'r','linewidth',2)
hold on
plot(occupancy3309_2011(:,2),'k','linewidth',2)
title('Occupancy prediction for room 1 on 20-11 based on movement')
legend('predicted occupancy','actual occupancy')
ylabel('number of people')

mov2264_2011MA=movmean(movement2264_2011(:,1),10);
for i=1:length(movement2264_2011)
  if movement2249_2011(i,1)==0 && movement2264_2011(i,1)==0
        mov2264_2011MA(i)=0;
  end
end
Occ_pred2309=zeros(length(mov2264_2011MA),1);
for i=1:length(mov2264_2011MA)
    if mov2264_2011MA(i)==0
        Occ_pred2309(i)=0;
    elseif mov2264_2011MA(i)<(mean(movmeanmov3256_1)+2)
        Occ_pred2309(i,1)=round(b2_mov_1*mov2264_2011MA(i),0);
    elseif mov2264_2011MA(i)<(mean(movmeanmov3256_2)+2)
        Occ_pred2309(i,1)=round(b2_mov_2*mov2264_2011MA(i),0);
    else
        Occ_pred2309(i,1)=round(b2_mov_3*mov2264_2011MA(i),0);
    end
end
figure
plot(Occ_pred2309,'r','linewidth',2)
```

```
hold on
plot(occupancy2309_2011(:,2),'k','linewidth',2)
title('Occupancy prediction for room 2 on 20-11 based on movement')
legend('predicted occupancy','actual occupancy')
ylabel('number of people')

% Results did not show any improvement using data from a more varried day.
% A possible reason is that the added variation yields regression
% coefficients for a higher number of people as well as the ones that the
% other data also yielded, but the additional ones were not used due to a
% lower occupancy on the days of the test data. Also, since the variation
% on the current set is higher, less samples are used to generate the
% regression coefficients which me cause them to be less accurate. Next
% I'll try an optimisation step to both combine and optimise the
% coefficients for bith movement and CO2.

%% Simulink
% The regression coeffcicient will act as initial values for the parameters
% that need to be optimised. The challenge now becomes defining a suitable
% cost function in terms of the available data and selecting an appropriate
% optimisation algorithm. So what do we want? The output of the
% optimisation needs to match the observed number of people as closely as
% possible. The cost function can therefore look like minimising the
% squared error of the observations minus the output: E=min   (O-Y)^2. The
% observations are already nicely captured in the "occupancy" variables in
% the workspace. The output needs to be expressed in terms of the inputs
% and the regression coefficients. Since the regression coefficients are
% not based on the raw inputs but rather on the occupancy-corrected inputs
% (so the data corresponding to a constant occupancy value), the way to do
% this is not immediately evident. The output will eventualy look like
% Y=a_1*M+a_2*C with a_i the combined regression coefficient matrices, M the
% movement data matrix and C the CO2 data matrix. Since the observations
% cover 2 days and 2 rooms (1 on the first day and 2 on the second) and
% these vectors are of different length, it makes sense to split the
% problem up in two parts: one for the first day and one for the second.
% This means the output on the first day will be one-dimensional whereas
% the output on the second day will be two dimensional. The movement data
% for each room is generated by 2 sensor nodes per room and will always have
% dimension nx2m (where m is the dimensionality of the output and n is the
% number of samples). the CO2 data is produced by a single sensor node per
% room and will have the same size as the output. The equation for the
% first day would look like Y=c_1*M*a_1+c_2*C*a_2, with Y [nx1], a_1[2x1], M[nx2],
% a_2[1x1], C[nx1] and c_i a constant that determines the priority of each
% data type and follow sum(c_i)=1. The results show that CO2 is more
% reliable and will therefore have a higher priority. However, we still
% need to take into account that we have multiple regression coefficients
% for each data type to choose from. How do we select the most suitable
% regression coefficient for a specific input (or series of inputs if we're
% working with a moving average)? My first thought is to design a switched
% system in Simulink.
```

Data management function

```
function manageddata=datamanagement(data)
% This function takes raw data that contains missing samples as an input,
% identifies and fills the gaps with interpolated values.
i=1;
    while i<(length(data)-1)
        if ((data((i+1),4)-data(i,4))>0.00085) && ((data((i+1),4)-data(i,4))<0.0015)
            data=[data(1:i,:); [NaN,NaN,NaN,NaN]; data(i+1:end,:)];
            i=i+2;
        elseif (data((i+1),4)-data(i,4)>0.0017) && (data((i+1),4)-data(i,4)<0.0025)
            data=[data(1:i,:); [NaN,NaN,NaN,NaN]; data((i+1):end,:)];
            data=[data(1:(i+1),:); [NaN,NaN,NaN,NaN]; data((i+2):end,:)];
            i=i+3;
        elseif (data((i+1),4)-data(i,4)>0.0025) && (data((i+1),4)-data(i,4)<0.0032)
            data=[data(1:i,:); [NaN,NaN,NaN,NaN]; data((i+1):end,:)];
            data=[data(1:(i+1),:); [NaN,NaN,NaN,NaN]; data((i+2):end,:)];
            data=[data(1:(i+2),:); [NaN,NaN,NaN,NaN]; data((i+3):end,:)];
            i=i+4;
        elseif (data((i+1),4)-data(i,4)>0.0032) && (data((i+1),4)-data(i,4)<0.004)
            data=[data(1:i,:); [NaN,NaN,NaN,NaN]; data((i+1):end,:)];
            data=[data(1:(i+1),:); [NaN,NaN,NaN,NaN]; data((i+2):end,:)];
            data=[data(1:(i+2),:); [NaN,NaN,NaN,NaN]; data((i+3):end,:)];
            data=[data(1:(i+3),:); [NaN,NaN,NaN,NaN]; data((i+4):end,:)];
            i=i+5;
        elseif (data((i+1),4)-data(i,4)>0.004) && (data((i+1),4)-data(i,4)<0.0046)
            data=[data(1:i,:); [NaN,NaN,NaN,NaN]; data((i+1):end,:)];
            data=[data(1:(i+1),:); [NaN,NaN,NaN,NaN]; data((i+2):end,:)];
            data=[data(1:(i+2),:); [NaN,NaN,NaN,NaN]; data((i+3):end,:)];
            data=[data(1:(i+3),:); [NaN,NaN,NaN,NaN]; data((i+4):end,:)];
            data=[data(1:(i+4),:); [NaN,NaN,NaN,NaN]; data((i+5):end,:)];
            i=i+6;
        else
            i=i+1;
        end
    end
manageddata(:,1:3) = fillmissing(data(:,1:3),'movmedian',10);
manageddata(:,4) = fillmissing(data(:,4),'linear');

figure
hold on
plot(manageddata(:,3),'r','linewidth',1.5)
plot(data(:,3),'b','linewidth',1.5)
legend('Interpolated data','Raw data')
end
```

Neural network training file

```
% This is the main file for the design and training of a three layer neural
% network with a mix of hyperbolic tangent and Leaky ReLu activation
% functions, trained through back propagation. The function DeepLearning.m
% contains the untrained network. This file uses the untrained network to
% derive the best performing weights and biases which combined with the
% function NeuralNetwork form the final network.
clc
clear all
close all hidden
run('datamanagement_addon.m')
% day 1 room 1
CO23263_1811MA=(movmean(CO23263_1811(:,1),5));
mov3256_1811MA=(movmean(movement3256_1811(:,1),5));
mov3263_1811MA=(movmean(movement3263_1811(:,1),5));
temp3256_1811MA=(movmean(temperature3256_1811(:,1),5));
temp3263_1811MA=(movmean(temperature3263_1811(:,1),5));
hum3256_1811MA=(movmean(humidity3256_1811(:,1),5));
hum3263_1811MA=(movmean(humidity3263_1811(:,1),5));

% day 2 room 1
CO23263_2011MA=(movmean(CO23263_2011(:,1),5));
```

```
mov3256_2011MA=(movmean(movement3256_2011(:,1),5));
mov3263_2011MA=(movmean(movement3263_2011(:,1),5));
temp3256_2011MA=(movmean(temperature3256_2011(:,1),5));
temp3263_2011MA=(movmean(temperature3263_2011(:,1),5));
hum3256_2011MA=(movmean(humidity3256_2011(:,1),5));
hum3263_2011MA=(movmean(humidity3263_2011(:,1),5));
% day 2 room 2
CO22264_2011MA=(movmean(CO22264_2011(:,1),5));
mov2249_2011MA=(movmean(movement2249_2011(:,1),5));
mov2264_2011MA=(movmean(movement2264_2011(:,1),5));
temp2249_2011MA=(movmean(temperature2249_2011(:,1),5));
temp2264_2011MA=(movmean(temperature2264_2011(:,1),5));
hum2249_2011MA=(movmean(humidity2249_2011(:,1),5));
hum2264_2011MA=(movmean(humidity2264_2011(:,1),5));

for i=1:length(movement3256_1811)
    if movement3256_1811(i,1)==0 && movement3263_1811(i,1)==0
        CO23263_1811MA(i)=0;
        mov3256_1811MA(i)=0;
        mov3263_1811MA(i)=0;
        temp3256_1811MA(i)=0;
        temp3263_1811MA(i)=0;
        hum3256_1811MA(i)=0;
        hum3263_1811MA(i)=0;
    end
end

for i=1:length(movement3256_2011)
    if movement3256_2011(i,1)==0 && movement3263_2011(i,1)==0
        CO23263_2011MA(i)=0;
        mov3256_2011MA(i)=0;
        mov3263_2011MA(i)=0;
        temp3256_2011MA(i)=0;
        temp3263_2011MA(i)=0;
        hum3256_2011MA(i)=0;
        hum3263_2011MA(i)=0;
    end
end

for i=1:length(movement2249_2011)
    if movement2249_2011(i,1)==0 && movement2264_2011(i,1)==0
        CO22264_2011MA(i)=0;
        mov2249_2011MA(i)=0;
        mov2264_2011MA(i)=0;
        temp2249_2011MA(i)=0;
        temp2264_2011MA(i)=0;
        hum2249_2011MA(i)=0;
        hum2264_2011MA(i)=0;
    end
end
%% Data normalisation
% For a better performaning neural network, normalisation of the input is
% critical. This is done using MIN-MAX scaling.
% day 1 room 1
for i=1:length(CO23263_1811MA)
    CO23263_1811S(i,1)=(CO23263_1811MA(i)-min(CO23263_1811MA))/ ...
        (max(CO23263_1811MA)-min(CO23263_1811MA));
    mov3256_1811S(i,1)=(mov3256_1811MA(i)-min(mov3256_1811MA))/
    (max(mov3256_1811MA)-min(mov3256_1811MA));
    mov3263_1811S(i,1)=(mov3263_1811MA(i)-min(mov3263_1811MA))/
    (max(mov3263_1811MA)-min(CO23263_1811MA));
    temp3256_1811S(i,1)=(temp3256_1811MA(i)-min(temp3256_1811MA))/
    (max(temp3256_1811MA)-min(temp3256_1811MA));
    temp3263_1811S(i,1)=(temp3263_1811MA(i)-min(temp3263_1811MA))/
    (max(temp3263_1811MA)-min(CO23263_1811MA));
    hum3256_1811S(i,1)=(hum3256_1811MA(i)-min(hum3256_1811MA))/
    (max(hum3256_1811MA)-min(hum3256_1811MA));
    hum3263_1811S(i,1)=(hum3263_1811MA(i)-min(hum3263_1811MA))/
    (max(hum3263_1811MA)-min(CO23263_1811MA));

    occ_1811S(i,1)=(occupancy_1811(i,2)-min(occupancy_1811(:,2)))/
```

```
    (max(occupancy3309_2011(:,2))-min(occupancy_1811(:,2)));
end

% day 2 room 1
for i=1:length(CO23263_2011MA)
    CO23263_2011S(i,1)=(CO23263_2011MA(i)-min(CO23263_2011MA))/
    (max(CO23263_2011MA)-min(CO23263_2011MA));
    mov3256_2011S(i,1)=(mov3256_2011MA(i)-min(mov3256_2011MA))/
    (max(mov3256_2011MA)-min(mov3256_2011MA));
    mov3263_2011S(i,1)=(mov3263_2011MA(i)-min(mov3263_2011MA))/
    (max(mov3263_2011MA)-min(mov3263_2011MA));
    temp3256_2011S(i,1)=(temp3256_2011MA(i)-min(temp3256_2011MA))/
    (max(temp3256_2011MA)-min(temp3256_2011MA));
    temp3263_2011S(i,1)=(temp3263_2011MA(i)-min(temp3263_2011MA))/
    (max(temp3263_2011MA)-min(CO23263_2011MA));
    hum3256_2011S(i,1)=(hum3256_2011MA(i)-min(hum3256_2011MA))/
    (max(hum3256_2011MA)-min(hum3256_2011MA));
    hum3263_2011S(i,1)=(hum3263_2011MA(i)-min(hum3263_2011MA))/
    (max(hum3263_2011MA)-min(CO23263_2011MA));

    occ3309_2011S(i,1)=(occupancy3309_2011(i,2)-min(occupancy3309_2011(:,2)))/
    (max(occupancy3309_2011(:,2))-min(occupancy3309_2011(:,2)));
end

% day 2 room 2
for i=1:length(CO22264_2011MA)
    CO22264_2011S(i,1)=(CO22264_2011MA(i)-min(CO22264_2011MA))/
    (max(CO22264_2011MA)-min(CO22264_2011MA));
    mov2249_2011S(i,1)=(mov2249_2011MA(i)-min(mov2249_2011MA))/
    (max(mov2249_2011MA)-min(mov2249_2011MA));
    mov2264_2011S(i,1)=(mov2264_2011MA(i)-min(mov2264_2011MA))/
    (max(mov2264_2011MA)-min(mov2264_2011MA));
    temp2249_2011S(i,1)=(temp2249_2011MA(i)-min(temp2249_2011MA))/
    (max(temp2249_2011MA)-min(temp2249_2011MA));
    temp2264_2011S(i,1)=(temp2264_2011MA(i)-min(temp2264_2011MA))/
    (max(temp2264_2011MA)-min(CO22264_2011MA));
    hum2249_2011S(i,1)=(hum2249_2011MA(i)-min(hum2249_2011MA))/
    (max(hum2249_2011MA)-min(hum2249_2011MA));
    hum2264_2011S(i,1)=(hum2264_2011MA(i)-min(hum2264_2011MA))/
    (max(hum2264_2011MA)-min(CO22264_2011MA));

    occ2309_2011S(i,1)=(occupancy2309_2011(i,2)-min(occupancy2309_2011(:,2)))/
    (max(occupancy3309_2011(:,2))-min(occupancy2309_2011(:,2)));
end

%% Define input and target vectors
close all
Xnn=[CO23263_1811S mov3256_1811S mov3263_1811S temp3256_1811S temp3263_1811S ...
    hum3256_1811S hum3263_1811S; CO23263_2011S mov3256_2011S mov3263_2011S ...
    temp3256_2011S temp3263_2011S hum3256_2011S hum3263_2011S];
% target matrix
Ynn=[5*occ_1811S; 5*occ3309_2011S];
Xval=[CO22264_2011S mov2249_2011S mov2264_2011S temp2249_2011S temp2264_2011S ...
    hum2249_2011S hum2264_2011S];
Yval=5*occ2309_2011S;
save('optim.mat');
alpha=0.1;
%%
close all
maxit=500;
p=0.01;
[w1_best, w2_best, w3_best, w4_best, b1_best, b2_best, b3_best, b4_best, perf_final, ...
    maxperf, ep, perf_train]=TrainingNN(Xnn, Ynn, 0.00001, p, maxit, 5);
save('DeepNN.mat')
% Starting with a learning rate of 0.1 and using a rule that incremently
% adjusts the learning rate based on the performance, the final learning
% rate that reached the highest performance was 0.02. This yielded a
% training performance of 69.7% (53.1% when the training data was used as
% validation) and a validation performance of 75.2%. The system showed
% the capability of reaching occupancy predictions of 0, 1, 2, 3 and 5
% people. the fact that an occupancy of 4 people was never predicted may be
```

```
% due to the fact that an occupancy of 4 people was hardly ever observed.

%% optimisation
% Here the Genetic Algorithm is used to find the optimal values for the
% learning rate and number of neurons per layer. The GA usually did not
% converge and was often stopped prematurely when the results no longer
% showed improvements.

% best values found by the GA
% 0.0001    12.0000    12.0000    59.0000
% 62.2%, 63.9%
% 0.0002    19.0000    21.0000    39.0000
% 62.0%
% 0.0001    86.0000     8.0000    10.0000
% 63.3%

maxit=100;
FitnessFnc= @TrainingNN_opt;
% A = [1 1];                          % Constraints
% B = 150; % Inequality constraints
% Aeq = [1 1/1000];   % Equality constraints
% Beq = 0.01+25/1000;
LB = [10e-6, 5, 5, 5];
UB = [0.1, 100, 100, 100];
IntCon=[2, 3, 4];
options = gaoptimset('UseParallel', true,...
                     'Vectorized', 'off','FitnessLimit',0.35,...
                     'EliteCount', 2, ...
                     'PlotFcns', {@gaplotbestf,@gaplotstopping});
opts = optimoptions('ga','PlotFcn',{@gaplotbestf,@gaplotstopping}, 'FitnessLimit', 0.35);
[x, fval, exitflag] = ga(FitnessFnc, 4,[],[],[],[],...
    LB,UB,[],IntCon,options)
% I'm trying different combinations of activation functions. Using tanh
% throughout and leaky ReLu for the output resulted in performance of ~65%.
% Using tanh on every layer except the 2nd for which I use leaky ReLu
% resulted in performance of ... well, Matlab crashed. Performance didn't
% look impressive though with the maximum around 55%.
%%
close all
figure
label1 = {string(maxperf)};
plot(perf_final,'b')
hold on
plot(ep,maxperf,'ko','markerfacecolor','r')
text(ep,maxperf,label1,'VerticalAlignment','bottom','HorizontalAlignment','right')
title('Training performance')
axis([0 maxit 0 1])
legend('Performance','Maximum performance','Location','northwest')
save('DeepNN.mat')
% Using tanh improved training results significantly as well as making the
% training process more consistent. This is probably due to the tanh
% superior range and responsiveness to negative inputs while also keeping
% inputs of zero at zero. The NN is still unable to reach all levels of
% occupancy. It's able to reach 2 and 0 but not much else. Try optimizing
% over the number of hidden neurons and the learning rate
%%
run('TestDeepLearning.m')
```

Neural network testing file

```
% This file tests the final network using the validation data and the data
% used for training.
% load('optim');
close all
load('DeepNN_best_final.mat');

[Ytraining]=NeuralNetwork(w1_best, w2_best, w3_best, w4_best, b1_best, b2_best, ...
    b3_best, b4_best, Xnn);

Xval=[CO22264_2011S mov2249_2011S mov2264_2011S temp2249_2011S temp2264_2011S ...
```

```
    hum2249_2011S hum2264_2011S];
Yval=5*occ2309_2011S;
[Yresult]=NeuralNetwork(w1_best, w2_best, w3_best, w4_best, b1_best, b2_best, b3_best, ...
    b4_best, Xval);
Yhot=zeros(length(Ynn),6);
    for i=1:length(Yhot)
        Yhot(i,Ynn(i)+1)=1;
    end
[¬, ¬, ¬, ¬, ¬, ¬, ¬, ¬, ¬, out]=DeepLearning(w1_best, w2_best, w3_best, w4_best, ...
    b1_best, b2_best, b3_best, b4_best, Xnn, Yhot, alpha, p);

%%
% close all hidden
% Ytraining=[Ytraining(:,1:2) Ytraining(:,4:6)];
[¬,idx]=max(Ytraining,[],2);
correction=ones(length(idx),1);
Y_train=idx-correction;
figure
plot(Ynn)
hold on
plot(Y_train)
title('Training output versus target')
ylabel('Number of people')
legend('Training target','Network output')
perft=0;
for i=1:length(idx)
    if Ynn(i)==Y_train(i)
        perft=perft+1;
    end
end
perf_train=perft/i

perft_1=0;
for i=1:length(idx)
    if Y_train(i)==(Ynn(i)+1)
        perft_1=perft_1+1;
    elseif Y_train(i)==(Ynn(i)-1)
        perft_1=perft_1+1;
    end
end
perft1=perft_1/i

perft_more=0;
for i=1:length(idx)
    if Y_train(i)>(Ynn(i)+1)
        perft_more=perft_more+1;
    elseif Y_train(i)<(Ynn(i)-1)
        perft_more=perft_more+1;
    end
end
perftmore=perft_more/i
RMSError_tr=sqrt(mean((Y_train-Ynn).^2))
%%
% Yresult=[Yresult(:,1:2) Yresult(:,4:6)];
[¬,idx]=max(Yresult,[],2);
correction=ones(length(idx),1);
Y_val=idx-correction;

figure
plot(Yval)
hold on
plot(Y_val)
title('Output using validation data versus validation target')
ylabel('Number of people')
legend('Validation target','Network output')
perfv=0;
for i=1:length(idx)
    if Yval(i)==Y_val(i)
        perfv=perfv+1;
    end
end
```

```
perf_val=perfv/i

perf_1=0;
for i=1:length(idx)
    if Y_val(i)==(Yval(i)+1)
        perf_1=perf_1+1;
    elseif Y_val(i)==(Yval(i)-1)
        perf_1=perf_1+1;
    end
end
perf1=perf_1/i

perf_more=0;
for i=1:length(idx)
    if Y_val(i)>(Yval(i)+1)
        perf_more=perf_more+1;
    elseif Y_val(i)<(Yval(i)-1)
        perf_more=perf_more+1;
    end
end
perfmore=perf_more/i

RMSError=sqrt(mean((Y_val-Yval).^2))
```

Untrained network function

```
function [w1, w2, w3, w4, b1, b2, b3, b4, perf, out]=DeepLearning(w1, w2, w3, w4, b1, ...
    b2, b3, b4, X, Y, alpha, p)
% This function trains a three layer neural network with a mix of
% hyperbolic tangent and Leaky ReLu activation functions using back
% propagation. The number of neurons per layer is determined by the
% inputs wi and bi. The ouput will be a one-hot encoded matrix of [nx6],
% with the columns representing occupancy values ranging from 0 (column 1)
% to 5 (column 6). The target matrix Y also needs to be a one-hot encoded
% matrix.
perf_init=0;
out=zeros(size(Y));
SZ=size(w1*X(1,:)');
    for i=1:length(X)
        xi=X(i,:)';
        y=Y(i,:)';

            A=ones(size(w1*xi));
            input_of_HL1=A.*(w1*xi+ones(size(w1*xi)).*b1);
            output_of_HL1=tanh(input_of_HL1);

            B=ones(size(w2*output_of_HL1));
            input_of_HL2=B.*(w2*output_of_HL1+ones(size(w2*output_of_HL1)).*b2);
            output_of_HL2=tanh(input_of_HL2);

            C=ones(size(w3*output_of_HL2));
            input_of_HL3=C.*(w3*output_of_HL2+ones(size(w3*output_of_HL2)).*b3);
            output_of_HL3=tanh(input_of_HL3);

            D=ones(size(w4*output_of_HL3));
            input_of_outputnode=D.*(w4*output_of_HL3+ones(size(w4*output_of_HL3)).*b4);
            final_output=tanh(input_of_outputnode);
            out(i,:)=final_output;

            error=(y-final_output);
            [¬,idx1]=max(y,[],1);
            [¬,idx2]=max(final_output,[],1);
            if idx1==idx2
                perf_init=perf_init+1;
            end

            ∆ = error;

            error_HL3= w4'*∆;
            ∆3=(input_of_HL3>0).*error_HL3;
```

```
                error_HL2= w3'*Δ3;
                Δ2=(input_of_HL2>0).*error_HL2;

                error_HL1= w2'*Δ2;
                Δ1=(input_of_HL1>0).*error_HL1;

                adjustment_w4=alpha*Δ*output_of_HL3';
                adjustment_w3=alpha*Δ3*output_of_HL2';
                adjustment_w2=alpha*Δ2*output_of_HL1';
                adjustment_w1=alpha*Δ1*xi';

                adjustment_b4=alpha*Δ;
                adjustment_b3=alpha*Δ3;
                adjustment_b2=alpha*Δ2;
                adjustment_b1=alpha*Δ1;

                w1=w1+adjustment_w1;
                w2=w2+adjustment_w2;
                w3=w3+adjustment_w3;
                w4=w4+adjustment_w4;

                b1=b1+adjustment_b1;
                b2=b2+adjustment_b2;
                b3=b3+adjustment_b3;
                b4=b4+adjustment_b4;

                out(i,:)=final_output;
        end
        perf=perf_init/length(X);

end
```

Trained network function

```
function [out]=NeuralNetwork(w1, w2, w3, w4, b1, b2, b3, b4, X)
% This function contains a three layer neural network with a mix of
% hyperbolic tangent and Leaky ReLu activation functions. The number of
% neurons per layer is determined by the inputs wi and bi. The ouput will
% be a one-hot encoded matrix of [nx6], with the columns representing
% occupancy values ranging from 0 (column 1) to 5 (column 6).
out=zeros(length(X),6);
        for i=1:length(X)
            x=X(i,:)';

                A=ones(size(w1*x));
                input_of_HL1=A.*(w1*x+ones(size(w1*x)).*b1);
                output_of_HL1=tanh(input_of_HL1);

                B=ones(size(w2*output_of_HL1));
                input_of_HL2=B.*(w2*output_of_HL1+ones(size(w2*output_of_HL1)).*b2);
                output_of_HL2=tanh(input_of_HL2);

                C=ones(size(w3*output_of_HL2));
                input_of_HL3=C.*(w3*output_of_HL2+ones(size(w3*output_of_HL2)).*b3);
                output_of_HL3=tanh(input_of_HL3);

                D=ones(size(w4*output_of_HL3));
                input_of_outputnode=D.*(w4*output_of_HL3+ones(size(w4*output_of_HL3)).*b4);
                final_output=tanh(input_of_outputnode);
                out(i,:)=final_output;
      end
end
```

Neural network training function

```
% This function trains a deep neural network with 3 hidden layers using
% back propagation. If only one number of neurons is specified, all layers
% will have the same number of neurons. If you want different number of
```

```
% neurons per layer, specify all three numbers. Make sure the inputs are
% normalised but the output is not!

% This function outputs the weights and biases that achieved the highest
% training performance, the final training performance, the best training
% performance and the performance of the final neural net when tested with
% the training data.

function [w1_best, w2_best, w3_best, w4_best, b1_best, b2_best, b3_best, b4_best, ...
    training_performance, best_training_performance, epoch_best, ...
    perf_avg_opt]=TrainingNN(X, Y, learning_rate, dropout_rate, ...
    max_iterations,number_neurons_1, number_neurons_2, number_neurons_3, ...
    number_neurons_4)
    if ¬exist('number_neurons_2','var')
     % second parameter does not exist
        number_neurons_2 = number_neurons_1;
    end

    if ¬exist('number_neurons_3','var')
     % third parameter does not exist
        number_neurons_3 = number_neurons_1;
    end

    if ¬exist('number_neurons_4','var')
     % third parameter does not exist
        number_neurons_4 = number_neurons_1;
    end

Yhot=zeros(length(Y),6);
    for i=1:length(Yhot)
        Yhot(i,Y(i)+1)=1;
    end

w1=2*rand(number_neurons_1,size(X,2))-1;
w2=2*rand(number_neurons_2,size(w1,1))-1;
w3=2*rand(number_neurons_3,size(w2,1))-1;
% w4=2*rand(number_neurons_4,size(w3,1))-1;
% w5=2*rand(size(Yhot,2),size(w4,1))-1;
w4=2*rand(size(Yhot,2),size(w3,1))-1;
b1=-rand(1);
b2=-rand(1);
b3=-rand(1);
b4=-rand(1);
% b5=-rand(1);
alpha=learning_rate;                        % Learning rate
p=dropout_rate;                               % Dropout rate for dropout layer
f=waitbar(0);
training_performance=zeros(max_iterations,1);
for i =2:(max_iterations+1)
    [w1, w2, w3, w4, b1, b2, b3, b4, training_performance(i)]=DeepLearning(w1, w2, w3, ...
        w4, b1, b2, b3, b4, X, Yhot, alpha, p);
    waitbar((i-1)/max_iterations,f,'epoch '+string((i-1)))
    [best_training_performance,ep]=max(training_performance);
        w1_best=w1;
        w2_best=w2;
        w3_best=w3;
        w4_best=w4;

        b1_best=b1;
        b2_best=b2;
        b3_best=b3;
        b4_best=b4;
    if training_performance(i-1)==best_training_performance
        w1_best=w1;
        w2_best=w2;
        w3_best=w3;
        w4_best=w4;
%          w5_best=w5;
        b1_best=b1;
        b2_best=b2;
        b3_best=b3;
```

```
            b4_best=b4;
%           b5_best=b5;
        end
%   Check how the network performs on the training data and change the learning rate if ...
        the performance doesn't increase
[Ytraining]=NeuralNetwork(w1_best, w2_best, w3_best, w4_best, b1_best, b2_best, ...
        b3_best, b4_best, X);
[¬,idx]=max(Ytraining,[],2);
correction=ones(length(idx),1);
Y_train=idx-correction;
perft=0;
for j=1:length(idx)
        if Y(j)==Y_train(j)
            perft=perft+1;
        end
end
perf_train(1)=0.3821;
perf_train(i)=perft/j;
perf_avg_opt=1-mean(perf_train);
% if perf_train(i)<0.5
%       if perf_train(i)<perf_train(i-1)
%           alpha=alpha-0.001;
%       else
%           alpha=alpha+0.001;
%       end
% end


        label1 = {string(best_training_performance)};
        label2 = {string(perf_train(i))};
        plot(perf_train,'b');
        title('Training performance')
        axis([0 max_iterations 0 1])
        hold on
        p=plot(ep,best_training_performance,'ko','markerfacecolor','r');
        hold on
        t=text(ep,best_training_performance,label1,'VerticalAlignment',
        'bottom','HorizontalAlignment','right');
        hold on
        s=text(i,perf_train(i-1),label2,'VerticalAlignment','bottom',
        'HorizontalAlignment','right');
        hold on
        drawnow
        delete(p)
        delete(t)
        delete(s)
end
close(f)
close all
figure
plot(training_performance,'b')
hold on
plot(ep,best_training_performance,'ko','markerfacecolor','r')
text(ep,best_training_performance,label1,'VerticalAlignment','bottom',
'HorizontalAlignment','right')
title('Training performance')
axis([0 max_iterations 0 1])
legend('Performance','Maximum performance','Location','northwest')
% test performance on training data

epoch_best=ep;


save('DeepNN.mat')
end
```

Bibliography

[1] Energy Information Administration. “2003 Commercial Buildings Energy Consumption Survey (CBECS)”. In: *End-Use Equipment Tables, Tables B39 and B41* (2006).

[2] Energy Information Administration. “2012 Commercial Buildings Energy Consumption Survey (CBECS)”. In: *End-Use Equipment Tables, Tables B39 and B41* (2015).

[3] M.D. Asaduzzaman and M.D. Shahjahan. “Faster Training using Fusion of Activation Functions for Feed Forward Neural Networks”. In: *International Journal of Neural Systems* 19.6 (2009), pp. 437–448.

[4] ASHRAE. “ANSI/ASHRAE standard 55-2004: Thermal Environmental Conditions for Human Occupancy”. In: *American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Inc.: Atlanta, GA.* (2010).

[5] bGrid. *API documentation, bGrid system version 3.4*. 2019, pp. 1–28.

[6] bGrid. *BACnet documentation, bGrid system version 3.4*. 2018, pp. 1–10.

[7] bGrid. *Functional Specifications, bGrid system version 3*. 2018, pp. 1–19.

[8] bGrid. *Integrated Light Control in bGrid*. 2017, pp. 1–7.

[9] E. Bonabeau. “Agent-based modeling: methods and techniques for simulating human systems”. In: *Proc. Natl. Acad. Sci. U.S.A.* 99 (2002), pp. 7280–7287.

[10] V.M. Alvarado-Martínez C.J. Zuñiga Aguilar J.F. Gómez-Aguilar and H.M. Romero-Ugalde. “Fractional order neural networks for system identification”. In: *Chaos, Solitons and Fractals* 130 (2019).

[11] F. Causone et al. “A data-driven procedure to model occupancy and occupant-related electric load profiles in residential buildings for energy simulation”. In: *Energy and Buildings* 202 (2019), pp. 1–14.

[12] P.E. Christopher Wilkins and Ph.D. M.H. Hosni. “Heat Gain From Office Equipment”. In: *ASHRAE Journal* (2000), pp. 33–39.

[13] Thomas L. Dean and Michael P. Wellman. *Planning and Control*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991. ISBN: 1-55860-209-7.

[14] Richard J. de Dear. “A Global Database of Thermal Comfort Field Experiments”. In: *ASHRAE transactions 104* (1998), pp. 1141–1152.

[15] Richard J. de Dear and Gail S. Brager. “Thermal comfort in naturally ventilated buildings: revisions to ASHRAE Standard 55”. In: *Energy and Buildings by Elsevier* 34 (2002) (2002), pp. 549–561.

[16] Faiyaz Doctor, Hani Hagras, and Victor Callaghan. “A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments”. In: *IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans* 35 (2005) (2005), pp. 55–65.

[17] Gérard Dupont et al. “Multi objective particle swarm optimization using enhanced dominance and guide selection”. In: *International Journal of Computational Intelligence Research* 4 (2) (2008), pp. 145–158.

[18] Diana Enescu. “A Review of Thermal Comfort Models and Indicators for Indoor Environments”. In: *Renewable and Sustainable Energy Reviews by Elsevier* 79 (2017) (2017), pp. 1353–1379.

[19] *Engineering Toolbox*. 2001. URL: https://www.engineeringtoolbox.com (visited on 11/06/2019).

[20] Drew Fudenberg and Jean Tirole. *Game Theory*. 1991. ISBN: 9780262061414.

[21] Arno Schlueter Gabriel Happle Jimeno A. Fonseca. “Context-specific urban occupancy modeling using location-based services data”. In: *Building and Environment* 175 (2020), pp. 1–18.

[22] Geoffrey Hinton et al. “Dropout: a Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958.

[23] E. Halawa and J. van Hoof. “The Adaptive Approach to Thermal Comfort: A Critical Overview”. In: *Energy and Buildings by Elsevier* 51 (2012) (2012), pp. 101–110.

[24] Jan L. M. Hensen. “Literature Review on Thermal Comfort in Transient Conditions”. In: *Building and Environment by Elsevier* 25 (1990) (1990), pp. 309–316.

[25] P. R. Höppe. “Heat balance modelling”. In: *Experientia* 49.9 (1993), pp. 741–746.

[26] Peter. J Hughes and Elvezio M. Ronchetti. *Robust Statistics*. Wiley, 2009. ISBN: 9780470129906.

[27] M.A. Humphreys. “Outdoor Temperatures and Comfort Indoors”. In: *Building Research and Practice (Journal of CIB)* 6 (2) (1978), pp. 92–105.

[28] Farrokh Jazizadeh, Ali Ghahramani, and Burcin Becerik-Gerber. “A knowledge based approach for selecting energy-aware and comfort-driven HVAC temperature set points”. In: *Energy and Buildings by Elsevier* 85 (2014) (2014), pp. 536–548.

[29] Farrokh Jazizadeh et al. “Human-Building Interaction Framework for Personalized Thermal Comfort-Driven Systems in Office Buildings”. In: *Journal of Computing in Civil Engineering (ASCE)* Volume 28, Issue 1 (2014).

[30] Farrokh Jazizadeh et al. “User-led decentralized thermal comfort driven HVAC operations for improved efficiency in office buildings”. In: *Energy and Buildings by Elsevier* 70 (2014) (2013), pp. 398–410.

[31] Henry F. Kaiser. “The Application of Electronic Computers to Factor Analysis”. In: *Educational and Psychological Measurement* 20 (1960), pp. 141–151.

[32] James Kennedy and Russel Eberhart. “Particle Swarm Optimization”. In: *Neural Networks, 1995. Proceedings., IEEE International Conference on* 6 (1995), pp. 1942–1948.

[33] M. B. Kjærgaard et al. “DCount - A Probabilistic Algorithm for Accurately Disaggregating Building Occupant Counts into Room Counts”. In: *2018 19th IEEE International Conference on Mobile Data Management (MDM)*. 2018, pp. 46–55.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems* 25 (2012), pp. 1097–1105.

[35] C. Pout L. Pérez-Lombard J. Ortiz. “A review on buildings energy consumption information”. In: *Energy and Buildings* 40.3 (2008), pp. 394–398.

[36] T. Leach. “Implementing a BACnet Network”. In: *ASHRAE Journal* 59.3 (2017), pp. 40–48.

[37] J. Harris M. Wigginton. “Intelligent Skins”. In: *Architectural Press Oxford* (2002).

[38] Gary A. Montague Mark R. Warnes Jarmila Glassey and Bo Kara. “On Data-Bases Modelling Techniques for Fermentation Processes”. In: *Process Biochemistry* 31.2 (1995), pp. 147–155.

[39] Norm G. Miller. “Downsizing and Workplace Trends in the Office Market”. In: *Real Estate Issues* 38.3 (2013), pp. 28–35.

[40] D. mora et al. “Occupancy patterns obtained by heuristic approaches: Cluster analysis and logical flowcharts. A case study in a university office”. In: *Energy and Buildings* 186 (2019), pp. 147–168.

[41] Michael C. Mozer. “The Neural Network House: An Environment that Adapts to its Inhabitants”. In: *AAAI Technical Report* SS-98-02 (1998), pp. 110–114.

[42] Michael C. Mozer, Lucky Vidmar, and Robert H. Dodier. “The Neurothermostat: Predictive Optimal Control of Residential Heating Systems”. In: *Advances in Neural Information Processing Systems, MIT Press* 9 (1997) (1997).

[43] Vinod Nair and Geoffrey Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines”. In: *Proceedings of ICML* 27 (2010), pp. 807–814.

[44] J.F. Nicol and M.A. Humphreys. “Adaptive Thermal Comfort and Sustainable Thermal Standards for Buildings”. In: *Energy and Buildings by Elsevier* 34 (2002) (2002), pp. 563–572.

[45] P. Siddarth P. Sibi S. Allwyn Jones. “Mass transfer study of water deoxygenation in arotor–stator reactor based on principal component regression method”. In: *Chemical Engineering Research and Design* 132 (2018), pp. 1264–1268.

[46] George Philipp, Dawn Song, and Jaime G. Carbonell. “Gradients explode - Deep Networks are shallow - ResNet explained”. In: *CoRR* abs/1712.05577 (2017).

[47] Y. Rahmat-Samii, Dennis Gies, and Jacob Robinson. “Particle Swarm Optimization (PSO): A Novel Paradigm for Antenna Designs”. In: *the Radio Science Bulletin* 305 (2003), pp. 1–9.

[48] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors”. In: *Nature* 323 (October 1986) (1986), pp. 533–536.

[49] S. Carlucci et al. “Modelling Occupant Behavior in Buildings”. In: *Building and Environment* 174 (2020), pp. 1–24.

[50] A. Weihua Li S. Valle and S.K. Qin. “Selection of the number of principal components: the variance of the reconstruction error criterion with a comparison to other methods”. In: *Industrial Engineering Chemistry Research* 38 (1999), pp. 653–658.

[51] Haşim Sak, Andrew Senior, and Françoise Beaufays. “Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition”. In: *arXiv preprint arXiv:1402.1128* (2014).

[52] Fisayo Caleb Sangoboye and Mikkel Baun Kjærgaard. “PLCount: A Probabilistic Fusion Algorithm for Accurately Estimating Occupancy from 3D Camera Counts”. In: *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 147–156. ISBN: 9781450342643. DOI: 10.1145/2993422.2993575. URL: https://doi.org/10.1145/2993422.2993575.

[53] I. Shafi et al. “Impact of Varying Neurons and Hidden Layers in Neural Network Architecture for a Time Frequency Application”. In: *2006 IEEE International Multitopic Conference* (2006), pp. 188–193.

[54] J. Sjöberg, H. Hjalmarsson, and L. Ljung. “Neural Networks in System Identification”. In: *IFAC Proceedings Volumes* 27.8 (1994), pp. 359–382.

[55] Leslie N. Smith. *A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay*. 2018. arXiv: 1803.09820 [cs.LG].

[56] J. Sola and J. Sevilla. “Importance of input data normalization for the application of neural networks to complex industrial problems”. In: *IEEE Transactions on Nuclear Science* 44.3 (1997), pp. 1464–1468.

[57] J.C. Stevens and K.K. Choo. “Temperature Sensitivity of the Body Surface over the Life Span”. In: *Somatosensory & motor Research* 15 (1998), pp. 13–28.

[58] Sasha Targ, Diogo Almeida, and Kevin Lyman. “Resnet in Resnet: Generalizing Residual Architectures”. In: *ArXiv* abs/1603.08029 (2016).

[59] The World Business Council for Sustainable Development. “Annual Energy Review 2010”. In: *EIA* (2011).

[60] Lingfeng Wang and Rui Yang. “Multi-objective optimization for decision-making of energy and comfort management in building automation and control”. In: *Sustainable Cities and Society by Elsevier* 2 (2012) (2011), pp. 1–7.

[61] Svante Wold. “Cross-Validatory Estimation of the Number of Components in Factor and Principal Components Models”. In: *Technometrics* 20 (1978), pp. 397–405.

[62] Xin Yan and Xiao Gang Su. *Linear Regression Analysis: Theory and Computing*. World Scientific Publishing Co., Inc., 2009. ISBN: 9789812834102.

[63] Rui Yang and Lingfeng Wang. “Development of multi-agent system for building energy and comfort management based on occupant behaviors”. In: *Energy and Buildings by Elsevier* 56 (2012), pp. 1–7.

[64] Rui Yang and Lingfeng Wang. "Multi-zone building energy management using intelligent control and optimization". In: *Sustainable Cities and Society by Elsevier* 6 (2013) (2012), pp. 16–21.

[65] Rui Yang, Zhu Wang, and Lingfeng Wang. "Multi-agent Control System with Intelligent Optimization for Smart and Energy-efficient Buildings". In: *IECON Proceedings (Industrial Electronics Conference)* 12 (2010), pp. 1144–1149.

[66] Sang-Hoon Oh Youngjik Lee and Myung Won Kim. "An analysis of premature saturation in back propagation learning". In: *Neural Networks* 6.5 (1993), pp. 719–728.

[67] Zemeng Zhao et al. "Analysis of Different Activation Functions using Back Propagation Neural Networks". In: *Journal of Theoretical and Applied Information Technology* 47.3 (2013), pp. 677–685.