Norwegian
University of
Life Sciences

Master's Thesis 2025 **30 ECTS**
Faculty of Environmental Sciences and Natural Resource Management

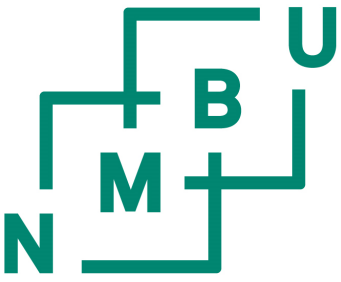
Vehicle-to-grid: Investigating Factors Affecting the Efficiency of Bidirectional Charging

Ylva Ivås

M30 - MASTER THESIS,
RENEWABLE ENERGY

Vehicle-to-grid: Investigating Factors Affecting the Efficiency of Bidirectional Charging

Ylva Ivås

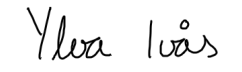


Norwegian University of Life Sciences
Faculty of Environmental Science and Nature Management

Preface

This paper is a master's thesis about V2G written in the spring semester of 2025 by a second-year master's student at the Norwegian University of Life Sciences, NMBU. The thesis is based on data collected by the NeX2G research group from NMBU, where one of the researchers, Thomas Martinsen, was the supervisor. I want to thank Thomas Martinsen and PhD student Muhammad Tabish Parray for answering any questions and dedicating their time to assist me throughout this thesis.

Ås, 15.05.2025



Ylva Ivås

Abstract

With the expected future growth in power consumption, it is essential to maintain the balance between electricity production and consumption within the power grid, given the lack of solutions for large-scale energy storage. Utilising electric vehicles as temporary energy storage, vehicle-to-grid can deliver power to the grid when demand is high, while charging the vehicle during off-peak hours.

Based on the given data, this study aimed to determine how much loss occurs during a round trip by assessing the efficiency of bidirectional charging. The data used in this thesis belongs to the research group NeX2G at NMBU, which is studying whether the batteries in parked electric vehicles can be used as temporary energy storage. The data was collected using the battery management system, where data was retrieved from several Nissan Leaf vehicles. Several cycles of bidirectional charging were performed. Cycles were identified from the data using a function created for this purpose. The cycles varied in state of charge level, how much power the vehicle received or returned to the grid, and temperature. To find the losses in the system, how much energy was sent into the car and returned via discharging was calculated. The loss could then easily be found by determining how much energy was not returned.

The results of the analyses conducted in this study are within expectations. Both patterns and efficiencies were observed, and they have similarities with findings in similar studies. However, the variations in the efficiencies of the different cycles may indicate room for improvement in the function used to identify the cycles. There is also a need for more data from months with very high and low temperatures to determine the extent to which the temperature affects round-trip efficiency.

Sammendrag

Med den forventede fremtidige veksten i kraftforbruk, er det viktig å opprettholde balansen mellom strømproduksjon og forbruk i strømnettet, på grunn av mangelen på gode løsninger for storskala energilagring. Ved bruk av elektriske kjøretøy som midlertidig energilagring kan toveislading levere strøm til nettet når etterspørselen er høy, mens kjøretøyet lades i lavlastperioder.

Oppgaven tok sikte på å bestemme tur-retur energitapet for toveisladning ved å finne virkningsgraden. Dataene som ble brukt i oppgaven tilhører forskergruppen NeX2G fra NMBU, som forsker på om batteriene i parkerte elbiler kan brukes som midlertidig energilager. Data ble samlet inn ved hjelp av kjøretøyenes batteristyringssystem, der data ble hentet fra flere Nissan Leaf biler. Flere sykluser med toveislading ble gjennomført. Utfra dataene ble det identifisert sykluser ved hjelp av en funksjon som ble laget til dette formålet. Syklusene varierte både i ladetilstand, hvor mye effekt bilen fikk tilført eller returnerte, og temperatur. For å finne energitapet ble det kalkulert hvor mye av den energien som ble tilført bilen som ble returnert via utladingen. Tapet kunne da enkelt finnes ved å se på hvor mye energi som ikke kom i retur.

Resultatene av analysene som ble gjennomført i denne studien er i hovedsak som forventet. Det ble observert både mønstre og virkningsgrader som har likheter med funn gjort i lignende studier. Variasjonene i de ulike syklusenes virkningsgrad kan imidlertid tyde på at det er rom for forbedringer i funksjonen som ble benyttet for å identifisere syklusene. Det er også behov for mer data fra måneder med veldig høye og veldig lave temperaturer for å kunne trekke konklusjoner om i hvilken grad temperatur påvirker tur-retur virkningsgrad.

Contents

Glossary

Deintercalation	When ions are removed from the layers in a structure.
Depth of discharge	Defined as how much percentage of a battery has been used.
Electrode	A type of conductor that touches a circuit's non-metallic part.
Intercalation	When ions are inserted between layers in a structure.
Interface	The place where two things, such as computers, electronics or systems, meet/connect.
Memory effect	An effect that occurs in batteries that are constantly charged or discharged less than they are capable of, making the battery remember a capacity that is lower than it originally is, making the battery think it is full or empty earlier than it actually is.
Organic compound	Most compounds that contain carbon and hydrogen tied together. Exceptions include carbon dioxide and, monoxide, carbides and carbon salts.
RFID	A digital recognition system that communicates wirelessly with data systems and microchips placed inside or on objects.
Self-discharge	When a battery loses charge as time passes, even when the battery is not in use.
Thermal runaway	When a battery's temperature rises due to charging or overcharging, resulting in a higher flow of current, which generates more heat and further raises the temperature.

Abbreviations

AC	Alternating current
BEV	Battery electric vehicle
BMS	Battery management system
CC	Constant current
CCS	Combined charging system
CV	Constant voltage
DC	Direct current
DoD	Depth of discharge
DSO	Distribution system operator
EV	Electric vehicle
EVSE	Electric vehicle supply equipment
ISO	International organization for standardization
JEVS	Japan electric vehicle standard
PHEV	Plug-in hybrid electric vehicle
RFID	Radio frequency identification
RTE	Round-trip efficiency
SEI	Solid electrolyte interphase
SOA	Safe operating area
SoC	State of charge
SQL	Structured query language
TEPCO	Tokyo electric power company
TSO	Transmission service operator
V2G	Vehicle to grid

Units

A	Ampere, the measurement for electrical current
Ah	Ampere-hour, measurement for how many amps flow per hour
°C	Celsius, measurement for temperature
J	Joules, an SI unit that measures energy
kV	Kilovolt, a thousand volts
kW	kilowatt, a thousand watts
MW	Megawatt, consists of 10^6 watts
TWh	Terawatt hour, consists of 10^{12} watt hours
Volt	Unit used to measure electrical voltage
Watt	Unit used to measure power

List of Figures

List of Tables

1 Introduction

In the report "Consumption Development in Norway 2022-2050", Statnett estimates that power consumption in Norway in 2050 will be around 220 Terawatt hour, TWh, which is an increase of almost 60% from today's consumption [1]. As there is a lack of large-scale energy storage solutions, ensuring efficient grid balancing is one of the main challenges, one that will become even more challenging as electrification increases in more sectors. About 98% of Norwegian power production is renewable, where 95% comes from hydropower [2]. This makes the Norwegian power grid relatively stable, as Norway has a decent reservoir capacity. However, local grid congestion and high demands continue to be a challenge to balancing the grid. Using vehicle-to-grid compatible vehicles as temporary electricity storage is one of the possible solutions to resolve this problem.

Utilising vehicles as temporary energy storage, vehicle-to-grid, can help balance the local grid by charging during times of low demand and discharging back into the grid when demand is high. This makes vehicle-to-grid an excellent alternative for balancing the grid without performing infrastructure upgrades. However, during the vehicle-to-grid process, losses can occur at multiple stages. These losses add up, reducing the effectiveness of vehicle-to-grid.

One method that can be used to find the system losses is to calculate the round-trip efficiency and subtract it from a fully efficient system at 100%. The round-trip efficiency is calculated by using the ratio between energy output and input. Round-trip efficiency is affected by technical and environmental variables. One of the categories is component losses, where losses depend on the component used; for example, when converting alternating and direct current, losses depend on the inverter used. Another category is battery factors, such as the state of charge, where higher and lower charges can affect internal resistance. Uncontrollable factors such as ambient temperature can affect the battery's ion mobility, which affects its ability to store and deliver energy. Lastly, the battery management system can limit what the vehicle receives and gives to protect the battery, which may entail additional losses.

Schram et al. [3] found round-trip efficiencies between 79.01-87.8% per cycle, in an AC-to-AC system. This thesis will also focus on determining system losses during vehicle-to-grid operations by calculating the round-trip efficiency for a DC-to-DC system. Therefore, in this thesis, calculations have been performed for power deviations and round-trip efficiency. Additionally, how ambient temperature and state of charge affect the deviation and round-trip efficiency has been studied.

The data used in this thesis was collected by the research group NeX2G at NMBU for the research project "Grid balancing from large parking facilities with electric vehicles and commercial buildings". The project investigates the possibility of utilising energy stored in parked electric vehicles to contribute to a more flexible and economical energy use in buildings.

2 Background

This section includes information that is not directly affecting the round-trip efficiency, but gives insight into the conditions and suggested framework under which vehicle-to-grid operates. This includes information about the Norwegian power system, standards and different charger types.

2.1 The Norwegian power system

In Norway, the power grid is a monopoly regulated by the state [4]. The grid consists of three levels: the distribution grid and the regional grid, which are both operated by distribution system operators, DSO, as they are both regarded as distribution grids according to EU legislation [4]. The last level is the transmission grid, operated by Statnett, the Norwegian transmission service operator, TSO [4]. The transmission grid typically transmits voltages between 300-420 kilovolts, kV, but depending on location, they may transmit a voltage of 132 kV [5]. The regional grid transmits voltages between 33 and 132 kV, while the local distribution grid transmits between 0-22 kV [5]. This is because the local distribution grid supplies electricity to end-users with lower demands, such as homes, while the regional grid and the transmission grid supply larger end-users, such as factories [5].

Most of Norway's electricity is produced from hydropower plants. At the start of 2023, Norway had 1769 hydropower plants, with a joint installed capacity of 33,691 Megawatt, MW. These hydropower plants produce 136.49 TWh in a normal year, which is about 88% of Norway's entire power production. Norway also has a lot of storage reservoirs, 1240 at the start of 2023, with a combined storage capacity of 87 TWh. This makes production flexible, as it can easily be increased or decreased based on real-time demand, while still keeping the production costs low. The remaining 12% of Norwegian power production is mainly covered by wind farms, which cover about 11% of the total Norwegian production capacity. [6]

2.2 V2G

Vehicle-to-grid, V2G, refers to how the energy saved in batteries from electric vehicles, EVs, can be returned to the grid. This technology allows the EV owner to use their car as short-term storage. The same technology can be used for other purposes, including vehicle-to-home, V2H, or vehicle-to-everything, V2X. Using this technology is beneficial in multiple ways, as it can help EV owners reduce their total electricity bill while supplying energy to the electricity grids in times of high demand. [7]

2.3 ISO15118

ISO 15118, also known as "Road Vehicles - Vehicle to Grid Communication Interface", is a set of standards suggested by the International Organization for Standardization, ISO. The goal of ISO15118 is to establish a standardised set of protocols describing how EVs, plug-in hybrid electric vehicles, PHEVs, battery electric vehicles, BEVs and electric vehicle supply equipment, EVSE, should communicate with chargers and the grid. [8]

The vehicle and the charging structure use their own syntax and programming language, making it difficult to implement a communication protocol and standard. Due to these difficulties, a contract certificate was proposed instead of creating a common language. The contract certificate will contain information about the vehicle type and its owner. During a charging session, the contract certificate stored electronically in the vehicle is automatically read by the charging station when connected. The charging station easily determines ownership and authenticates the payment before charging the vehicle. An invoice is then sent from the certificate distributor to the certificate owner. This differentiates ISO15118 from other standards as its focus is the communication between the vehicle and the charging station, while other standards mostly focus on the components in the infrastructure. This standard will also simplify the charging process that EV owners face today, including needing different apps or cards for different charging networks. [8]

There are several advantages of the ISO 15118. The first advantage is that the electricity would be evenly distributed between property, vehicles using smart charging, and the grid, reducing the grid's load during periods of high demand. The second advantage is that it would enhance the capabilities for V2G and bi-directional charging, which can relieve some of the grid pressure. The third advantage is that this standard provides encrypted communication between the charger and the vehicle, preventing unauthorised access and leaks through radio frequency identification, RFID readings. The fourth advantage is that this standard allows the owner to bypass manually providing passwords and billing information for every session. Lastly, the fifth advantage is that these advantages combined create an optimal charging experience for the driver. [8]

There are possible disadvantages associated with ISO 15118, regarding security, bias, and acceptance. The contract certificates have security vulnerabilities, making them more exposed to cyberattacks. Another possible disadvantage could be the power the certificate providers may gain from implementing this standard, as they could, to a large extent, affect the charging. To implement this standard, automotive manufacturers must also accept the certificate distributors' authority and reliability, something few manufacturers seem willing to do. Although concerns have been raised around these problems, the consensus seems to be that they can be resolved, making the standard worth implementing, as it would support the process of EV adoption. [8]

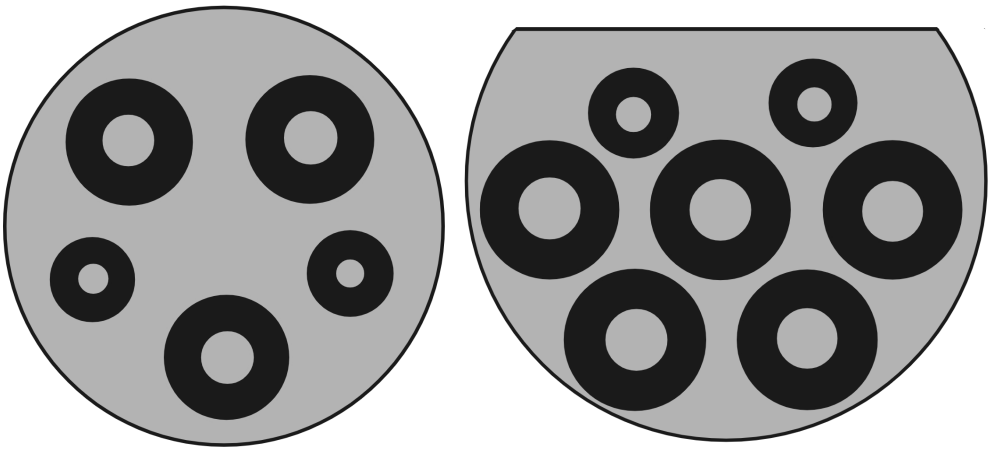
2.4 Peak shaving and load shifting

A typical way to balance the grid supply and demand is by utilising load shifting [9]. The main goal of load shifting is moving demand from hours with high demand, or peak hours, to hours with lower demand, off-peak hours [9]. With load shifting, the total energy consumption remains the same, but it sometimes results in a lower cost, optimising energy consumption [9]. Utilising V2G can contribute to load shifting by discharging into the grid during peak hours when demands and prices are high, and charging during off-peak hours when demands and prices are low.

Another technique used to optimise energy consumption is peak shaving. The difference between load shifting and peak shaving is that peak shaving is primarily used by businesses and focuses on preventing peak demand by reducing power consumption during high-demand periods through on-site power generation or using renewable energy. Meanwhile, load shifting focuses on changing the time of electricity consumption while maintaining the same overall energy consumption. [9]

2.5 Charger types

The two main charger types used for regular charging are type 1 and type 2, which is illustrated in Figure 2.1. A type 1 charger, illustrated in Figure 2.1a, only supports single-phase charging, delivering a charging power of up to 7.4 kilowatt, kW from a 230 Volt, V grid, when operating at 32 amperes, A [10, 11]. Type 1 chargers are mainly used on Asian and North American car models and older cars [10, 12]. The type 2 charger, illustrated in Figure 2.1b, was tailored for the European power grid for alternating current, AC [10]. A type 2 charger can be both single-phase and three-phase, where the three-phase option gives a faster charging speed than single-phase chargers [10]. Typically, home chargers only support one phase, while public AC chargers support three phases [10].



(a) Illustration of what a charger of type 1 looks like [10].

(b) Illustration of what a charger of type 2 looks like [10].

Figure 2.1: Illustration of the two main types of chargers that are used today.

The difference between regular charging and fast charging is how much power is delivered in kW per time unit, as well as where the conversion from AC into direct current, DC, occurs, as car batteries use DC. During fast charging, considerably more power is provided to the battery than during regular charging, defined as a power of 43 kW or more. Fast charging sends DC directly to the car battery since the conversion from AC to DC is done in the external charging station. Regular charging is defined as charging below 22 kW. During regular charging, the charger supplies AC to the car, which is converted into DC in the car's onboard charger. Semi-fast charging also exists, delivering power between 22-43 kW. Semi-fast charging also delivers AC, like regular charging, which is converted to DC in the onboard charger. [13]

Type 2 chargers are often delivered with the option of fast charging in the combined charging system, CCS, form [10]. This is a charger that can be used for both the daily AC charging, as well as DC fast charging, where the top part of the plug is the standard type 2 AC plug, while the bottom has two extra pins that enable the fast charging [14]. An illustration of a type 2 charger with CCS is shown in Figure 2.2. The charging power that can be reached with CCS has developed and used to be 50 kW, but has since reached 150+ kW [14].

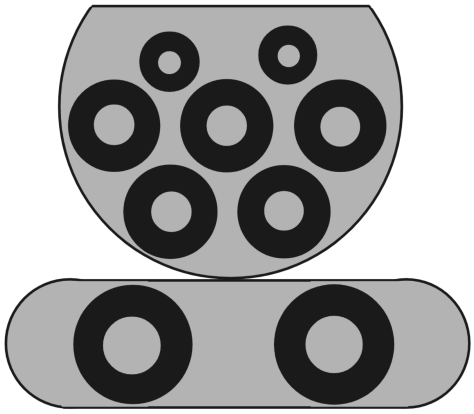


Figure 2.2: Illustration of type 2 plug with CCS [10].

Another fast charger that can be used is the CHAdeMO. The word CHAdeMO originates from "Charge de move", which means "charge for moving" [15]. CHAdeMO was developed in Japan and is based on Japan's electric vehicle standard, JEVS [14]. CHAdeMO is a charging protocol for fast charging

that uses DC and was made by Tokyo Electric Power Company, TEPCO, in 2010 [16]. CHAdeMO was on the market before CCS and used to be the standard until the car industry collectively started leaning towards CCS [14]. Today, CHAdeMO can reach a charging power of up to 50 kW. An illustration showing roughly how a CHAdeMO charger looks can be seen in Figure 2.3 [14].

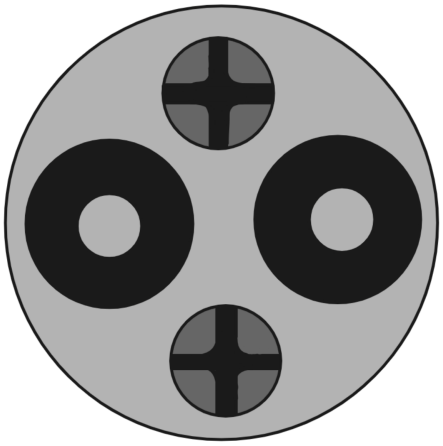


Figure 2.3: Illustration of a CHAdeMO plug [17].

3 Theory

This section covers the background theory necessary to understand V2G, as well as related concepts. This includes variables such as C-rate, round-trip efficiency, state of charge, battery management systems and more.

3.1 Lithium-ion batteries

Lithium-ion batteries are widely used as EV batteries due to their properties. These properties include high energy efficiency, high power-to-weight ratio, long lifespan, and low self-discharge [18]. Additionally, lithium-ion batteries have good high-temperature performance, with an optimal operating temperature range of between 15°C and 35°C [18, 19]. Lithium-ion batteries consist of an electrolyte, cathode, anode, separator and a positive and a negative collector as illustrated in Figure 3.1. The anode and cathode are positioned at opposite ends of the battery, separated by the separator. The electrolyte normally consists of lithium salt dissolved in an organic compound [20]. The electrolyte helps transport the lithium ions through the separator. [21]

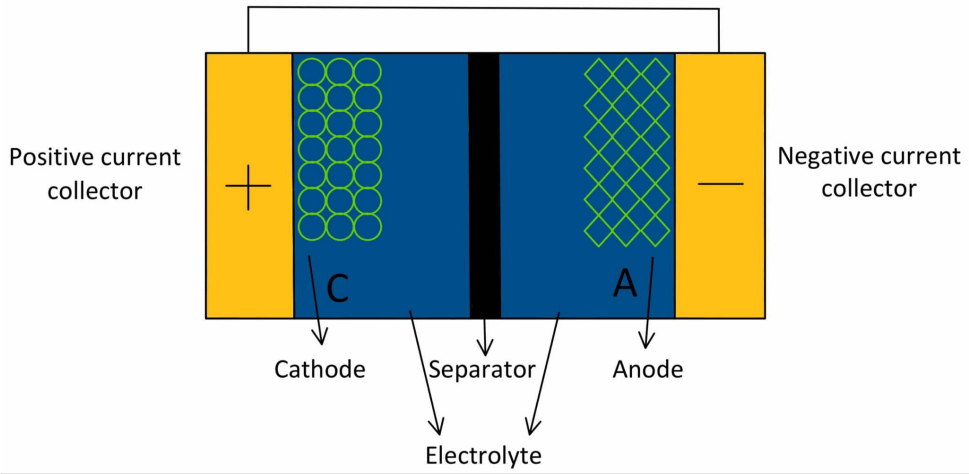


Figure 3.1: An illustration showing the structure of a lithium ion battery [21, 22].

When the battery is charging, positive lithium ions are released by the cathode and sent through the separator with the help of the electrolyte to the anode side. Simultaneously, the electrons are collected by the positive current collector at the cathode side and are sent through an external circuit before ending at the negative current collector at the anode side, restoring energy in the battery. [21]

When the battery is discharging, the opposite occurs; lithium ions with a positive charge are released from the anode, which is carried to the cathode side through the separator with the assistance of the electrolyte. Simultaneously, the electrons are collected by the negative current collector at the anode side and sent through an external circuit before reaching the positive current collector at the cathode side, providing energy to the connected device. [21]

3.2 Battery degradation

As the performance of V2G is heavily dependent on the EV's battery performance, battery degradation plays a big part in the V2G assessment. Battery degradation refers to how a battery's ability to store and deliver energy progressively declines [23]. This happens as chemical reactions ensue during both charging and discharging, making the battery's chemistry change over time, causing degradation [24]. In other words, it is a slow, natural and unavoidable process that results in an overall decline in the battery's performance. However, the extent of the battery degradation will vary based on the battery type [23].

Degradation is usually sorted into two sub-categories, calendar degradation and cyclic degradation [25]. Calendar degradation is degradation that occurs when the battery is not in use [25]. Therefore, calendar degradation is affected by variables including state of charge - SoC, and ambient temperature [25]. Solid electrolyte interphase, SEI, is a layer that grows with time and forms on the surface where the electrode meets the electrolyte [26]. Over time, the SEI grows while consuming lithium, leading to increased lithium loss, cell resistance, and fading of battery capacity [25, 26]. Cyclic degradation is caused by usage and is generally affected by the input and output of current and temperature [25]. Inside the battery, stress increases during cycle degradation as the remaining lithium expands and contracts [25]. This change in volume inside the battery results in the SEI growing larger, as the volume causes cracks in the electrode, uncovering more of the exterior to the electrolyte [25]. This additional growth of the SEI further leads to a decline in battery capacity and, in addition, causes a perpetual drop in voltage [25].

There are a lot of factors, both controllable and uncontrollable, that affect how fast or slow battery degradation transpires. Uncontrollable factors include variables like extreme temperatures and time, while controllable factors include charging habits [23]. Higher temperatures can affect the battery's electrolyte, causing it to break down and accelerate the battery's chemical reactions [24]. Operating batteries at high temperatures can lead to capacity reduction and an increase in the internal resistance, which increases the heat generation, and escalates the ageing process [27]. Additionally, lower temperatures can slow down any chemical reactions that occur within the battery [24]. To slow down the degradation process stemming from temperature, charging the batteries in an environment with moderate temperature if possible, and preheating the EV in moments of very low temperatures, are some measures that can be taken [23]. Degradation from time happens due to the fact that the chemical reactions within the battery are not completely reversible, leading to a gradual decrease in performance and capacity as time passes [24].

Each time a battery is charged or discharged, it is exposed to stress, as the ions in the battery are moved between the anode and cathode, contributing to the degradation process [28]. Therefore, to best maintain the battery's health, it is important to keep Depth of Discharge, DoD, from being too deep and avoid overcharging. This means keeping the battery charge between 20-80% to avoid full charges/discharges, slowing down the battery degradation process. Avoiding constant fast charging also slows the degradation process, as any excess heat produced can also speed up the degradation process. [23]

Overcharging a lithium-ion battery can cause several unwanted reactions. First, the electrolyte can gradually break down the surface of the cathode. As a result, the battery temperature will gradually increase. Then lithium-ion deintercalation occurs, meaning that ions are removed from the cathode. If too many ions are removed, the cathode material can become unstable. After a while, dendrites will build up on the anode. This reaction will generate gas and heat, leading to cells overheating and fracturing. [29]

An important factor affecting lithium-ion batteries' performance is the ion-transport rate [30]. At lower temperatures, the electrolyte becomes more viscous and less conductive, constraining the dispersion of ions by slowing down the transport rate [30].

3.3 Battery management system

A battery management system, BMS, is a system designed to give insight and maintenance to the battery pack, which consists of multiple battery cells. A BMS system have multiple tasks to perform, such as monitoring the battery, supplying battery protection, estimating and reporting on the battery's operational state as well as optimising the battery performance. Two of the main features of BMS include battery pack protection management, and capacity management. Battery pack protection management is further split into two sub-features, electrical protection and thermal protection. [31]

The electrical protection prevents the battery from operating outside its defined safe operating area, SOA, as it could cause damage. The SOA is constrained by the current and voltage applied to the battery cell. The current limit for lithium-ion cells is different, based on whether it is charging or discharging. The battery can handle high peak currents both during discharging and charging, but only for short periods. Most BMSes, therefore, provide current protection with a set maximum continuous current. If a sudden increase in current occurs, the BMS takes this into account and tracks these high currents. If the current stays too high for an extended period, the BMS will either completely cut off the current or reduce the available current as a peak current monitoring measure. This ability enables the BMS system to react almost instantaneously to dangerous situations such as extremely high current peaks or short-circuit conditions. [31]

Voltage-wise, the SOA sets very rigid limits to extend the battery's life. If high voltages are approaching, the BMS may require a progressive reduction in the charging current or a full termination if the limit is reached. For the BMS not to make decisions prematurely regarding the voltage state, intrinsic voltage hysteresis considerations are used. Hysteresis means lag/delay, and intrinsic voltage hysteresis refers to how a battery does not instantly return to its natural state. For example, when a battery is charging, the voltage may appear higher than it is and lower than it is when discharging. If the BMS reacts too quickly and makes a decision based on this, it might make an unsatisfactory decision, like cutting off or reducing voltage too early. Therefore, hysteresis is added, giving the battery time to let the voltage slowly settle, and the BMS can read the accurate value, making the best decisions based on this, avoiding rapid switching and unnecessary wear. [31]

Thermal protection maintains or changes the temperature, keeping the battery in its SOA. The temperature can be controlled both passively and actively, where passive cooling relies on the vehicle's movement, causing natural air flow to cool the battery, active cooling actively regulates the temperature by using cooling and heating systems, and fans to maintain optimal battery performance. A system often used is a thermal-hydraulic active cooling system, which circulates an ethylene-glycol coolant with the help of a pump, through pipes and cooling plates around the battery. This, in unison with a radiator that releases any surplus heat, keeps the battery at a safe temperature. At low temperatures, battery capacity declines, since chemical reaction rates become slower. Charging in temperatures lower than 0 °C can also be problematic as it can cause a metallic lithium coating on the anode in the battery, causing further permanent damage. BMS can prevent this by controlling the temperature to remain within the SOA limits.

One of BMS's most crucial tasks is maximising a battery pack's capacity. Battery pack stacks consist of multiple series of cells, which, right after being manufactured, may be equal in performance, but over time these cells obtain different leakage and rates of self-discharge due to temperature differences, charge and discharging cycles, and ageing. BMS tries to keep all the cells at the same charge level, since if one cell reaches the charge limit first, the charging process will be stopped, leaving the remaining cells only partially charged, reducing the battery's overall capacity. To keep the charging level between the cells similar, the BMS tries to balance the SoC between the different stacks. This can be done both passively and actively. [31]

Passive balancing is commonly used, as it is easier to implement [31]. Passive balancing works by having a resistor connected parallelly to each individual cell so energy can be dispersed, lowering the SoC of the cell [31, 32]. Active balancing, on the other hand, redistributes charge between the cells by using power conversion, which generates less heat, gives higher energy efficiency and a faster balancing time, resulting in a longer operating range [32]. When the cells become fully charged they reject any additional current, and extra energy sent to these cells transforms into heat. This causes the voltage increase rapidly, possibly becoming a liability as it may cause lasting damage and make operating conditions dangerous [31].

Preventing current, voltage and temperature from transcending the SOA is critical while charging and discharging. If the SOA limits are not followed for a longer period, the battery pack may become

compromised and thermal runaway could occur. If lithium-ion batteries are kept in a low-voltage state, dendrites on the anode side of the battery could be created, contributing to higher self-discharging rates. In addition, staying below the voltage limits could also result in memory effects and capacity loss, making it extremely important to keep the battery within the limits. As batteries carry out electrochemical processes, which are easily affected by temperature, the BMS thermal management must ensure that the battery operates within the SOA for optimal battery performance. Generally, having the BMS keep all variables within the SOA promotes a longer battery life by keeping the battery healthy. [31]

3.4 Constant current and voltage

To regulate the current and voltage during charging, constant current, CC, and constant voltage, CV, are used. During charging, the CC method is applied while the voltage in the battery increases. With time, the voltage reaches its limit, causing the charging method to go from CC to CV. When the method changes to CV, the current decreases until the charging is complete. When discharging starts, the CC method is again utilised, while the voltage decreases until it reaches its lower limit, meaning discharging is complete. [33]

3.5 State of charge

SoC is a parameter, provided in percentage, that represents the amount of charge that is available in a battery, in relation to the same battery containing a full charge [34]. Multiple variables can affect the SoC, including charge and discharge current, charging voltage, temperature, battery type, degradation, C-rate, self-discharge, DoD, user behaviour, battery management systems and energy management systems. SoC is also affected by intense temperatures, which can reduce the battery's capacity, making charging under such conditions less efficient. Over time, batteries can lose charge, although they are not in use, as a result of self-discharge. How much the batteries lose is decided by the battery chemistry, which can affect the SoC. Some battery chemistries can also display non-linear voltage responses when the battery is rapidly charged or discharged, which can impact the SoC estimations. [35]

Other variables affecting SoC is user habits, including DoD, where if the discharge is very deep and is something that happens often, it can affect the SoC. Overcharging the battery can also affect the SoC and damage the battery. Today, most EV car batteries have a BMS, which assists in managing the battery's state, including SoC, where factors like temperature, voltage and current are considered. [35]

3.6 Round-trip efficiency

Round-trip efficiency, RTE, is a measurement given in percentage, presenting how efficiently a battery can store and discharge energy [36]. To calculate RTE, the energy output is divided by the energy input and then multiplied by 100 to obtain the result as a percentage, as shown in Equation 3.1 [36]. For V2G, a higher RTE is desired because if the vehicle cannot optimally store and return the energy it receives, it will not be favourable to use as temporary energy storage.

$$\text{RTE } [\%] = \frac{\text{Energy output}}{\text{Energy input}} \cdot 100 \tag{3.1}$$

3.7 C-rate

C-rate measures the ratio between the current it takes to charge/discharge the battery, given in A, in relation to the battery's capacity, shown in ampere-hours, Ah. Equation 3.2 shows the C-rate calculation method. In other words, C-rate describes how fast a battery can be charged or discharged.

If a battery has a capacity of 5 Ah, and is being charged/discharged at a rate of 5 A, then the C-rate would equal one.

$$\text{C-rate} = \frac{\text{Current [A]}}{\text{Battery capacity [Ah]}} \tag{3.2}$$

Table 3.1 shows a few C-rates and their conversion. The higher the C-rate is, the shorter the charging/discharging session will be. Understanding C-rates is essential, since it can be helpful in preventing overcharging/over-discharging, ensuring safe operation by keeping the C-rate within regulatory limits, and minimising risks, such as thermal stress. Knowing the C-rate of batteries allows for comparisons between different battery types and sizes. [37]

Table 3.1: Examples of C-rates showing how C-rates are converted into charging/discharging time [38].

C-rate	**Time [h:min]**
5	0:12
2	0:30
1	1:0
0.5	2:0
0.2	5:0

4 Previous studies and results

Relevant papers had to be studied to compare results later. The results of these studies are presented in this chapter, including a summary of their methodology and a representation of how the results were calculated. Both Apostolaki-Iosifidou, Codani, and Kempton [39] and Schram et al. [3] have found AC-to-AC losses. Apostolaki-Iosifidou, Codani, and Kempton [39] measures power losses between the grid and the battery, making it an AC-to-AC study since inverter losses are included. Similarly, Schram et al. [3] has placed the point of measurement between the grid and the charger station, meaning inverter losses are included, making it an AC-to-AC study.

4.1 Previous round-trip efficiency studies

In Apostolaki-Iosifidou, Codani, and Kempton [39], power loss during charging and discharging of EVs was studied. For this study, two BMW MiniEs were used. The measured components in the study included the EV battery, power electronic units, the EVSE, breakers and the transformer. A DC and AC meter were used to measure current, voltage and power in the different components, and a building sub-meter was used for the measurements related to the transformer. Round-trips had a SoC range between 20% to 80%, and measurements were taken for amps between 10 A to 70 A. Measurements were only taken in moderate indoor temperatures, ranging from 25°C to 28°C. The round-trips were only performed on the EV components. The losses of the building components, including EVSE, breakers and the transformer, were calculated according to Equation 4.1, where the l stands for the power losses in %, while P_{in} and P_{out} represents the instantaneous power that goes in and out of the component measured. [39]

$$l = 100 \cdot \frac{P_{\text{in}} - P_{\text{out}}}{P_{\text{in}}} \tag{4.1}$$

Losses were calculated slightly differently for the EV components, including the EV battery and power electronic units. As mentioned, round-trips were performed for these components, where a predetermined SoC interval was followed with fixed AC values. In addition, the round-trips performed were neutral coulomb cycles, meaning the number of coulombs that went in and out of the battery was equal. The cumulative energy flowing into and out of each component was calculated first by using Equations 4.2 and 4.3. To calculate the actual losses, the cumulative energy results were then used in a modified version of Equation 4.1, shown in Equation 4.4. [39]

$$E_{\text{in}} = \sum_{i=1}^{N} \int_{t_i}^{t_i + t_{\text{c}}} V_{\text{in}}(t)\, I_{\text{in}}(t)\, \mathrm{d}t, \tag{4.2}$$

$$E_{\text{out}} = \sum_{i=1}^{N} \int_{t_i + t_{\text{c}}}^{t_i + t_{\text{c}} + t_{\text{d}}} V_{\text{out}}(t)\, I_{\text{out}}(t)\, \mathrm{d}t, \tag{4.3}$$

$$l = \frac{E_{\text{in}} - E_{\text{out}}}{E_{\text{in}}} \cdot 100\%, \tag{4.4}$$

In Equation 4.2 and 4.3, t_i is the ith coulomb cycles starting time, t_c is the time duration of a single ampere-hour neutral charging cycle in seconds, t_d is the time duration of a single ampere-hour discharging cycle in seconds. $V_{in}(t)$ and $V_{out}(t)$ are the instantaneous voltage on the AC/DC line/bus, at time t, and $I_{in}(t)$ and $I_{out}(t)$ is instantaneous current on the AC/DC line/bus. [39]

Apostolaki-Iosifidou, Codani, and Kempton [39] found EVSE losses between 0.10% to 1.42%, breaker losses between 0% and 2.80%, transformer losses between 10.20% to 14.60%, EV battery losses of 0.64% and power electronic unit losses between 6.28% and 16.67% when the AC gives 10 A. This

provides a total loss of between 17.22% and 36.13% when an AC of 10 A is used. When the AC is 40 A, losses vary between 1.69% and 1.91% for the EV battery, 5.77% and 19.23% for the power electronic unit, 0.29% and 1.39% for the EVSE, 3.33% and 6.65% for the transformer and 0.60% and 1.30% for the breaker. Apostolaki-Iosifidou, Kempton, and Codani [40] later replied to a comment on their study, stating that the higher end of the reached round-trip efficiency would be closer to 70%, as a typical transformer case would be losses of about 20% for the EV battery, about 2% for the EVSE, about 3% for the breakers, 4% for transformers, giving a total loss of around 29%.[39, 40]

Schram et al. [3] also researched round-trip efficiency. In this study, two different setups were used, where the first one had the inverter inside the charging station, while the second used an on-board inverter. In the first setup, a Nissan Leaf was charged and discharged with an eNovates DC V2G 10 kW charging station. In contrast, the second setup utilised a V2G prototype version of the Renault ZOE, with a WeDriveSolar v1.1 charging station paired with a Last Miles Solutions controller. Both EVs reported their SoC status on a two-second basis, and multiple charging/discharging cycles were performed. These cycles were performed so the EV would start at a SoC value, charge up to a predetermined SoC, and then discharge back to the SoC value the cycle started at. [3]

After performing the discharging and charging cycles, the AC-to-AC round-trip efficiency was calculated by taking the negative sum of exported watts multiplied by the delta time from the charging station and dividing it by the imported watts multiplied by the delta time. Delta time represents the interval between two power measurements; in this study, it was two seconds. Therefore, the sum of watts exported/imported in the cycle is multiplied by two to estimate the most accurate energy. This is because the measurements in the study are taken on a two-second basis, and since power is energy transferred per second, including every measurement twice would cover every second in the cycle. The formula used to calculate the round-trip efficiency in Schram et al. [3] is shown in Equation 4.5. [3]

$$\eta = \frac{E_{\text{out}}}{E_{\text{in}}} = \frac{-\sum_{t_{\text{SoCmax}}}^{t_{\text{SoCmin,end}}} P_t \cdot \Delta t}{\sum_{t_{\text{SoCmin,start}}}^{t_{\text{SoCmax}}} P_t \cdot \Delta t} \tag{4.5}$$

The results achieved in Schram et al. [3] were round-trip efficiencies per cycle between 79.1% and 87.8%. The visit with the highest average round-trip efficiency came from the Nissan Leaf visit on 23/04/2019, where four cycles were performed, giving an average round-trip efficiency of 87.0%. It was also observed that the average test result retrieved in April had a higher efficiency than the results retrieved in November of the same year. This was mentioned as likely due to a higher average temperature in April at 15.3 °C, in comparison to November at 5.5 °C. [3]

Schram et al. [3] also concluded that charging efficiency tends to be higher when the SoC is medium, around 50%, in contrast to higher/lower SoCs, such as 15% and lower or 85% and higher. Schram et al. [3] also stated that the results align with previous research, meaning that batteries display higher internal resistance and heat generation during charging and discharging when the SoC is low. Still, the differences are small enough to be considered insignificant, meaning that V2G can still be used in low and high SoCs without impacting the efficiency. Schram et al. [3]

4.2 V2G effects on battery degradation

In Sagaria, Kam, and Boström [25], a study was conducted on how BEVs with lithium-ion batteries degrade when used with variable C-rates in V2G. The study considered six scenarios, three from separate user profiles and three from the fleet perspective. A co-simulation model calculated battery degradation until the battery reached the end of its life. The battery's end of life was defined as when it could only obtain 80% capacity compared to its original capacity threshold. Usually, the degradation of a lithium-ion battery over 10 years without V2G mainly consists of calendar degradation, which accounts for about 85% to 90% of the degradation, while cyclic degradation

accounts for the remaining 10% to 15%. With V2G, the study showed that the cyclic degradation increased between 20% and 25%, resulting in an average total degradation increase of 0.31% yearly, when 33 cycles were performed yearly. Overall, over 10 years, the battery degradation rate increases between 9% and 14% when V2G is applied. [**sagaria_vehicle–grid_2025**]

5 Case description

The data used in this thesis has been collected through the research group NeX2G at NMBU in Norway. In this project, a particular zone in the P10 parking lot at Gardermoen was reserved, and a parking spot in Ås and chargers that could handle V2G were installed off-board. Specifically, Wallbox's Quasar 1, with a maximum charging power of 7.4 kW [41]. The setups are shown in Figure 5.1, where Figure 5.1a shows the Gardermoen setup and Figure 5.1b shows the Ås setup. After the chargers were installed, anyone interested in the project who owned a Nissan Leaf or Nissan E-NV 200 that was made after 2013 could send an e-mail stating their interest, the period their car would be parked, and their desired SoC at departure. Only the Nissan Leaf and Nissan E-NV 200 were accepted because these are the ones accepted for use in V2G, while also using the CHAdeMO standard. After receiving the e-mail, the car registration numbers would be sent to Avinor and OnePark for approval. The participants were also told to turn off any charging management system that the car may have.

To best preserve the battery's capacity, the battery would be kept at a SoC of 50% after arrival at the charging location and in times when algorithms were not being tested for charging/discharging. Before departure from the charging station, the battery was charged to the car owner's desired SoC, as long as it was kept below 93%, to preserve the battery's capacity.

To send commands to the chargers, a secure server at NMBU was built so that commands could be sent remotely. Through this server, one could get remote access to the secure server at Gardermoen, which in turn gave access to the chargers. This way, one could manually send instructions to the chargers. Two programming languages were used to send commands, Structured Query Language, SQL and Python. SQL is a programming language that is used for storing, changing and fetching data found in databases, while Python is more versatile and can be used in most situations [42].

The commands were made in SQL, where the requested charge/discharge was set for different times throughout the cars' stay. These queries were then sent through Python to apply MODBUS. MODBUS is a data communication protocol that quickly transfers data between intelligent devices [43]. The queries are further sent to the Ås chargers or the secure server in Gardermoen. The queries sent to Gardermoen are additionally sent through the land line to the chargers in the airport. Then the data, like SoC status, was retrieved through the BMS and sent back to SQL, which stored the data. This process was done multiple times to collect data from different vehicles.

(a) Figure showing the charger setup in Gardermoen. *(b) Figure showing the charger setup in Ås.*

Figure 5.1: Figure showing the charger setups from the NeX2G project in the different locations.

6 Methodology

This section explains how the research was conducted and provides in-depth details on how the data was collected and filtered. Any limitations, assumptions, equations, or programming used are also specified.

6.1 Process description

Below is a list showing the steps taken to obtain the results. A more detailed description will follow. Additionally, Figure 6.1 shows a flow chart illustrating the order of steps and when each dataset was incorporated into the process.

- Cleansed the EV data in RStudio.
- Retrieve weather data for the period data was obtained, and incorporate it into the cleansed EV dataset.
- From the cleansed EV dataset, it was split into eight separate datasets, one for each vehicle.
- Checked if any observations from the cleansed EV dataset were not used in any car datasets.
- Made a function that could identify cycles and calculate the RTE.
- Calculated the RTE for every identified cycle in each vehicle dataset and then calculated the average RTE from all the cycles to get the average RTE for each vehicle.
- Combined the separate vehicle datasets to look at SoC and temperature effects.
- Calculated the deviation, both per temperature and SoC class.
- Calculated the average RTE per battery capacity.
- Calculated average RTE per temperature and SoC class.
- Analysed the results.

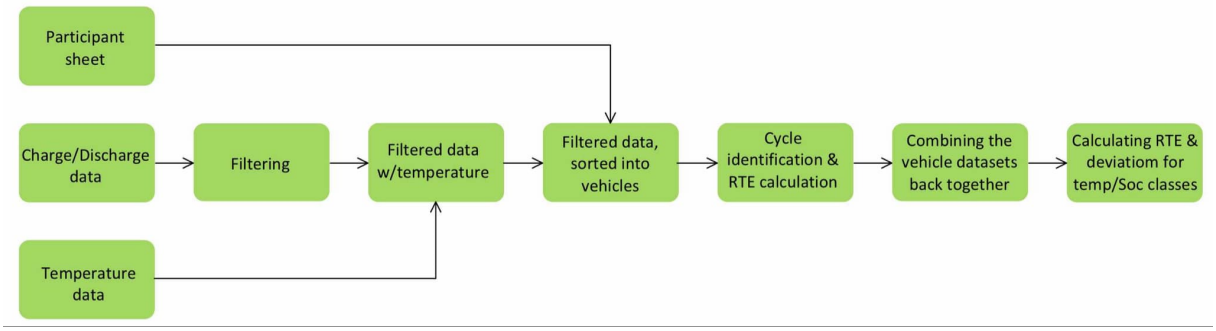


Figure 6.1: The order of steps performed.

6.2 Assumptions

A list of assumptions is provided to provide transparency about what the results have or have not considered. This includes assumptions regarding the data collection and other factors that have been assumed negligible.

- The EV data was collected through the vehicles' BMS. Though these measurements are not 100% accurate, due to factors as hysteresis, it was assumed to be accurate enough for the analysis.

- Temperature may have varied from what the weather station measured, as the cars are parked inside a concrete building, but since the concrete building has a very open solution, a lot of air can still flow through the building. Therefore, it was decided that these temperatures were close enough.
- It is unknown how much the temperature inside the vehicle affects the V2G results. Since no temperature measurements were taken inside the car, this variable is not considered in this thesis.

6.3 Data sheets

Two main data sheets were given. The participant sheet contained information regarding all the participants, including their car registration numbers, the dates they were connected to the charger, and which charger they were connected to. The other data sheet, the EV dataset, contained the variables SoC in percentage, charger ID, the requested charge in watts, W, power wattage in watts, the date and time, and the charger status. Throughout this thesis, the columns will be referred to as such. A row will also be referred to as an observation. Figure 6.2 shows some observations from the pre-filtered EV data. In the EV dataset, charger zero represents the charger in Ås, while chargers one, two, three and four are located in Gardermoen.

	ev_log_id	charger_id	ev_datetime	soc	power_wattage	charger_status	requested_charge
35	35	0	2023-02-07 15:47:00	90	6591	Charging	7200
37	37	0	2023-02-07 15:47:00	90	6591	Charging	7200
39	39	0	2023-02-07 15:48:00	90	6599	Charging	7200
41	41	0	2023-02-07 15:48:00	90	6603	Charging	7200

Figure 6.2: Sample showing how the unfiltered EV dataset looked.

Throughout the process, additional variables were added to the EV dataset as shown in Figure 6.3. Where columns have the same value, such as energy in, energy out and RTE, it is because it is calculated for the entire cycle, and every observation in that cycle is then marked with the value for that cycle.

temp	battery_capacity_Wh	time_diff_hours	new_visit	cycle_nr	delta_t	delta_t_watt	energy_in	energy_out	RTE	dataset_id
9	40000	46.21666667	1	1	480	-960000	55328232	47500902	85.85292	RTE_cycle_40_1_new
9	40000	0.13333333	0	1	480	-960000	55328232	47500902	85.85292	RTE_cycle_40_1_new
8	40000	0.13333333	0	1	480	-480480	55328232	47500902	85.85292	RTE_cycle_40_1_new
8	40000	0.13333333	0	1	540	-540000	55328232	47500902	85.85292	RTE_cycle_40_1_new
8	40000	0.15000000	0	1	480	-479040	55328232	47500902	85.85292	RTE_cycle_40_1_new

Figure 6.3: Sample showing what variables were added to the EV dataset when the filtering and cycle identification were done.

6.4 Data cleansing

The participant sheet and the EV dataset needed to be thoroughly cleansed to establish a solid foundation for the thesis. This included fixing any inconsistencies and filtering out any irrelevant data.

6.4.1 Participant sheet

The participant sheet was carefully filtered to get the most accurate basis for this thesis. The documentation was not ideal since the participant sheet was started later in the research. In addition,

due to human error and charger errors, not all registered visits occurred, the wrong charger was used, or data failed to be retrieved. Therefore, the sheet was manually and carefully reviewed to double-check with the dataset to confirm or refute whether participants showed up and whether data had been registered during that time frame.

The first participant filtering was based on checking the dates noted by the research team, checking them against the dataset, and adjusting the time of arrival and departure, as participants gave an estimation. Some arrived earlier/later than presumed. If data was not found for the period written or the dates a participant was supposed to be present were skipped in the dataset, it was assumed that the participant did not show, and the participant sheet was adjusted accordingly.

Occasionally, the used charger was not written in the participant sheet, or the wrong charger was written because the participant used a charger other than the one they had been assigned. This was fixed by matching the date and approximate time of the visit to whichever charger was in use in that period. It was also checked if two cars were present on the exact dates that they were assigned to different chargers, as one of the visits always listed the correct charger. This meant the second car had to be plugged into whichever other charger was currently in use. After filtering and adjusting the participant sheet, the result showed that eight cars participated as of February 2025. The car models that participated and were used in the study, and the cars' battery capacity, are shown in Table 6.1.

Table 6.1: An overview of the car types being analysed in this project.

Car	Amount
Nissan Leaf 40 kWh	4
Nissan Leaf 62 kWh	4

6.4.2 EV dataset

As the original dataset consisted of over 800,000 observations, it was determined that any observations containing non-helpful information were removed. Observations determined to be unhelpful included observations where the SoC, power wattage, and requested charge were all equal to zero, which meant nothing was connected to the charger. It was also discovered that some duplicates of the same data existed, meaning the two observations had the same EV log ID, which should not be the case. These were also eliminated, so only one of each unique input was left. There were also a couple of observations where the value of the requested charge was stated as "NULL" or "NA". For these observations, the observations with "NA" were first changed to "NULL" as Rstudio does not recognise "NA", and then all observations with "NULL" were removed since if an uncompleted observation is to be used, it can give an incomplete picture or skewed results, further contributing to wrongful interpretations.

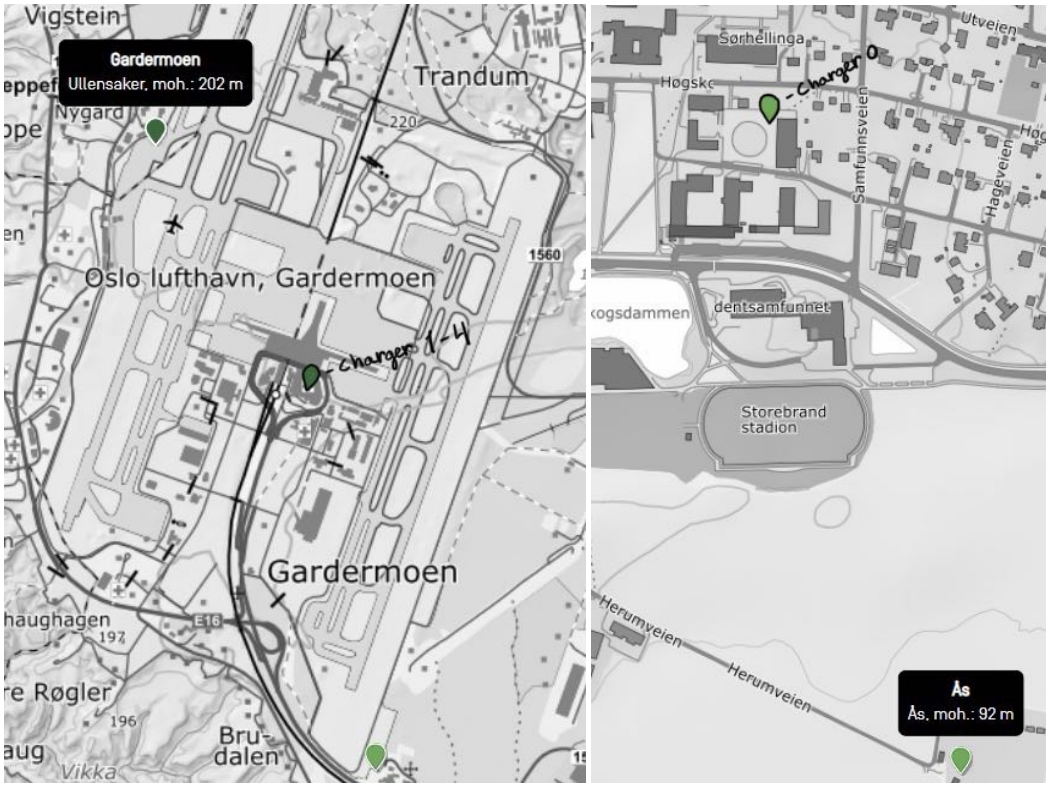
Only observations containing a value in SoC, requested charge and power wattage were kept. If values were present in the requested charge and power wattage, but the SoC was zero, it would still be sorted out. This was done to focus on V2G-relevant actions. Some observations were also observed to have a power wattage of more than ±60 kW. These observations were removed, as they were likely due to an error in the BMS system. It was also decided to remove any observations where the SoC was equal to, or higher than 92%, as this is when the BMS system would kick in, and take over the charging process to protect the battery. These observations were therefore not kept, as the thesis focused on the requested energy inputs and outputs.

6.5 Temperature data

As the relationship between temperature and V2G was to be investigated, the relevant temperature data had to be obtained and integrated into the dataset. Two different data sets had to be used, one for Ås and one for Gardermoen. The site seklima.met.no was used to retrieve temperature data [44].

For time resolution, minutes were initially chosen; however, when selecting minutes for the dataset from Ås, the warning "This combination of weather element(s) and station(s) and time frame has no data" appeared, as well as the message "931680 observations has to be downloaded, please limit the selection" beside the "show result" button. Therefore, the time resolution for the Ås dataset was changed to hourly, while the Gardermoen resolution was kept on a 30-minute basis.

In weather elements, air temperature, given in °C, was chosen, and the timeframe was set to the dates 7th of February 2023-14th of November 2024, as these are the dates that remained after having filtered the original dataset. The specific weather stations chosen were SN17850 in Ås as illustrated in Figure 6.4b, and SN4780 in Gardermoen as illustrated by Figure 6.4a. The location of these weather stations is shown in Figure 6.4, where the location of the chargers has also been marked to illustrate the distance. The CSV files were then downloaded.



(a) Map showing the location of both the weather station used to find the weather data in Gardermoen, as well as the location of the chargers 1-4 [44].

(b) Map showing the location of both the weather station used to find the weather data in Ås, as well as the location of charger 0 [44].

Figure 6.4: Map showing the distance from the weather station to their respective chargers [44].

6.6 System boundary and losses

Since the BMS was utilised to retrieve the measurements, this study only reflects DC-to-DC losses. Figure 6.5 illustrates the system boundary that was set for this thesis, where the vehicle interface represents the place where the EV charger meets with the vehicle to exchange power. As the BMS was used to retrieve all measurements and is designed to monitor the battery, it performs its measurements where the current enters or exits the battery. This results in the measurements including losses that occur due to internal resistance or temperature in the battery. Losses that are not accounted for, therefore, include any losses that occur between the charger and the grid, such as losses due to cable resistance, both during charging and discharging. Any losses between the vehicle and the charger are also not accounted for. Typically, the components used in the system affect the results, such as the inverter. However, as the inverter is in the off-board charger, any losses during the

AC/DC or DC/AC conversion are not considered, as they are outside the scope of this thesis.

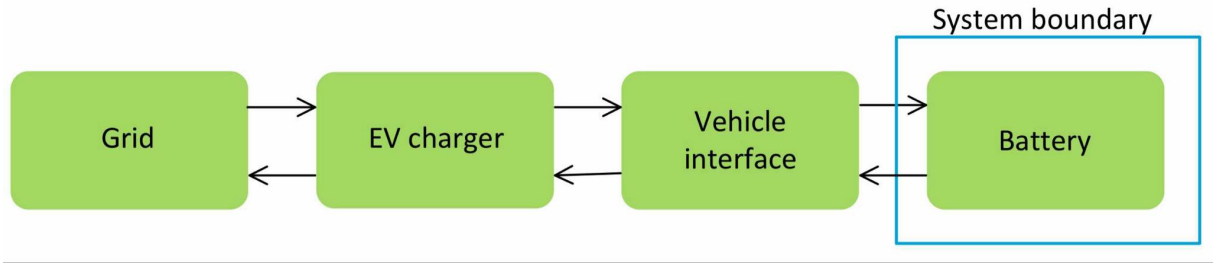


Figure 6.5: The defined system boundary for this thesis.

6.7 Cycle detection and AI

To improve the quality and clarity of the thesis, Grammarly was used for proofreading tasks. Grammarly has not been used as generative AI but to assist with editing. Grammarly has done tasks such as correcting grammar or spelling mistakes and suggesting punctuation where appropriate. Additionally, Grammarly pointed out longer sentences, which assisted in deciding if they should be shortened. Lastly, Grammarly sometimes suggested switching a word in a sentence, such as "multiple" instead of "many". Each suggestion made by Grammarly was reviewed, and it was decided whether or not it should be implemented based on the context.

It was decided that the best way to calculate the RTE would be to base it on the charging/discharging cycles. Since the datasets did not come with set cycles, these cycles had to be identified. Going through every dataset and defining these cycles manually into the dataset would be highly time-consuming. Therefore, a decision was made to create a function to perform this task. Due to all the limitations this function had to remember, it became advanced, and therefore, AI, specifically ChatGPT, was used to build it. Additionally, ChatGPT was utilised to assist with the other parts of the programming, such as explaining any error messages or creating code, for example, the code for defining what data belonged to which vehicle.

Creating the cycle finder function started by showing ChatGPT a sample of the filtered dataset and describing the basic concept. This included setting a goal to create a function in RStudio that would define cycles, assign them a special ID, and impose some fundamental limitations. It started simple, and then the process was gradually built upon by adding variables along the way, as new problems or barriers were encountered. The different versions of the function provided by ChatGPT were also checked multiple times during the process and manually calculated where possible to check that the function met all requirements. If any mistakes were found when implementing the function, this was written to ChatGPT to explain how it was supposed to work, often with an example, to fix the mistake. With the help of ChatGPT the cycle detection function was made, and can be seen with the rest of the code, in Appendix A.

The function had two primary missions. The first mission was cycle detection, and the second was RTE calculation. The function had to take several requirements into consideration. Cycles were defined as going from charging to discharging or from discharging to charging, since different visits/vehicles might start with different statuses depending on their SoC when they park. The function was also assigned a prioritisation order, where it was preferred to create a perfect cycle where possible, meaning that the start SoC, the first observation in the cycle, equals the ending SoC, the last observation in the cycle.

If a perfect cycle could not be identified, the subsequent prioritisation was to find a next-to-perfect cycle with a SoC allowance of ±3%. This ±3% allowance more specifically meant that if the cycle ends in discharging, the SoC could go 3% less down than what the start SoC was (start SoC +3%), while if the cycle ends in charging, it could go 3% over what the start SoC was (start SoC -3%). It was not

allowed the other way around, since that would have meant a perfect cycle could have been created instead. An example of a ±3% scenario would be if a cycle begins with charging at SoC equal to 43%, charges up to 80% and then discharges back to 46% before it starts charging again. It would still be allowed to count as a cycle, as in this instance, it would not be possible to go back down to 43%.

The last prioritisation was that if neither perfect cycles nor perfect cycles plus the allowance can be produced, the function should start looking for cycles that do not start at the first observation after the change of charging status. So, for example, if a cycle goes from charging at SoC equal to 20%, charges up to SoC equal to 75% and then discharges back down to 35%, this would be outside the ±3% SoC allowance, but there could still be made a cycle from this if it starts later at SoC equal to 35% then goes up to 70% and down to 35% again.

Other requirements the function had to follow included only finding complete cycles within the prioritisation guidelines, and if data did not fit into any of the three prioritisations, it was ignored and not put into any cycle. Occasionally, multiple observations in a row had the same SoC, for example, at the start or end of a cycle, the function was coded to include all of them, not just the first/last. So, if a cycle ended with three observations with a SoC equal to 73%, all three would be included in the same cycle. It was also decided to include a minimum amount of discharging and charging observations to count as a cycle to avoid small cycles with extremely low or high RTE results. This minimum was set to three observations for discharging and three for charging. Additionally, a variable called SoC transition threshold was made, stating that if the cycle, when switching charger status, had lost more than 10% SoC, this would not be counted as a valid cycle as it would significantly affect the RTE calculations.

The function also had to consider that a car dataset can consist of multiple visits and that a singular visit can consist of numerous charging/discharging cycles. This was incorporated into the function by defining that if more than 24 hours passed, this would be a new visit from the same car. The function was also programmed not to make a cycle from split data. For example, charging observations followed by discharging observations with a time difference of 24 hours or more cannot make a cycle together. In this example, the discharging observations would belong to a new visit, while the charging observations would belong to the previous visit. The function was also modified to count the new visits and cycles, which would count upwards from one, and then reset with every new visit. These counts were put into their new columns in the dataset so the cycles could be manually reviewed to verify that the function worked as intended.

When calculating the RTE, energy output and input were needed. As the dataset only contained information about the power, energy had to be calculated by multiplying the power by seconds, as per the Equation 6.1, to obtain the energy in joules, J. Inspired by Schram et al. [3] a new variable called delta_t was created for every observation, which stated the time difference between the present observation and the following observation in seconds. Another variable was called delta_t_watt, which followed Equation 6.1 and calculated the power wattage multiplied by the delta_t, so every second in the cycle would be included. Limitations had to be applied to this part, as in the last observation of a cycle, delta_t would consist of the time between the last observation of the cycle until the next observation. This caused the RTEs to be extremely low or high since additional energy would be added into the charging/discharging when this did not occur, as the vehicle's cycle had either stopped. Therefore, a limit was put into the function where if it is the last observation of a cycle, delta_t is automatically set to one.

$$J = W \cdot s \tag{6.1}$$

It was also observed during the programming that some cycles contained observations with significant time gaps between them, either in the middle of charging/discharging or in the charger status switch.. A limitation was added to avoid this affecting the results: if the time gap is more than

30 minutes between the observations, it is assumed that the charger is not in use and is on a break, and delta_t is set to one. The function was then programmed to calculate the RTE for every detected cycle and put the results into a column. The function also calculated and printed the total energy output and input, as well as the average RTE for the dataset, by calculating the average of all the RTEs from the detected cycles in the selected dataset. The results from the function are shown in the Appendix B, and a flow chart showing how the function handles the data is illustrated in Figure 6.6.

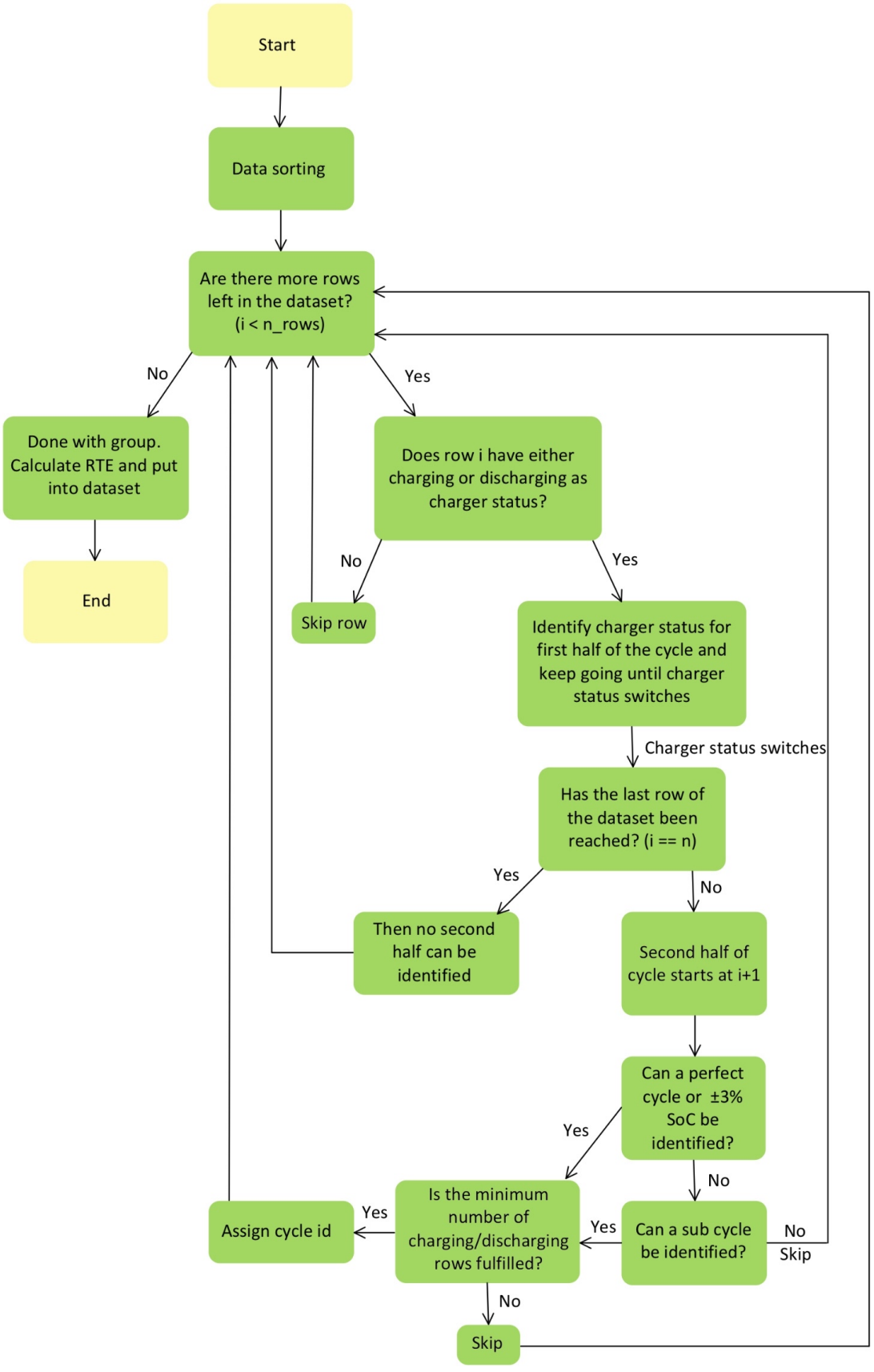


Figure 6.6: Flow chart showing the most important steps in the function, illustrating how the function works.

6.8 Calculations

To look into how deviation and RTE are affected by temperature and SoC, they first have to be calculated. As multiple connections were going to be studied, various versions of the same equations were used.

6.8.1 Deviation

Deviation, in this case, refers to the difference between the requested charge and the power wattage. Although the deviation does not directly affect the RTE in this thesis' calculations, any consistent deviation can indicate inefficiencies or losses in the system. This will indirectly affect the RTE. The two factors suspected of influencing the deviation were the vehicle's SoC and temperature. Therefore, the deviation was calculated and categorised based on those two aspects.

The average deviation for different temperature classes was calculated to examine how temperature affected deviation. For this approach, only observations belonging to cycles were used, where observations from the same cycle were grouped. For each cycle, the average temperature was calculated. Then, all observations in the cycle would be put into the corresponding temperature class, based on the average temperature. Consequently, after all the cycles had been sorted into their respective temperature classes, the average deviation for each temperature class would be calculated as a percentage by following Equation 6.2.

$$\text{Deviation } [\%] = 100 - \left(\left(\frac{\sum\limits_{\substack{i \in \text{temperature class} \\ i \in \text{cycle}}} \left| P_{\text{actual},i}[W] \right|}{\sum\limits_{\substack{i \in \text{temperature class} \\ i \in \text{cycle}}} \left| P_{\text{requested},i}[W] \right|} \right) \cdot 100 \right) \tag{6.2}$$

To look into possible connections between deviation and SoC it was decided only to consider the observations included in a cycle, as these are the data that best reflect energy transmission. SoC classes with a gap of 10% were created, where the SoC classes started at 21% and ended around 91%. These classes were picked because 21% was the lowest observed SoC in a cycle, while the highest observed SoC was 91%. The observations belonging to cycles were sorted separately into their respective SoC classes, without considering which cycle they belonged to. The absolute value of the requested charge and power wattage was used to calculate the average deviation separately for charging and discharging, as represented in Equation 6.3, to obtain the deviation expressed as a percentage. It was decided to calculate deviation as a percentage because a missing 100 W is more severe if it is missing from a 1 kW reading than from a 7 kW reading.

$$\text{Deviation } [\%] = 100 - \left(\left(\frac{\sum\limits_{\substack{i \in \text{SoC class} \\ i \in \text{charger status}}} \left| P_{\text{actual},i}[W] \right|}{\sum\limits_{\substack{i \in \text{SoC class} \\ i \in \text{charger status}}} \left| P_{\text{requested},i}[W] \right|} \right) \cdot 100 \right) \tag{6.3}$$

It was decided to calculate deviation differently for SoC and temperature, although both were planned to be sorted into classes. When considering deviation, it was decided to use all observations from the same cycle and place them in the same temperature class. For SoC classes, it was decided to separate observations from the same cycle and focus solely on the SoC of each observation. This was because temperature throughout a day typically tends to have less variation, while a cycle can, depending on the cycle, have a SoC range between 0% and 100%. Therefore, it was decided to create SoC classes of 10%, and temperature classes of 5%.

6.8.2 Calculation of RTE

With the information in the dataset, it was decided to use the power wattage in this calculation. Power wattage was used and not requested charge because the RTE equation uses energy output and input. Although the requested charge is what the car is supposed to receive/give, the power wattage is what is genuinely transferred to or from the vehicle. Therefore, using the requested charge for the RTE calculations would be unrealistic, resulting in an energy output that is either higher or lower than what the car was genuinely capable of with the energy it received. For the RTE calculation, a round-trip was defined as the battery going from one SoC, going up or down from that SoC and then returning back to the exact or close to the same SoC the cycle started at.

RTE was calculated for every cycle in every car dataset, and then an average RTE was calculated from those cycles' RTEs. This was performed by calculating the sum of all energy output in a cycle divided by the energy input in the cycle, where the energy output is the sum of all the observations from delta_t_watt in a cycle with discharging status, and energy input is the sum of all the observations from delta_t_watt in a cycle with charging status. The absolute value was used on the sum of energy output to avoid negative RTE values from discharging. The energy output was then divided by the energy input and multiplied by 100 to calculate the RTE as a percentage in each cycle.

The RTE calculation method is shown in Equation 6.4, which is a modified version of the original RTE Equation 3.1 mentioned in Section 3.6, which is used for each cycle. In Equation 6.4, P_i is the power wattage at observation i and Δt_i is the time between observation i until the next observation in seconds. In total, 104 cycles were detected, and although some cycles had an RTE above 100% and some had an RTE of less than 50%, it was decided to keep all results, as the cycles were still valid by the definition that was set.

$$\text{RTE } [\%] = \left(\frac{\sum_{i \in \text{Discharging}} |-P_i[W]| \cdot \Delta t_i[s]}{\sum_{i \in \text{Charging}} P_i[W] \cdot \Delta t_i[s]} \right) \cdot 100 \tag{6.4}$$

For temperature, the calculation method was similar to the one used for the deviation. The average temperature was calculated for each cycle, similar to how it was calculated for deviation. However, for RTE, a single observation was created from each cycle that included the RTE for that cycle and the calculated average temperature. These observations, representing each cycle, were used when sorting cycles into the corresponding temperature class of the average temperature. After sorting all cycles, the average RTE could be calculated by adding them together and dividing the sum by the number of observations sorted into the temperature class, as shown in Equation 6.5. RTE was sorted differently with SoC compared to temperature due to the difference in temperature and SoC ranges.

$$\text{Average RTE } [\%] = \frac{\sum \text{RTE}_{\text{temperature class}} \ [\%]}{n_{\text{observations, temperature class}}} \tag{6.5}$$

To find patterns between RTE and SoC, the average RTE was calculated for each SoC class. The first step was to calculate the average SoC for each cycle. Then, a single observation with the average SoC and the RTE for that cycle is sorted into the SoC class corresponding to the average SoC. This was done for every cycle. After every cycle had been sorted, the average RTE was calculated by adding all RTEs belonging to the same SoC class and dividing the sum by the number of observations in each SoC class, as shown in Equation 6.6.

$$\text{Average RTE } [\%] = \frac{\sum \text{RTE}_{\text{SoC class}} \ [\%]}{n_{\text{observations, SoC class}}} \tag{6.6}$$

7 Results

From the eight cars, a total of 104 cycles were identified, with RTEs between 2.51% to 139.09%, resulting in an overall average RTE of 82.90%. On average, a cycle went up or down 14.60% from its original SoC, and had a temperature range of 2.9°C. Out of the 104 cycles identified, ten cycles had an RTE of more than 100%, where five cycles were discharging cycles, and five cycles were charging cycles. Eight cycles had an RTE of less than 50%, with one of the charging cycles below 10%. Out of the cycles below 50%, six were discharging cycles, and two were charging cycles. The remaining 86 cycles varied between an RTE of 54.66% to 99.87%. The most significant SoC range can be found in cycle number three from the first 40 kWh vehicle, with a SoC range of 45%, and the smallest SoC range originates from the first 62 kWh vehicle in cycle number 21, where the cycle never changes SoC, resulting in a range of 0%.

7.1 Deviation trends across temperature classes

In Table 7.1, the average deviation between requested power and power wattage has been calculated per temperature class. The highest deviation comes from the [-5, 0⟩ class, at a deviation of 6.61%, and the lowest comes from the [-20, -15⟩ class at 0.44%. It can be observed that the deviation is at about 6% both in the [-5,0⟩ class, but also at [15, 20⟩ class.

Table 7.1: An overview of the average deviation in percentage between the requested charge and the power wattage, sorted into temperature classes.

Temperature class [°C]	Total deviation [%]	Number of observations in temperature class
[-25, -20⟩	-	-
[-20, -15⟩	0.44	199
[-15, -10⟩	-	-
[-10, -5⟩	4.56	324
[-5, 0⟩	6.61	4000
[0, 5⟩	4.00	6372
[5, 10⟩	4.93	1527
[10, 15⟩	3.88	1183
[15, 20⟩	6.06	55
[20, 25]	-	-

7.2 Deviation trends across SoC classes

When deviations based on the SoC classes are divided by charger status, it can be observed from Table 7.2 that the highest deviation belongs to the [31, 41⟩ class. The lowest deviation is found in the [81, 91] class during charging at 2.70%. It can be observed that for every SoC class, there are more observations of discharging than of charging, and that out of the seven SoC classes, charging has the higher deviation in four of them.

Table 7.2: An overview of the deviation in each SoC class in percentages, between requested charge and power wattage. This table further shows the deviation distribution between the charger statuses.

SoC class [%]	Charger status	Total deviation [%]	Number of observations included in class
[21, 31⟩	Charging	5.48	85
	Discharging	2.89	161
[31, 41⟩	Charging	7.71	260
	Discharging	8.37	394
[41, 51⟩	Charging	6.42	610
	Discharging	5.44	760
[51, 61⟩	Charging	4.17	749
	Discharging	4.92	1063
[61, 71⟩	Charging	6.22	1009
	Discharging	4.38	1422
[71, 81⟩	Charging	4.55	1332
	Discharging	3.66	1904
[81, 91]	Charging	2.70	1757
	Discharging	6.81	2154

7.3 Vehicles' average RTE

Figure 7.1 shows the average RTE per vehicle. The figure shows that the lowest average RTE comes from the third Nissan Leaf 40 kWh, at 44.14%, with the second to last being 72.95% from the Nissan Leaf 40 kWh in Ås. The highest RTE value is 94.67% from the second Nissan Leaf 40 kWh. It can be observed from the figure that the average RTE can vary, even within cars of the same battery capacity. Within the 62 kWh, there is a gap of about 8.71% and 50.53% between the 40 kWh.

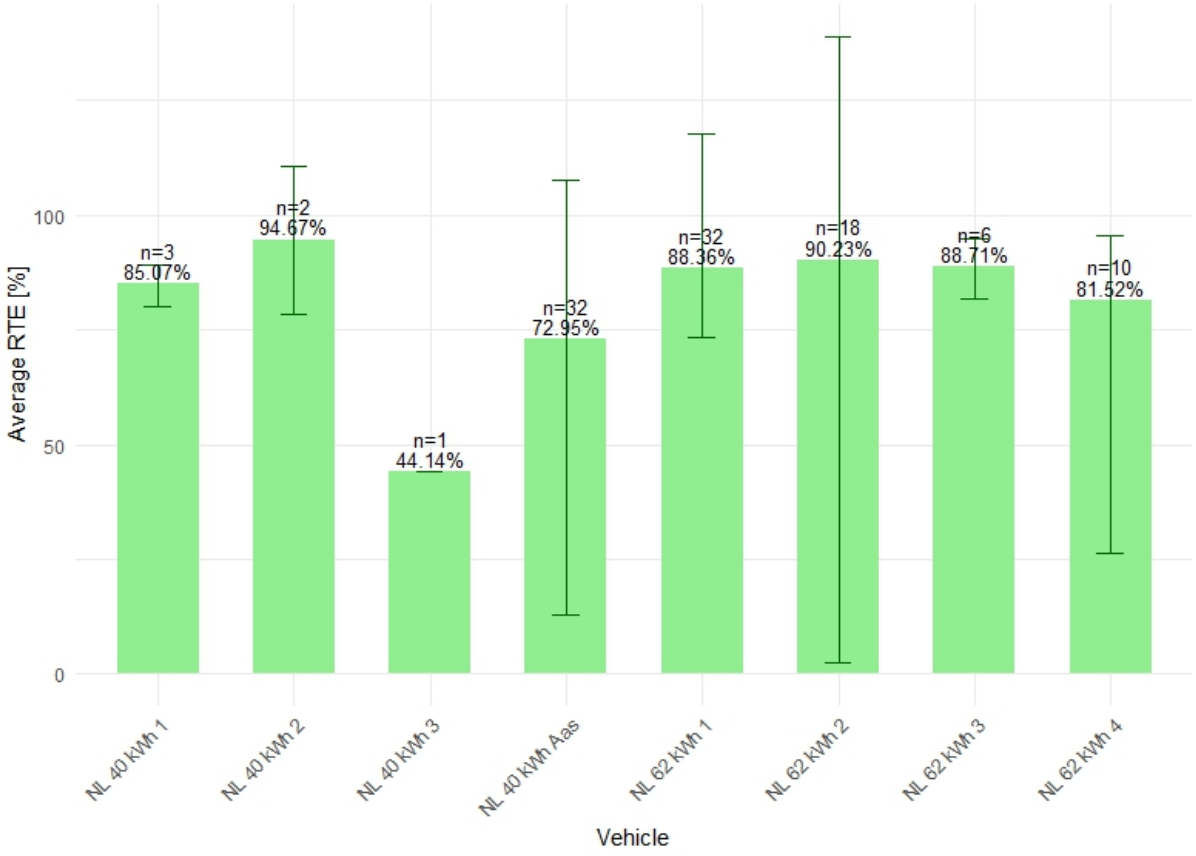


Figure 7.1: Diagram illustrating the average RTE of each vehicle. n shows the number of cycles identified in each vehicle, and the vertical lines represent the RTE range in the cycles in each vehicle.

The average RTE per battery capacity class was also calculated, see Table 7.3. This resulted in an average RTE of 87.86% for the 62 kWh. This is substantially higher than that of the 40 kWh vehicles with an average RTE of 74.29%.

Table 7.3: The different cars' average RTE, based on their battery capacity.

Battery capacity	RTE [%]
Average 40 kWh	74.29
Average 62 kWh	87.86

7.4 RTE trends across temperature classes

In Table 7.4, the results from the RTE calculations per temperature class are presented. This table shows that, ignoring the 0%, the lowest RTE is from the class [-10, -5⟩ at 69.6%, and the highest stems from the [15, 20⟩ class at 86.8%. This creates a gap of 17.2% from the highest to the lowest. It can also be observed that three different temperature classes do not display an RTE value. This is because no cycles had an average RTE temperature corresponding to these temperature classes.

Table 7.4: The average RTE per temperature class.

Temperature class [°C]	Average RTE [%]	Average Temperature [°C]	Number of cycles per class
[-25, -20⟩	-	-	0
[-20, -15⟩	78.52	-18.3	1
[-15, -10⟩	-	-	0
[-10, -5⟩	69.04	-7.4	3
[-5, 0⟩	84.11	-2.2	29
[0, 5⟩	82.49	1.8	46
[5, 10⟩	83.74	7.1	16
[10, 15⟩	84.29	12.6	8
[15, 20⟩	88.45	17.1	1
[20, 25]	-	-	0

The range of RTEs sorted into each class is further illustrated in Figure 7.2. It can be observed that with an increasing number of cycles in the class, the range of RTE also increases. Therefore, the classes with the smallest sample size also have the smallest RTE range.

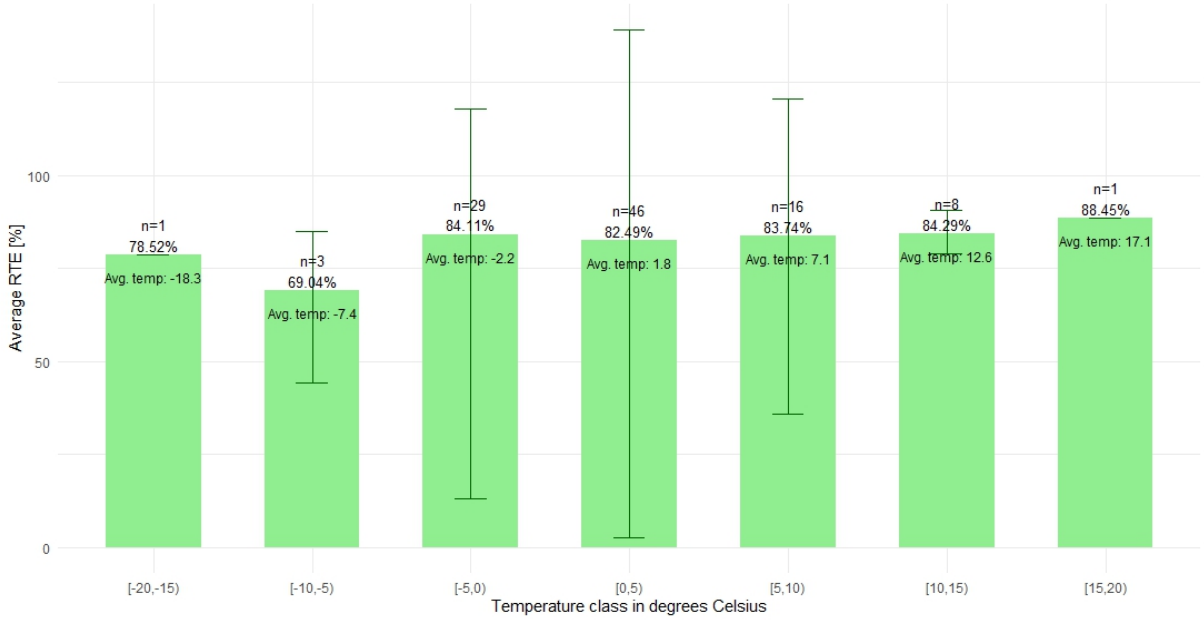


Figure 7.2: Diagram illustrating the average RTE per temperature class. n represents the number of cycles observed in each class and the vertical line represents the range of RTE values that are included in each class. The average temperature of the cycles included in each class is displayed in the middle of the bar.

7.5 RTE trends across SoC classes

The average RTE was calculated per SoC class to identify any trends between SoC and RTE. The results are shown in Table 7.3. The highest RTE can be found in the [31,41⟩ class at 93.59%. The lowest RTE at 79.27% occurs in the [61,71⟩ class.

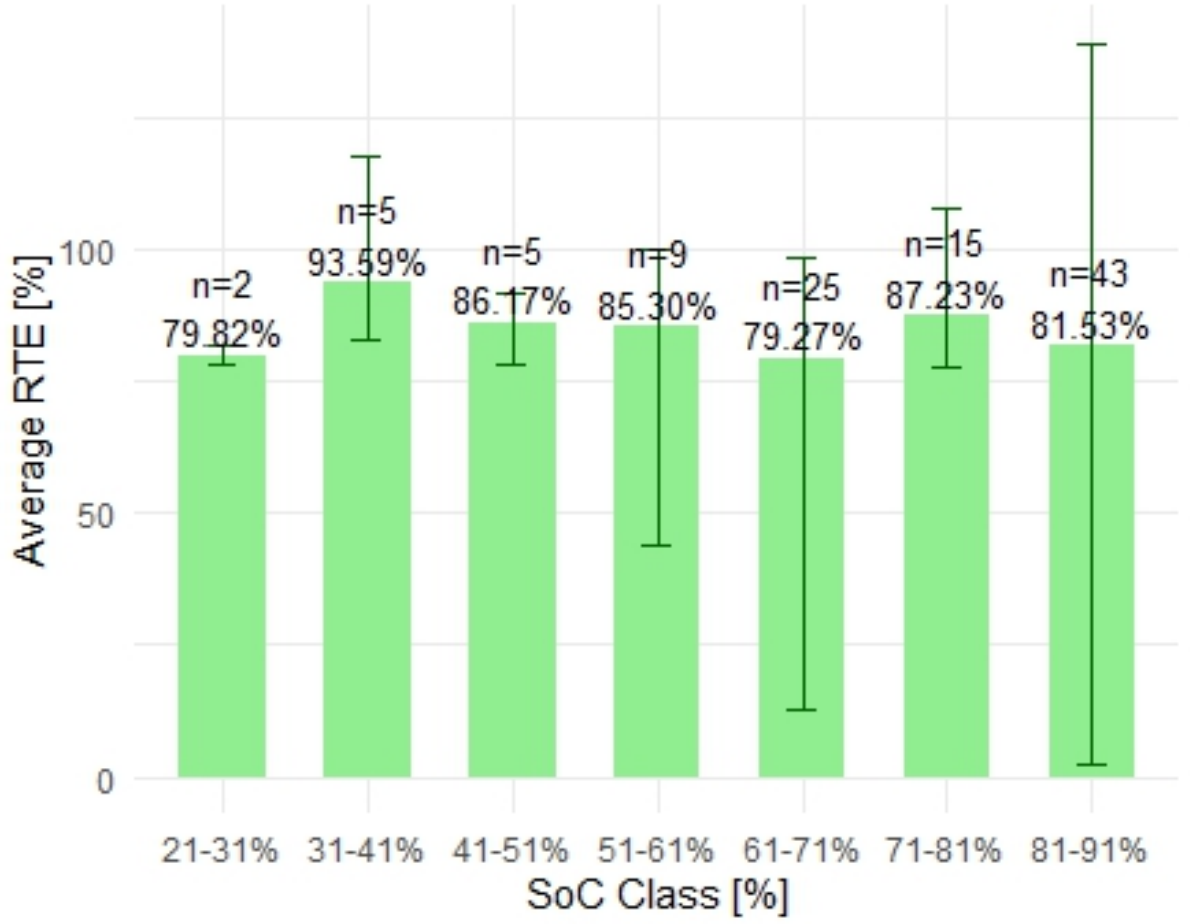


Figure 7.3: Diagram illustrating each SoC class's average RTE. The bar illustrates the average RTE for its respective SoC class. The average RTE, and number of cycles, n, sorted into the class are also listed above the bar. The thin line represents the range of RTEs sorted into the SoC class.

8 Discussion

This section aims to provide a deeper understanding by presenting and discussing the faults, results, and anomalies of this thesis. Where possible, the results will be compared to earlier research, and any anomalies or outliers will be identified and examined.

8.1 Dataset and filtering

Regarding the EV dataset, there are some months when no cycles are identified, such as January, July, and August. Only one cycle occurs in May and December. This lack of data, especially in July, August, January, and December, is likely one of the main reasons data in the highest and lowest temperature classes are missing, as these are some of the hottest and coldest months. Collecting more data would therefore contribute to a more complete picture.

The data sheet also had no fixed interval between observations; the time between observations in the cycles could vary by several minutes. Theoretically, this is not a problem as the added variable delta_t considers the time difference between observations. However, as the intervals between observations increase, the estimation of energy being delivered/received becomes less accurate. While this does introduce some uncertainty, most of the observations occur within one minute of each other. Any deviation between the calculated and actual energy likely has no significant effect on the RTE results, except for cycles with few observations.

The original filtering included removing observations with a SoC equal to or above 92%, as this was when the BMS system of the vehicle tended to take over the charging/discharging to protect the battery. Had these observations been included, the deviation results would have been highly affected by the BMS, as it would have strongly limited the current to protect the battery from stress and excessive voltage. Although this filtering slightly reduced the number of observations and removed some realistic behaviour, as the BMS would do this during real-world operations under high SoC, it also helped the study focus on the externally controlled energy inputs and outputs.

The dataset originally contained some observations with power wattage of more than ±60 kW. These observations were filtered out as this is exceptionally high. Furthermore, the maximum requested charge in this thesis was 7.36 kW, making ±60 kW a remarkable deviation. Additionally, when they occurred, it was during discharging, except for one instance. The requested charge was always around 2-4 kW. Since the BMS system in the vehicles was used to measure and retrieve the data, the cause of these high values is likely a measurement, communication, or software error that made the BMS report an event that did not occur.

Only observations containing a value in both SoC, requested charge and power wattage were kept to ensure the dataset's quality and avoid misleading information due to statistical noise. If any of those variables were an NA or a zero, the entire observation would be filtered out, even if the other two variables had a value. Although this reduced the amount of data to be analysed, the filtering is justified, as the interpretation of those observations would not be meaningful, because they do not provide a complete overview. Additionally, this ensured consistency in the dataset, making the data more reliable and simplifying the process.

8.2 Function variables and limitations

Several limitations were incorporated into the function to identify cycles and calculate the average RTE to improve the study's accuracy. The minimum number of charging and discharging observations for the hardcoded parameters should have been higher. A cycle with six measurements is too small a sample size to be considered realistic. Such cycles are likely to yield skewed results, especially if the cycle has measurements every minute, meaning the total cycle would be about six minutes, which is very short. Initially, the limit was created to avoid identifying cycles with a

significantly skewed number of measurements between charging and discharging, while keeping as much data as possible. In hindsight, the limit was not high enough, and although it stopped some of the data from becoming skewed, some still slipped through. Therefore, the limit should have been increased to make the cycles more robust, such as 20 observations, meaning a cycle would have to consist of at least 40 measurements. This would result in a better balance between robust cycles while still not disregarding too much data.

Another limit that was applied was the SoC transition. In a cycle, when the charger status changed and had a SoC jump of more than 10%, this cycle would be skipped. Similar to the time gap restraint, this was implemented to ensure that the second half of the cycle starts in a similar state, close to where the first half ended, but with a buffer in case the vehicle was temporarily unplugged or self-discharge occurred. As one of the main tasks of the thesis was to calculate RTE, it was essential to have cycles that were as complete as possible to minimise the loss of energy input and output. For example, if a vehicle is charging and used, it will return with a lower SoC than when it left. This will also mean that some energy put into the vehicle during charging has been lost. If the vehicle starts discharging upon return, this could later be identified as a cycle. When the RTE for that cycle is calculated, the registered energy output will be lower than it realistically was, resulting in a lower RTE. Therefore, adding the limit helped preserve the relevant data.

One of the most essential variables in the function is the delta_t. Due to how the data sheet was filtered, there were jumps in the time column, instead of showing the observations where the cycle paused, with observations displaying power and SoC values of zero. Delta_t, in addition to calculating energy, was used to prevent cycles from registering false charging periods by implementing the 30-minute stop. If an observation in a cycle had a delta_t of more than 1800 seconds, meaning more than 30 minutes passed from the current observation of energy exchange to the next, it was assumed that the vehicle stopped charging and only one second of that energy exchange occurred. Another observation would have appeared before 30 minutes had passed if there had been further energy exchange, this cut-off seemed appropriate to implement. In addition, delta_t was automatically set to one for every instance where the observation was the last in the cycle, as a time gap often followed this until the next cycle began.

The downside of implementing this variable and its limitations is that it has reduced the energy output or input, as the charge for the observations affected most likely lasted more than one second. However, the general limitation was implemented since there is uncertainty about how long the vehicles are charged or discharged after those observations. In addition, had delta_t and its limitations not been implemented, the RTE values would be skewed even worse, as the cycles would think that the breaks, where some are hours long, were also charging or discharging data, meaning a considerable amount of extra energy would be added into the calculations, that did not occur in reality. Therefore, these limitations have affected the results, more so the cycles with few observations, as more of the overall cycle has been disregarded. However, the more observations a cycle consists of, the less impact the delta_t shaving has on the overall effect. Had delta_t not been implemented, this would result in extremely skewed RTE results.

8.3 Non-ideal cycles

Multiple of the cycles resulted in an RTE of more than 100%. Whether or not these cycles should have been filtered out can be discussed. It seems impossible to have a higher energy output than input. Still, in these 100%+ cycles, it happens because the calculation is based on the SoC. Since the cycles do not start when the SoC is at zero and then additionally return to zero, but instead start at a random SoC value, the vehicle will, in reality, have more energy available than what is included in the input in that specific cycle. This is the reason why some of the identified cycles have reached an RTE of more than 100%.

For example, when looking into the second 62 kWh vehicle, which contains the highest RTE of all

the identified cycles at 139.09%, it can be observed that the energy output in this cycle is 122% of the energy input. One of the reasons is that the cycle consists of nine observations, making it very sensitive. In this specific cycle, there are five discharging observations, which discharge at about 5 kW and four charging observations that charge at a rate of about 6 kW. Additionally, delta_t was set to one, making the result more skewed, with technically three charging observations and five discharging observations. This results in the sum of energy output becoming even higher than energy input, causing the RTE of the cycle to be higher than what is realistically possible.

In the cycles below 50%, the same situations occur as in the cycles with RTE above 100%, just in the opposite direction. The lowest cycle originates from the second 62 kWh vehicle with an RTE of 2.51%. This cycle consists of seven observations in total, with three discharging observations and four charging observations. As this is a charging cycle, the last discharging observation is again affected by the limitation set to put delta_t to one in the final observation of a cycle. In addition, this cycle lasts for eight hours and 20 minutes, but out of these, seven hours and 48 minutes are spent on a break between the charger status switch, meaning the actual charging and discharging in this cycle only last 32 minutes. During this cycle, it charges at 7.3 kW but only discharges at 0.5 kW, causing an even greater divide between energy input and output, resulting in the extremely low RTE. Overall, in these extreme cycles, the same problems persist. Cycles with few observations make the cycle sensitive to outliers, which can significantly affect the RTE for that cycle.

8.4 Classification of deviation

Only observations from a cycle were used to determine how temperature can affect the deviation. All observations from the same cycle were grouped into the temperature class based on the cycles' average temperature. This decision was made because a cycle's temperature range is typically small, as illustrated by the overall average temperature range, at 2.9°C. Additionally, this would reduce the sensitivity of any random fluctuations. Still, it is essential to remember that although the overall average temperature range in a cycle is 2.9°C, cycles can still have a bigger temperature range.

Another challenge regarding how deviation was calculated for temperature classes is that significant temperature changes during a cycle may lead to the average temperature not being the best representation. Some cycles might have observations with temperatures outside the 5°C classes. This is especially relevant for longer cycles, as the longer a cycle is, the longer the ambient temperature have to change. Another aspect that might have affected the results is that temperature may not have a uniform effect on a cycle. For example, it might be more sensitive at the start due to colder temperatures. Patterns like these cannot be observed due to the sorting. Cycles, therefore, had to be manually reviewed to see if such effects could be observed.

8.5 Classification of RTE

It was ensured that only one RTE was contributed from each cycle to avoid statistical bias when calculating the average RTE per temperature class. Similarly to deviation, it was also decided to sort entire cycles into the same temperature class, as temperature ranges in charging and discharging cycles typically have a smaller range. Still, cycles can have a larger temperature range or a range that falls between two temperature classes, which can lead to inaccuracies.

The RTE was sorted into SoC classes by using one representative observation from each cycle to give each cycle equal statistical weight. Still, sorting SoC this way does have its disadvantages. For RTE specifically, the SoC classes might have benefited from being more than 10%, as although some cycles do have a SoC range of 10% or less, many cycles also have larger SoC ranges. This results in a higher chance of observations being wrongfully sorted due to the smaller classes. Had the classes had SoC ranges of, for example, 20%, this would mean overall in a cycle, fewer observations would be misplaced as it covers a bigger SoC range. Therefore, when observing SoC and its relations to RTE, it has to be interpreted carefully.

8.6 Ambient temperatures and their effect on deviation

It was assumed that deviation would increase in lower temperature classes. In colder temperatures, the battery's internal resistance increases, causing its ability to accept current to decrease. This results in a higher deviation. Meanwhile, the battery's internal resistance decreases for warmer temperatures, meaning these classes should result in a lower deviation. Simultaneously, if the temperatures become too high, the BMS might intervene, which could lead to higher deviations.

Table 7.1 indicates that deviation seems the most consistent in the 0°C to 15°C range, as the deviation is between 3.88% to 4.93% even with the large number of observations in these classes. In most classes, total deviation is less than 5%. This is expected as some deviation occurs in regular charging and discharging. The highest deviation occurs in the [-5,0⟩ class at 6.61%. This is higher than for other subzero classes, likely due to the small sample sizes of the other subzero classes. Although the deviation in this class is high, it is neither due to the temperature nor the SoC. In all cycles when requesting ±7 kW, the power requested was not accurately met. The most accurate delivery was 6.81 kW, and the most inaccurate delivery was -2.12 kW. The SoC ranges for these requests varied between 19-91% and temperatures between -15°C to 19°C. Although the deviation may vary with SoC and temperature, since the deviation is so high, it suggests that the deviation is due to the charger. The Wallbox Quasar 1 charger has a rated max power of 7.4 kW, meaning it cannot deliver or receive more than 7.4 kW. Still, real-world applications are often lower than this theoretical maximum. The pattern observed, therefore, suggests that the charger caps the output, leading to constant under-delivery.

Both the second-lowest and highest deviation is affected by the smaller sample sizes. The [-20,-15⟩ only contains four measurements with a power request of 7 kW, making deviation in this class extremely small. While in class [15,20⟩ all observations consist of requests of 6-7 kW, making the deviation high due to the charger capping the input and output.

8.7 SoC and its effect on deviation

When charging and discharging were split, see Table 7.2, the highest deviation for charging and discharging occurred in [31,41⟩ class. The deviation in this class is at 7.71% and 8.37% for charging and discharging, respectively. In the data, higher deviations mostly occur when power requests are ±7 kW or more. As an average amount of 7 kW requests were observed in the SoC class, this alone is likely not the reason for the high deviation. A couple of observations that seem to be the first observation after status switch, where delivered power is often around half of what is requested, if not less, were also observed in this class. This might have been caused by the BMS, as when the charger status switches, the BMS hysteresis causes a delay in the response. In this period, if requested charge increases or decreases, then the actual power remains at the same level it was, while requested charge increases/decreases from that power level, temporarily increasing the deviation. This, together with the ±7 kW might be the reason for the high deviation in the class.

For charging, the deviation is higher in the lower SoC classes and lower in high SoC classes, as was seen by the total deviation for [21,31⟩ and [81,91] with a total deviation of 5.48% and 2.70% respectively. These results reflect how the BMS imposes limitations to protect the battery. Under higher SoCs, the battery is close to being fully charged, and internal resistance is increased. The BMS limits the current to avoid overcharging. With the strict limitations applied, smaller power requests are given to the battery. These are easier for the battery to handle, resulting in the battery receiving smaller power amounts, causing less deviation overall.

The opposite occurs in low SoCs as was seen by [91,81] and [21,31⟩ with a total deviation of 6.81% and 2.89% respectively. During low SoCs the battery will be close to being fully discharged, meaning it cannot handle high power requests. This causes the BMS to intervene to avoid over-discharging by limiting the power at which the battery is discharged. This reduction in power requests is easier for the battery to fulfil, causing lower deviation. However, when discharging during higher SoCs, the

BMS will not intervene as the battery can handle discharging, causing higher deviations.

8.8 Average RTE

The lowest average RTE in Figure 7.1 was at 44.14% from the third 40 kWh. This vehicle identified only one cycle with 27 measurements in total. There is a clear difference between the requested charge under charging and discharging, respectively 1-3 kW, and 0.5-0.8 kW. This caused the cycle to have an imbalance between the power delivered and received, resulting in low amounts of energy output by the time the SoC reached the same SoC the cycle started with.

The highest average RTE was found in the second 40 kWh vehicle, at 94.67%. This vehicle had two identified cycles, one with an RTE of 78.52% and one with an RTE of 110.83%. This result is affected by the 110.83% RTE, which occurs because of the vehicle's stored energy. Had the sample size been bigger, the result would likely be closer to the first cycle of 78.52%.

There was a big variation between the different vehicles' average RTE, ranging from 44.14% to 94.67%, see Figure 7.1. As the second 62 kWh vehicle is also affected by the cycles with an RTE higher than 100%, containing four of them, the highest unaffected average RTE would be the third 62 kWh vehicle with an average RTE of 88.71%. This would be more in line with the findings of Schram et al. [3], whose highest average RTE was at 87.00%. Additionally, Apostolaki-Iosifidou, Kempton, and Codani [40] also studied losses during charging and discharging, and arrived at an RTE of about 70% [39, 40]. This study has a slightly higher average RTE because it is a DC-to-DC study, while Schram et al. [3] and Apostolaki-Iosifidou, Codani, and Kempton [39] studied AC-to-AC losses, meaning they also accounted for losses between the charger and the grid, resulting in a lower RTE. However, as the system boundary for this thesis was restricted to the battery, a slightly higher RTE was expected. This may imply room for improvement in the cycle identification function.

8.9 Average RTE per battery capacity

The results in Table 7.3 imply that RTE may improve with bigger battery capacity. The C-rate calculation is based on the vehicle's battery capacity. If the same amount of current is sent to a 40 kWh and a 62 kWh vehicle, the 62 kWh vehicle will divide this current across more cells. This causes less internal resistance, resulting in less energy wasted as heat, causing a higher RTE. For the 62 kWh vehicle, the C-rate is expected to be lower than that of the 40 kWh vehicle. A lower C-rate indicates a longer charging/discharging time, which helps extend the battery's lifespan.

When inspecting further, it is evident that the average for the 40 kWh vehicles is affected by the RTEs from the second and third vehicle, which were caused by the small sample size in each vehicle. However, although the average is affected, the pattern may still be relevant, but is expected not to have as big of an effect as Table 7.3 implies.

8.10 Effect of ambient temperature on RTE

It is clear from Table 7.4 and Figure 7.2 that temperature affects the RTE. In colder temperatures, the RTE is noticeably lower than in warmer temperatures. Even though there are few cycles sorted into temperature classes lower than -5°C, these findings align with those of Schram et al. [3], whose RTE results also showed a decrease in RTE at lower ambient temperatures.

When a battery is charged in temperatures below 0°C, the rate of the chemical reaction is slowed down because the electrolyte becomes more viscous, which affects the dispersion of ions. When the ions take longer to travel, it temporarily affects the battery's ability to deliver or accept current. This increases the battery's internal resistance and heat generation, lowering RTEs. The results are mostly consistent with this, where both [-20, 15⟩, and [-10, -5⟩ on average had lower RTEs than the other temperature classes.

The RTE marginally increases with the temperature for each temperature class above zero degrees. Since the optimal operating temperature for lithium-ion batteries is around 15-35°C, it was expected that the highest temperature class included in this study would also have the highest RTE. In this study, this would have been the [20, 25⟩ class, but due to a lack of data, no cycles were sorted into this class. Therefore, the second highest temperature class, [15, 20⟩ class with one cycle ended up being the class with the highest RTE at 88.45%. This corresponds to the underlying theory, as in the optimal operating temperatures for lithium-ion batteries, the chemical reactions inside the battery become more efficient. This is because the electrolyte becomes less viscous at warmer temperatures, making it easier for the lithium ions to move through the electrolyte. This makes charging and discharging more efficient, resulting in higher RTEs.

Still, it is essential to acknowledge that the battery's temperature in colder and warmer temperatures will likely differ and will most often be higher than the ambient temperature. This is because, even under optimal conditions, batteries naturally generate heat as the reactions lose energy due to resistance that occurs inside. Additionally, since Nissan Leafs do not have an active cooling system and rely on passive cooling, most of this heat will be trapped inside the battery. This heat will also build up with each cycle performed. Therefore, it is ideal that most of the cycles have a break between the charging and discharging phases in each cycle, as this lets the battery cool down.

The BMS can try to prevent damage caused by temperature by limiting the current or completely stopping the charging or discharging. However, temperatures between 15°C and 25°C, the BMS is unlikely to intervene due to the temperature. This is reflected in the cycles as the requested charge is overall met with an insignificant deviation, except where the requested charge is 7 kW or more. This is likely due to the charger's limitations.

For the temperatures that are above 0°C, but less than 15°C, the RTE stays very consistent between 82.49% and 84.29%. Even though these RTEs occur in temperatures below the optimal operating temperature for lithium-ion batteries, the results suggest that the battery can perform satisfactory even outside its optimal operating temperatures, but that temperatures below 0°C might have the most significant impact on RTE and temperature.

An interesting finding in Table 7.4 is that RTE was higher by 1.62% in the temperature class [-5, 0⟩ when compared to temperature class [0, 5⟩. This does not follow the pattern represented in the rest of the table and therefore had to be investigated further. When investigating, it was discovered that the [0, 5⟩ class included four cycles with an RTE below 50%, and five with an RTE above 100%. Meanwhile, for the [-5, 0⟩ class, double as many cycles with an RTE of above 100% were observed as cycles with an RTE below 50% at two and four, respectively. The RTE range of temperature in classes [-5,0⟩ and [0, 5⟩ was respectively 13.22% to 117.75% and 2.51% to 139.09%. This skewed distribution of extreme RTEs in [-5, 0⟩ is, with high probability, the reason the class has a higher average RTE when compared with the [0, 5⟩ class.

8.11 SoCs impact on RTE

In SoC class [81-91] the average RTE was 81.53%, which is lower than most of the RTEs in mid-range SoCs. Although the [81-91] SoC class has the most cycles, the overall RTE range of the class includes both the cycle with the lowest and highest RTE. Still, this result does match the expectations of how battery performance is affected by higher SoCs. When the SoC rises, so does the voltage. When the voltage reaches its threshold, the charger has to change from keeping the current constant to keeping the voltage constant to avoid overcharging. The allowed current is gradually reduced so the battery can continue charging, since the voltage threshold cannot be exceeded. This causes a decrease in the power that is delivered, but allows the battery to continue charging. Simultaneously, the lithium ions move from the cathode to the anode, causing intercalation at the anode. Even when the anode is full of ions, more lithium ions continually try to insert themselves into the anode with increasing difficulty, causing internal resistance to increase. This results in more energy transforming into heat.

The overall result is that at higher SOCs more of the energy that is delivered is typically lost per unit of energy that is stored, which contributes to a lower RTE.

At the lowest SoC class, the average RTE is 79.82%. Though this class has a limited sample size, both cycles are similar in RTE at 77.99% and 81.63%. The RTE of this SoC class also follows the known pattern between battery performance and lower SoCs. During lower SoCs, the voltage drops and approaches the minimum voltage threshold. The BMS ensures that the session stays within the SOA by reducing current to avoid over-discharging. Simultaneously, deintercalation occurs in the anode while intercalation occurs in the cathode. As the amount of lithium ions decreases in the anode, it becomes increasingly challenging for the remaining lithium ions to leave the anode. This contributes to increasing the internal resistance, further causing an increase in heat generation, leading to more energy being lost as heat and reducing the energy output, ultimately affecting the RTE.

Schram et al. [3], found that the RTEs were marginally lower at low and high SoCs, compared to mid-ranged SoCs. Figure 7.3, shows that the results are mostly consistent with these findings, as both the highest and lowest SoC class also contains the lowest average RTE, except for [61, 71⟩. The range of RTEs that are included affects the results in most SoC classes. The [31, 41⟩ and [71, 81⟩ classes have obtained a higher average RTE due to the range, while SoC classes [51, 61⟩, [61, 71⟩ and [81,91] have obtained a lower average RTE, to varying degrees. However, seeing how the average RTE has been affected, it is likely that the same overall pattern would occur with a more accurate RTE range in each class, where the lower RTEs occur in the lowest and highest SoC class.

9 Conclusion

This thesis has investigated real-world V2G data, intending to find the system losses and what affects them. To achieve this, variables such as deviation and RTE were calculated, and it was also investigated how these variables are affected by temperature and SoC. Across the 104 identified cycles, an average RTE of 82.90% was achieved, resulting in average DC-to-DC losses at 17.10%. This is higher than the average loss found by Schram et al. [3], who found losses at 13%, but less than the losses found by Apostolaki-Iosifidou, Codani, and Kempton [39], who found total losses of 29%.

Although deviations do not directly affect the RTE results, they give insight into system performance, as they can represent how accurate the V2G system is. If the system constantly showed high deviations, meaning the BMS or charger often applies constraints to the V2G process, it would be hard to use V2G in real-world applications, since power requests would constantly have to be over- or underestimated, making V2G less reliable. The deviation analysis showed that under typical operating conditions, power deviations are expected to be around 5%, with charging on average having the higher deviations. There were some higher deviations around [31-41⟩ SoC class, however, since deviations overall stayed low, this shows that the system is dependable for use in real-world V2G operations.

The RTE analysis showed that in mid-range SoC intervals, the RTE was higher than in the end intervals. This behaviour suggests internal resistance is decreased in the mid-range SoCs, improving the energy output. Temperature also affected the RTE, showing that it decreased in sub-zero temperatures. Nevertheless, the coldest cycle with an average temperature of -18.3°C had an RTE of 78.52% which is just under a 4% difference from the average RTE at [0,5⟩. Although this value is from a singular cycle and therefore cannot be considered statistically significant, it may indicate that V2G can be viable even in the coldest winter months in southern parts of Norway. This is an important factor, as this is when the Norwegian grid has its highest loads, due to increased use of electric heating. Had RTEs been highly reduced under these temperatures, V2G would not be viable during the coldest months.

Overall, the findings show that V2G performance is affected by both SoC and temperature. Nevertheless, the DC-to-DC V2G system performed within the expected RTE ranges found in earlier studies, even with the extreme cycles being included. This shows the practical and technical feasibility of V2G, making it a promising option for delivering to the grid in periods of high demand.

10 Further work

As both the system boundary and the variables were limited, this leaves a lot of options for what could be explored further in the future. This section will therefore discuss any processes or methods that could be improved or added for future research on the RTE of V2G.

10.1 Further EV data collection

The most useful adjustment that can be made in future research is to collect more data. As observed in this thesis, there were temperature classes where no observations were retrieved, particularly for the coldest and hottest temperatures. From the data collection, no cycles were registered in January, May, July, and August, and in addition, only one cycle was registered in December. As these are the months with temperatures that are most likely to impact V2G efficiency significantly, more data is crucial to reflect a more comprehensive picture.

10.2 AC-to-AC

To account for all losses during the full V2G session, it is recommended that future studies focus on finding the AC-to-AC losses, as that takes the whole system into account. This would include inverter losses for both charging and discharging and any transmission losses that occur as the current flows through the grid. This approach would give a more comprehensive overview, showing how much energy input returns to the grid as energy output.

10.3 Predetermined SoC classes

While this thesis used real-world cycles with a lot of variation, both in temperature and SoC, it also introduced challenges when connecting observed patterns to a specific variable. Therefore, in future works, it is recommended to use predetermined SoC classes, for example, 10%-30%, 30%-50% and 50%-80%, and continuously perform the set classes throughout the year in different temperature conditions. Implementing this would make it easier to spot if trends are due to SoC or temperature in future studies.

10.4 State of health tests

Although this study did not investigate how battery degradation affects the V2G process, Sagaria, Kam, and Boström [25] found that the total degradation rate would increase with 9% to 14% over 10 years. As it is suggested that battery degradation significantly impacts the V2G process, it is recommended that future studies regularly perform tests that check the state of health of the battery. This would include checking the total energy delivered from a complete charge cycle, from low to high SoC and comparing that to the battery's rated capacity. This should be done regularly, with how often depending on whether the vehicles are used between V2G sessions or are stationary. If vehicles are used between V2G sessions, measuring both before and after a session is essential, so degradation stemming from driving is not classified as V2G-related degradation. Including this in future V2G studies would help understand how V2G influences the battery's durability.

References

[1] *Forventer kraftig vekst i kraftforbruket, avhengig av nett og mer kraftproduksjon*. no. Jan. 19, 2023. URL: https://www.statnett.no/om-statnett/nyheter-og-pressemeldinger/nyhetsarkiv-2023/forventer-kraftig-vekst-i-kraftforbruket-avhengig-av-nett-og-mer-kraftproduksjon/ (visited on 05/09/2025).

[2] Fornybar Norge. *Kraftprodusentene produserer fornybar strøm*. no. URL: https://www.fornybarnorge.no/stromguiden/hvor-mye-du-betaler-for-strom-avhenger-av-flere-faktorer/kraftprodusentene-produserer-fornybar-strom/ (visited on 03/07/2025).

[3] Wouter Schram et al. "Empirical Evaluation of V2G Round-trip Efficiency". In: *2020 International Conference on Smart Energy Systems and Technologies (SEST)*. Sept. 2020, pp. 1–6. DOI: 10.1109/SEST48500.2020.9203459. URL: https://ieeexplore.ieee.org/document/9203459 (visited on 03/13/2025).

[4] NVE. *The Norwegian power system. Grid connection and licensing*. en. Aug. 2018. URL: https://publikasjoner.nve.no/faktaark/2018/faktaark2018_03.pdf (visited on 01/24/2025).

[5] Energy Facts Norway. *The electricity grid*. en. Jan. 15, 2024. URL: https://energifaktanorge.no/en/norsk-energiforsyning/kraftnett/ (visited on 01/26/2025).

[6] Energy Facts Norway. *Electricity production*. en. May 16, 2024. URL: https://energifaktanorge.no/en/norsk-energiforsyning/kraftproduksjon/ (visited on 01/26/2025).

[7] Åsmund M Frengstad. *Guide: Here's What You Need To Know About Vehicle-to-Grid (V2G)*. en. n.d. URL: https://www.current.eco/resources/articles/vehicle-to-grid-v2g (visited on 01/16/2025).

[8] Current. *ISO 15118*. en. n.d. URL: https://www.current.eco/ev-glossary/iso15118 (visited on 01/24/2025).

[9] Exro. *Load Shifting: What Is It and How Does It Work?* en. URL: https://www.exro.com/industry-insights/load-shifting (visited on 02/23/2025).

[10] Elbilgrossisten. *Ladeguiden: Type 1 og type 2 plugg / kontakt - Elbilgrossisten*. nb. n.d. URL: https://www.elbilgrossisten.no/pages/ladeguiden-type1-type2 (visited on 02/18/2025).

[11] Elbilgrossisten. *Ladeguiden - What charging power can I get? — Elbilgrossisten*. en. URL: https://www.elbilgrossisten.no/en/pages/ladeguiden-hvilken-ladeeffekten-far-jeg (visited on 02/21/2025).

[12] tompitts. *Type 1 and Type 2 EV charging - what's the difference?* en-GB. Apr. 2024. URL: https://wepoweryourcar.com/type-1-and-type-2-ev-charging-whats-the-difference/ (visited on 02/19/2025).

[13] Elbilgrossisten. *Ladeguiden: Hurtiglading av elbil - Elbilgrossisten*. nb. n.d. URL: https://www.elbilgrossisten.no/pages/ladeguiden-hurtiglading (visited on 02/20/2025).

[14] Ladestasjoner. *Kontakttyper for elbil | Alt om lading av elbil | Ladestasjoner.no*. no. n.d. URL: https://www.ladestasjoner.no/lading/kontakttyper/ (visited on 02/18/2025).

[15] CHAdeMO. *FAQ | CHAdeMO*. en-US. URL: https://www.chademo.com/faq (visited on 02/22/2025).

[16] tagonline. *What is the CHAdeMO charging standard and how does it differ from others like CCS or SAE J1772?* en-US. Nov. 1, 2023. URL: https://www.electronicofficesystems.com/2023/11/01/what-is-the-chademo-charging-standard-and-how-does-it-differ-from-others-like-ccs-or-sae-j1772/ (visited on 02/22/2025).

[17] U.S Department of Transportation. *Charger Types and Speeds | US Department of Transportation*. en. Jan. 31, 2025. URL: https://www.transportation.gov/rural/ev/toolkit/ev-basics/charging-speeds (visited on 02/18/2025).

[18] U.S Department of Energy. *Alternative Fuels Data Center: Batteries for Electric Vehicles*. en. n.d. URL: https://afdc.energy.gov/vehicles/electric-batteries (visited on 01/17/2025).

[19] Shuai Ma et al. "Temperature effect and thermal impact in lithium-ion batteries: A review". In: *Progress in Natural Science: Materials International* 28.6 (Dec. 2018), pp. 653–666. ISSN: 1002-

0071. DOI: 10.1016/j.pnsc.2018.11.002. URL: https://www.sciencedirect.com/science/article/pii/S1002007118307536 (visited on 04/25/2025).

[20] Knut Hofstad. *litiumionbatteri*. no. Oct. 1, 2024. URL: https://snl.no/litiumionbatteri (visited on 01/17/2025).

[21] U.S Department of Energy. *How Lithium-ion Batteries Work*. en. Feb. 28, 2023. URL: https://www.energy.gov/energysaver/articles/how-lithium-ion-batteries-work (visited on 01/17/2025).

[22] Research Institutes. *What Are Lithium-Ion Batteries? - UL Research Institutes*. en-US. Section: Research Updates. Sept. 14, 2021. URL: https://ul.org/research-updates/what-are-lithium-ion-batteries/ (visited on 01/24/2025).

[23] Exro. *Battery Degradation: Maximizing Battery Life & Performance*. en. n.d. URL: https://www.exro.com/industry-insights/battery-degradation-explained (visited on 02/11/2025).

[24] ECO STOR. *Understanding Battery Degradation: Causes, Effects, and Solutions*. en. May 14, 2024. URL: https://www.eco-stor.com/en/resources/blog/understanding-battery-degradation-causes-effects-and-solutions (visited on 02/11/2025).

[25] Shemin Sagaria, Mart van der Kam, and Tobias Boström. "Vehicle-to-grid impact on battery degradation and estimation of V2G economic compensation". In: *Applied Energy* 377 (Jan. 1, 2025), p. 124546. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2024.124546. URL: https://www.sciencedirect.com/science/article/pii/S0306261924019299 (visited on 04/17/2025).

[26] Jelena Popovic. "10 - Nanostructured alkali and alkaline earth metal interfaces for high-energy batteries". In: *Frontiers of Nanoscience*. Ed. by Rinaldo Raccichini and Ulderico Ulissi. Vol. 19. Nanomaterials for Electrochemical Energy Storage. Elsevier, Jan. 2021, pp. 327–359. ISBN: 9780128214343. DOI: 10.1016/B978-0-12-821434-3.00007-7. URL: https://www.sciencedirect.com/science/article/pii/B9780128214343000077 (visited on 04/17/2025).

[27] Thomas R. B. Grandjean et al. "Accelerated Internal Resistance Measurements of Lithium-Ion Cells to Support Future End-of-Life Strategies for Electric Vehicles". en. In: *Batteries* 4.4 (Apr. 10, 2018). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, p. 49. ISSN: 2313-0105. DOI: 10.3390/batteries4040049. URL: https://www.mdpi.com/2313-0105/4/4/49 (visited on 05/09/2025).

[28] David M. Stewart. *What Stress Means for Batteries*. en. n.d. URL: https://www.energyfrontier.us/content/what-stress-means-batteries (visited on 02/11/2025).

[29] Yuqing Chen et al. "A review of lithium-ion battery safety concerns: The issues, strategies, and testing standards". In: *Journal of Energy Chemistry* 59 (Aug. 2021), pp. 83–99. ISSN: 2095-4956. DOI: 10.1016/j.jechem.2020.10.017. URL: https://www.sciencedirect.com/science/article/pii/S2095495620307075 (visited on 05/09/2025).

[30] Yujie Wang, Xingchen Zhang, and Zonghai Chen. "Low temperature preheating techniques for Lithium-ion batteries: Recent advances and future challenges". In: *Applied Energy* 313 (May 1, 2022), p. 118832. ISSN: 0306-2619. DOI: 10.1016/j.apenergy.2022.118832. URL: https://www.sciencedirect.com/science/article/pii/S0306261922002732 (visited on 05/10/2025).

[31] Synopsys. *What is a Battery Management System (BMS)? – How it Works | Synopsys*. en. n.d. URL: https://www.synopsys.com/glossary/what-is-a-battery-management-system.html (visited on 02/15/2025).

[32] Greg Kong Kelly & Zimmer. *Active Balancing: How It Works and Its Advantages | Article | MPS*. en. July 11, 2024. URL: https://www.monolithicpower.com/en/learning/resources/active-balancing-how-it-works-and-its-advantages?srsltid=AfmBOookcqf6V2nlmVIUt4f-vPjjr54Smvlc1WQ_gw2rRyVApIRNS348 (visited on 02/21/2025).

[33] Battery Inside. *Battery Glossary – CC/CV Charging* -. en-US. Section: Tech. Nov. 23, 2022. URL: https://inside.lgensol.com/en/2022/11/battery-glossary-cc-cv-charging/ (visited on 05/09/2025).

[34] W. Waag and D.U. Sauer. "SECONDARY BATTERIES – LEAD– ACID SYSTEMS | State-of-Charge/Health". In: *Encyclopedia of Electrochemical Power Sources*. Ed. by Jürgen Garche. Amsterdam: Elsevier, Dec. 8, 2009, pp. 793–804. ISBN: 978-0-444-52745-5. DOI: https://doi.org/10.1016/B978-044452745-5.00149-0. URL: https://www.sciencedirect.com/science/article/pii/B9780444527455001490 (visited on 01/16/2025).

[35] gridX. *What is State of Charge? – gridX*. en. Oct. 23, 2024. URL: https://www.gridx.ai/knowledge/what-is-state-of-charge (visited on 01/20/2025).

[36] Joshua Hebditch. *What is round trip efficiency in battery storage?* en-GB. Jan. 30, 2024. URL: https://givenergy.co.uk/what-is-round-trip-efficiency-in-battery-storage/ (visited on 01/21/2025).

[37] Eco Stor. *What is a c-rate*. en. n.d. URL: https://www.eco-stor.com/resources/ecopedia/c-rate (visited on 01/19/2025).

[38] Battery University. *BU-402: What Is C-rate?* en. Oct. 25, 2021. URL: https://batteryuniversity.com/article/bu-402-what-is-c-rate (visited on 01/19/2025).

[39] Elpiniki Apostolaki-Iosifidou, Paul Codani, and Willett Kempton. "Measurement of power loss during electric vehicle charging and discharging". In: *Energy* 127 (May 15, 2017), pp. 730–742. ISSN: 0360-5442. DOI: 10.1016/j.energy.2017.03.015. URL: https://www.sciencedirect.com/science/article/pii/S0360544217303730 (visited on 03/14/2025).

[40] Elpiniki Apostolaki-Iosifidou, Willett Kempton, and Paul Codani. "Reply to Shirazi and Sachs comments on "Measurement of Power Loss During Electric Vehicle Charging and Discharging"". In: *Energy* 142 (Jan. 1, 2018), pp. 1142–1143. ISSN: 0360-5442. DOI: 10.1016/j.energy.2017.10.080. URL: https://www.sciencedirect.com/science/article/pii/S0360544217317851 (visited on 03/14/2025).

[41] Carplug. *Wallbox QUASAR - CHAdeMO 7,4kW - WiFi & Bluetooth - Carplug*. en. URL: https://www.carplug.eu/wallbox-quasar-chademo-74kw-wifi-bluetooth (visited on 05/05/2025).

[42] W3schools. *SQL Tutorial*. en-US. n.d. URL: https://www.w3schools.com/sql/ (visited on 02/07/2025).

[43] Modbus tools. *Modbus Protocol*. en. n.d. URL: https://www.modbustools.com/modbus.html (visited on 02/07/2025).

[44] Norsk Klimaservicesenter. *Observasjoner og værstatistikk - Seklima*. no. n.d. URL: https://seklima.met.no/ (visited on 01/31/2025).

A Appendix: The full Rstudio code

```
#Master - V2G

#installing packages that may be needed:
require(tidyverse)
require(ggplot2)
require(readr)
require(readxl)
require(dplyr)
library(lubridate)



#Loading in and cleansing/filtering the main data:
################################################################################

#Loading the main data into Rstudio
ev_data <- read_excel(file.path("C:/Users/ylvai/OneDrive/Dokumenter/NMBU -
    Master - Fornybar Energi/4. Semester/M30 - Masteroppgave/Data","All_data_
    Jan2025.xlsx")) #read excel as it is an excel file and then the path in my
    computer to help R find the file

#As duplicates were observed in the original file (exact same ev_log_id), any
    duplicates had to be filtered out
ev_data <- ev_data %>%
  distinct(ev_log_id, .keep_all = TRUE) #Making it only keep 1 of each ev_log_id

#Making a smaller dataset with only data that has "actual" information, all rows
     with 0 included in soc, power_wattage and requested_charge is removed since
     if SoC
new_dataset <- ev_data %>%
  dplyr::filter(soc != 0 & power_wattage != 0 & !is.na(requested_charge) &
      requested_charge != 0) #Makes sure only the rows where SoC, power_wattage
      or requested_charge is NOT 0, and where requested_charge is NOT missing is
       kept



#Inserting time and temperature into the dataset:
################################################################################
#First had to load in the temperature data, and remember for Gardermoen it is an
     30-minute "resolution" but hourly in Aas
temp_gardermoen <- read.csv2(file.path("C:/Users/ylvai/OneDrive/Dokumenter/NMBU
    - Master - Fornybar Energi/4. Semester/M30 - Masteroppgave/Data","
    Airtemperature-Gardermoen-30min.csv"))
temp_aas <- read.csv2(file.path("C:/Users/ylvai/OneDrive/Dokumenter/NMBU -
    Master - Fornybar Energi/4. Semester/M30 - Masteroppgave/Data","
    Airtemperature-aas-hourly.csv"))

#Changes the new_dataset by including a new column and use POSIXct so Rstudio
    recognizes it as date-time value
new_dataset <- new_dataset %>%
  mutate(ev_datetime = as.POSIXct(ev_datetime, format = "%Y-%m-%d %H:%M:%S", tz
      = "CET")) #Year-month-date Hour-minute-seconds, Central European Time

#Again making it so Rstudio understands that it the column Tid is actual time
    and not random numbers
temp_aas <- temp_aas %>%
  mutate(Tid = as.POSIXct(Tid, format = "%d.%m.%Y %H:%M", tz = "CET"))  #In this
       dataset the readings are hourly, Year-month-date Hour-minute-seconds,
      Central European Time

```

```
temp_gardermoen <- temp_gardermoen %>%
  mutate(Tid = as.POSIXct(Tid, format = "%d.%m.%Y %H:%M", tz = "CET"))  #In this dataset the readings happen on a 30-minute basis, Year-month-date Hour-minute-seconds, Central European Time

#Matching the ambient temeprature to the closes timestamp in the dataset:
match_closest_temp <- function(dataset, temp_data) {
  dataset <- dataset %>%
    rowwise() %>% #For each row
    mutate(temp = temp_data$Lufttemperatur[which.min(abs(difftime(ev_datetime, temp_data$Tid, units = "mins")))]) #Find the closest timestamp in the temperature dataset and match it to the closest timestamp in the car dataset, then add inn the temperature belonging to that timestamp in a new column called temp.

  return(dataset) #Returns the modified dataset
}

#Separates the dataset by the charger, since the chargers have different locations, and matches the corresponding charger to the corresponding temperature from the temperature sheet at the right time.
new_dataset_aas <- new_dataset %>% filter(charger_id == 0) %>% match_closest_temp(temp_aas)  #Hourly temperature observations, aas
new_dataset_gardermoen <- new_dataset %>% filter(charger_id %in% 1:4) %>% match_closest_temp(temp_gardermoen)  #Temperature data observed every 30 min, Gardermoen

#Takes the two datasets and puts them back together, now with the "closest" air temperature included
new_dataset <- bind_rows(new_dataset_aas, new_dataset_gardermoen)



#More filtering/cleansing
################################################################################

#Converts NULL to NA, which Rstudio recognizes
new_dataset <- new_dataset %>%
  mutate(requested_charge = as.numeric(ifelse(requested_charge == "NULL", NA, requested_charge)))

#Only keeos rows where requested_charge is NOT NA or zero
new_dataset <- new_dataset %>%
  filter(!is.na(requested_charge) & requested_charge != 0)

#Noticed that some rows have +/- 60 000 watts on power_wattage which is extremely high so this needs to be investigated
summary(new_dataset$power_wattage)

#Observed the min is -65 536 and max is 61 540. This is way too high and needs to be fixed as max requested charge is 7300 W. Can see from the dataset when arranging power_wattage, that -6438 is the second lowest and 6809 is the second highest before it jumps to +-60 000+
#Therefore it is decided to limit the datasets power_wattage to be between +/- 7000 watts
new_dataset <- new_dataset %>%
  filter(power_wattage >= -7000 & power_wattage <= 7000) #Only keeps rows where power_wattage is between -7000 to +7000

#It was later decided to remove rows where soc was higher or equal to 92% as that is when the BMS took over and the requests were ignored.
new_dataset <- new_dataset %>%
  filter(soc < 92)
```

```
86
87
88
89 #Making the separate datasets for each car
90 #
    ##############################################################################################
91 #Visits from the same car are sorted into the same dataset
92
93 #Aas vehicle
94 #First have to define the start and end date for the vehicle
95 start_date_40_aas <- as.POSIXct("2023-02-07 16:00:00", tz = "CET") #definition
       of the start date, the day the car got there
96 end_date_40_aas <- as.POSIXct("2024-04-19 03:02:00", tz = "CET") #definition of
       the end date, the day the car left
97 #Then have to filter the data after those dates
98 nissan_leaf_40kwh_aas <- new_dataset %>%
99   filter(ev_datetime >= start_date_40_aas & ev_datetime <= end_date_40_aas &
         charger_id == 0) #Filter so only the data within the chosen dates and
         correct charger are in the new car dataset
100
101 #Then did the same, for every car that participated:
102
103 #40 kWhs:
104
105 start_date_40_1 <- as.POSIXct("2024-09-30 15:55:00", tz = "CET")
106 end_date_40_1 <- as.POSIXct("2024-10-04 05:00:00", tz = "CET")
107 nissan_leaf_40_1 <- new_dataset %>%
108   filter(ev_datetime >= start_date_40_1 & ev_datetime <= end_date_40_1 & charger
        _id == 4)
109
110 visits_nissan_leaf_40_2 <- data.frame(
111   start_date_40_2 = as.POSIXct(c("2024-02-08 11:45:00", "2024-11-01 00:05:00", "
        2025-02-06 11:00:00"), tz = "CET"),
112   end_date_40_2 = as.POSIXct(c("2024-02-09 19:00:00", "2024-11-03 13:05:00", "
        2025-02-09 19:00:00"), tz = "CET"),
113   charger_id = c(3, 3, 3) #multiple visits so have to define which charger was
        used for each visit
114 )
115 nissan_leaf_40_2 <- new_dataset %>%
116   filter(
117     (ev_datetime >= visits_nissan_leaf_40_2$start_date[1] & ev_datetime <=
          visits_nissan_leaf_40_2$end_date[1] & charger_id == visits_nissan_leaf_
          40_2$charger_id[1]) |
118       (ev_datetime >= visits_nissan_leaf_40_2$start_date[2] & ev_datetime <=
          visits_nissan_leaf_40_2$end_date[2] & charger_id == visits_nissan_leaf
          _40_2$charger_id[2]) |
119       (ev_datetime >= visits_nissan_leaf_40_2$start_date[3] & ev_datetime <=
          visits_nissan_leaf_40_2$end_date[3] & charger_id == visits_nissan_leaf
          _40_2$charger_id[3])
120   )
121
122 start_date_40_3 <- as.POSIXct("2023-12-23 06:00:00", tz = "CET")
123 end_date_40_3 <- as.POSIXct("2023-12-30 21:00:00", tz = "CET")
124 nissan_leaf_40_3 <- new_dataset %>%
125   filter(ev_datetime >= start_date_40_3 & ev_datetime <= end_date_40_3 & charger
        _id == 2)
126
127 #62 kWhs:
128
129 visits_nissan_leaf_62_1 <- data.frame(
130   start_date_62_1 = as.POSIXct(c("2024-02-17 06:30:00", "2024-09-25 16:00:00"),
        tz = "CET"),
```

```
  end_date_62_1 = as.POSIXct(c("2024-02-24 12:30:00", "2024-10-02 04:05:00"), tz
      = "CET"),
  charger_id = c(2, 3)
)
nissan_leaf_62_1 <- new_dataset %>%
  filter(
    (ev_datetime >= visits_nissan_leaf_62_1$start_date[1] & ev_datetime <=
       visits_nissan_leaf_62_1$end_date[1] & charger_id == visits_nissan_leaf_
       62_1$charger_id[1]) |
      (ev_datetime >= visits_nissan_leaf_62_1$start_date[2] & ev_datetime <=
         visits_nissan_leaf_62_1$end_date[2] & charger_id == visits_nissan_leaf
         _62_1$charger_id[2])
  )

visits_nissan_leaf_62_2 <- data.frame(
  start_date_62_2 = as.POSIXct(c("2023-10-25 11:40:00", "2024-04-11 08:00:00", "
     2024-11-06 14:15:00"), tz = "CET"),
  end_date_62_2 = as.POSIXct(c("2023-10-26 02:40:00", "2024-04-17 17:20:00", "
     2024-11-14 04:15:00"), tz = "CET"),
  charger_id = c(4, 4, 3)
)
nissan_leaf_62_2 <- new_dataset %>%
  filter(
    (ev_datetime >= visits_nissan_leaf_62_2$start_date[1] & ev_datetime <=
       visits_nissan_leaf_62_2$end_date[1] & charger_id == visits_nissan_leaf_
       62_2$charger_id[1]) |
      (ev_datetime >= visits_nissan_leaf_62_2$start_date[2] & ev_datetime <=
         visits_nissan_leaf_62_2$end_date[2] & charger_id == visits_nissan_leaf
         _62_2$charger_id[2]) |
      (ev_datetime >= visits_nissan_leaf_62_2$start_date[3] & ev_datetime <=
         visits_nissan_leaf_62_2$end_date[3] & charger_id == visits_nissan_leaf
         _62_2$charger_id[3])
  )

visits_nissan_leaf_62_3 <- data.frame(
  start_date_62_3 = as.POSIXct(c("2023-10-25 11:40:00", "2024-03-18 10:10:00"),
     tz = "CET"),
  end_date_62_3 = as.POSIXct(c("2023-10-26 11:30:00", "2024-03-20 14:05:00"), tz
      = "CET"),
  charger_id = c(2, 2)
)
nissan_leaf_62_3 <- new_dataset %>%
  filter(
    (ev_datetime >= visits_nissan_leaf_62_3$start_date[1] & ev_datetime <=
       visits_nissan_leaf_62_3$end_date[1] & charger_id == visits_nissan_leaf_
       62_3$charger_id[1]) |
      (ev_datetime >= visits_nissan_leaf_62_3$start_date[2] & ev_datetime <=
         visits_nissan_leaf_62_3$end_date[2] & charger_id == visits_nissan_leaf
         _62_3$charger_id[2])
  )

visits_nissan_leaf_62_4 <- data.frame(
  start_date_62_4 = as.POSIXct(c("2024-04-02 12:55:00", "2024-06-13 19:35:00"),
     tz = "CET"),
  end_date_62_4 = as.POSIXct(c("2024-04-11 23:55:00", "2024-06-20 10:05:00"), tz
      = "CET"),
  charger_id = c(3, 4)
)
nissan_leaf_62_4 <- new_dataset %>%
  filter(
    (ev_datetime >= visits_nissan_leaf_62_4$start_date[1] & ev_datetime <=
       visits_nissan_leaf_62_4$end_date[1] & charger_id == visits_nissan_leaf_
       62_4$charger_id[1]) |
```

```
      (ev_datetime >= visits_nissan_leaf_62_4$start_date[2] & ev_datetime <=
          visits_nissan_leaf_62_4$end_date[2] & charger_id == visits_nissan_leaf
          _62_4$charger_id[2])
  )



#Just checking that no data in the different car datasets overlap, as this
    should not happen
################################################################################
# Combining all the filtered car datasets
car_dataset <- rbind(nissan_leaf_40kwh_aas, nissan_leaf_40_1, nissan_leaf_40_2,
    nissan_leaf_40_3, nissan_leaf_62_1, nissan_leaf_62_2, nissan_leaf_62_3,
    nissan_leaf_62_4)

#There should be no duplicates
duplicated_rows_car_dataset <- duplicated(car_dataset)

any(duplicated_rows_car_dataset) #Checking if any of the values in duplicated_
    rows_car_dataset are true (which would indicate duplicates)
#It states "FALSE" = No duplicates found!, and it did state FALSE



#Checking for unused rows:
################################################################################
#Checking if any data in new_dataset is NOT in any of the car datasets: Can see
    in unused_rows how many rows are unused

#Checks which rows from new_dataset are not included in the car_dataset
unused_rows <- anti_join(new_dataset, car_dataset, by = "ev_log_id")
View(unused_rows)  #Shows unused rows, 679 rows in total,
#None of these could be found in the participant sheet or mails.



#40
nissan_leaf_40kwh_aas <- cbind(nissan_leaf_40kwh_aas, battery_capacity_Wh =
    40000)
nissan_leaf_40_1 <- cbind(nissan_leaf_40_1, battery_capacity_Wh = 40000)
nissan_leaf_40_2 <- cbind(nissan_leaf_40_2, battery_capacity_Wh = 40000)
nissan_leaf_40_3 <- cbind(nissan_leaf_40_3, battery_capacity_Wh = 40000)

#62
nissan_leaf_62_1 <- cbind(nissan_leaf_62_1, battery_capacity_Wh = 62000)
nissan_leaf_62_2 <- cbind(nissan_leaf_62_2, battery_capacity_Wh = 62000)
nissan_leaf_62_3 <- cbind(nissan_leaf_62_3, battery_capacity_Wh = 62000)
nissan_leaf_62_4 <- cbind(nissan_leaf_62_4, battery_capacity_Wh = 62000)



#To calculate RTE based on cycles, a function was created:
################################################################################
calc_rte_cycles_new <- function(df,
                                time_gap_hours = 24, #If 24 hours pass, this
                                    will count as a new visit
                                soc_leeway = 3, #When looking for a cycle, SoC
                                    is allowed to be off by 3%
                                soc_transition_threshold = 10, #When changing
                                    status in the cycle, the SoC jump is not
                                    allowed to be bigger than 10%
                                min_chg_rows = 3, #A cycle has to consist of 3
                                    charging and 3 discharging rows, minimum
```

```
                                    min_dchg_rows = 3) {
  library(dplyr) #dplyr = data manipulation
  library(lubridate) #lubridate = date and time

  df <- as.data.frame(df)  #Making sure it's a data frame and not a tibble,
      since tibbles behave a bit differently

  #Sorting data by charger and time so it can be read through chronologically
  df <- df %>% arrange(charger_id, ev_datetime)

  #Making the visits:
  df <- df %>%
    group_by(charger_id) %>% #group by charger
    mutate(
      time_diff_hours = as.numeric(difftime(ev_datetime, lag(ev_datetime,
          default = first(ev_datetime)), units = "hours")), #Calculates the time
           difference in hours between the current row and the previous row,
          where lag is used to check the time for the previous row, first row is
           assigned 0 since first row in the dataset
      new_visit = if_else(is.na(time_diff_hours) | time_diff_hours > time_gap_
          hours, 1, 0), #If the timegap is missing or larger than the set
          threshold of 24 hours, then the current row is the start of a new
          visit. 0 = FALSE, 1 = TRUE. The line in the middle means OR, so this
          statement is true both if EITHER time_diff_hours is missing OR it
          exceeds time_gap_hours. Both do not need to be true.
      visit_id = cumsum(new_visit) #Cumsum = cumulative sum. Every new visit
          flag adds up, so a unique visit_id is assigned to each visit (when new
          _visit = 1, visit id --> +1)
    ) %>%
    ungroup() #removes the grouping, as we do not want any future operations to
        be affected by this.

  #Creating the cycle-finding function:
  find_cycles_in_visit <- function(data_visit, soc_leeway, soc_transition_
      threshold,
                                   min_chg_rows, min_dchg_rows) { #Finds cycles
                                       that follow the set rules. (the ones
                                       mentioned at the start)
    data_visit <- data_visit[order(data_visit$ev_datetime), ] #Makes sure
        everything is in chronological order
    n <- nrow(data_visit) #Checks how many rows are in the visit

    cycle_nr_vec <- rep(NA_integer_, n) #Creates a vector with length n, and
        fills it with NA values of the integer type
    cycle_count <- 0 #Counts the number of cycles

    i <- 1 #Starts at the first row in the dataset
    while (i < n) { #As long as the condition stays true/until i (row nr.)
        reaches n (number of rows in the dataset), the while loop will run
      start_i <- i #States that the start is at row i
      start_status <- data_visit$charger_status[start_i] #Retrieves the charger
          status from the row at start_i
      start_soc <- data_visit$soc[start_i] #Retrieves the SoC from the row at
          start_i

      if (!start_status %in% c("Charging", "Discharging")) { #Checks that if the
           start_status is charging or discharging. If it is neither, it skips
          the row
        i <- i + 1 #Moves to the next row in the loop
        next #Jumps to the next part of the while-loop
      }

```

```
260       while (i < n && data_visit$charger_status[i + 1] == start_status) i <- i +
              1 #While loop that continues to run, as long as i is less than the
            number of rows (n) and the next rows charger status, is the same as
            start_status.
261       phase1_end <- i #Last row of the first phase of the cycle. (So last row in
              that cycle of charging/discharging depending on what the cycle
            started with)
262       if (i == n) break #If the last row of the dataset is reached, then exit
            the loop (as there would be nothing more to process)
263
264       #Second half of the cycle:
265       second_phase_start <- i + 1 #Sets the first row after the first phase as
            the possible start of the second phase
266       second_status <- data_visit$charger_status[second_phase_start] #Checks the
              charger_status at the start of the second phase
267       if (second_status == start_status) { #Checks if the second phase has the
            same charger_status as the first phase
268         i <- i + 1 #Move down one row
269         next #If the second phase has the same charger_status as the first phase
                , then it is skipped.
270       }
271
272       #Checking the SoC jumps:
273       soc_jump <- abs(data_visit$soc[phase1_end] - data_visit$soc[second_phase_
            start]) #Calculates the difference between the SoC status at the end
            of the first phase against the start of the second phase and uses
            absolute value to avoid negative values, as only the difference
            matters
274       if (soc_jump > soc_transition_threshold) {#Checks if the jump in SoC is
            bigger than the allowed threshold that was set (10%)
275         i <- i + 1 #If limit is surpassed, it starts over in an attempt to try
              to find a new cycle
276         next
277       }
278
279       #If second phase is within the SoC jump limit, the loop continues
280       i <- second_phase_start #States that second phase starts at row i. (first
            row of the second phase)
281       while (i < n && data_visit$charger_status[i + 1] == second_status) i <- i
            + 1 #As long as the charger status remains the same and the end of the
              dataset isn't reached, move on to the next row
282       end_i <- i #Once the loop is completed or the end of the second phase is
            reached, then remember that row i is the end of this phase
283       phase2_rows <- seq(second_phase_start, end_i) #Creates a list of the row
            numbers that belong to the second phase of the cycle
284
285       valid_cycle <- FALSE #State that so far we have not yet found a valid
            cycle, with the intention of later changing this to true when
            conditions are met
286       chosen_start_i <- start_i #Is set as a possible start for a valid cycle
287       chosen_end_i <- NA #Sets the end to NA, which will later be filled with
            the end of the cycle when that is found
288
289       #First we want to check if a perfect cycle exists where start soc = end
            soc
290       if (start_status == "Charging") { #Only check for cycles that starts with
            charging
291         perfect_matches <- phase2_rows[data_visit$soc[phase2_rows] == start_soc]
              #Compares the SoC values in the second phase (in this case
            discharging) to the start soc to look for perfect cycles
292         if (length(perfect_matches) > 0) { #Checks if any rows in the second
            phase have a soc equal to the start soc, if it does it is a valid
            cycle
```

```
293           chosen_end_i <- max(perfect_matches) #If multiple rows in a row have
                  the same soc, then the last one is included, giving the longest
                  possible cycle
294           valid_cycle <- TRUE #Marks it as a valid cycle
295         } else { #If a perfect cycle is not found then it will try to look for a
                cycle that works with the 3% allowance:
296           near_matches <- phase2_rows[data_visit$soc[phase2_rows] >= start_soc &
                   data_visit$soc[phase2_rows] <= start_soc + soc_leeway] #Checks
                  all rows with soc between start soc and start soc + leeway (3%)in
                  the second phase
297           if (length(near_matches) > 0) { #If at least one row matches
298             chosen_end_i <- max(near_matches) #Take the last matching row and
                    end the cycle
299             valid_cycle <- TRUE #Mark it as a valid cycle
300           }
301         }
302       } else {
303         perfect_matches <- phase2_rows[data_visit$soc[phase2_rows] == start_soc]
                #Searches through the second phase to look after rows where SoC is
                exactly the same as start soc
304         if (length(perfect_matches) > 0) { #Checks if any rows in the second
                phase have a soc equal to the start soc, if it does it is a valid
                cycle
305           chosen_end_i <- max(perfect_matches) #If multiple rows in a row have
                  the same soc, then the last one is included, giving the longest
                  possible cycle
306           valid_cycle <- TRUE #Marks it as a valid cycle
307         } else {
308           near_matches <- phase2_rows[data_visit$soc[phase2_rows] <= start_soc &
                   data_visit$soc[phase2_rows] >= start_soc - soc_leeway]  #Checks
                  all rows with soc between start soc and start soc + leeway (3%)in
                  the second phase
309           if (length(near_matches) > 0) { #If at least one row matches
310             chosen_end_i <- max(near_matches)  #Take the last matching row and
                    end the cycle
311             valid_cycle <- TRUE #Marks it as a valid cycle
312           }
313         }
314       }
315
316       if (!valid_cycle) { #Only runs if the previous attempts fail (no cycles
              are found)
317         end_soc <- data_visit$soc[end_i] #Takes the soc at the end of the second
                phase
318         block_rows <- seq(start_i, phase1_end) #Define the range of rows that
              are a part of the first phase
319
320         if (start_status == "Charging") { #If the cycle starts with charging
321           perfect_matches <- block_rows[data_visit$soc[block_rows] == end_soc] #
                  Try to find a perfect match where the soc is equal to the end soc
                  (soc status at the last row of phase 1)
322           if (length(perfect_matches) > 0) { #If at least one row matches
323             chosen_start_i <- min(perfect_matches) #From the rows that matched
                    the condition (matching or +13% soc), use the earliest occurance
324             valid_cycle <- TRUE #Marks it as a valid cycle
325             chosen_end_i <- end_i #At the end of the cycle, mark this as the
                    last row of the second phase.
326           } else {
327             near_matches <- block_rows[data_visit$soc[block_rows] >= end_soc &
                  data_visit$soc[block_rows] <= end_soc + soc_leeway] #Finds all
                  rows in the first phase where the soc is bigger than or equal to
                   the end soc, but not passing the limit (end_soc + leeway)
```

```
            if (length(near_matches) > 0) { #If at least one row matches,
                continue
              chosen_start_i <- min(near_matches) #picks the earliest occurance
                 that matches
              valid_cycle <- TRUE #Marks the cycle as valid
              chosen_end_i <- end_i #Uses the found second phase as the end of
                 the cycle
            }
          }
        } else {
          perfect_matches <- block_rows[data_visit$soc[block_rows] == end_soc] #
              Looks for rows inside block_rows where the soc matches end_soc
              exactly (from the end of the second phase) (block rows = the rows
              from when the curren status starts (start_i) until it switches to
              the opposite charging status (phas1_end))
          if (length(perfect_matches) > 0) { #If a match is found, continue
            chosen_start_i <- min(perfect_matches) #Uses the earliest occurance
                where a match is found
            valid_cycle <- TRUE #Marks it as a valid cycle
            chosen_end_i <- end_i #Uses the end from the second phase
          } else {
            near_matches <- block_rows[data_visit$soc[block_rows] <= end_soc &
                data_visit$soc[block_rows] >= end_soc - soc_leeway] #If perfect
                matches are not found it starts looking for matches with +13%
            if (length(near_matches) > 0) { #If a match is found, continue
              chosen_start_i <- min(near_matches) #Use the earliest matching row
                  as the start of the cycle
              valid_cycle <- TRUE #Mark the cycle as valid
              chosen_end_i <- end_i # Keep the same end_i
            }
          }
        }
      }

      if (valid_cycle && !is.na(chosen_end_i)) { #Checks if it is marked as a
          valid cycle, and has an end row been found for that cycle
        cycle_rows <- seq(chosen_start_i, chosen_end_i) #Creates a list with the
             row numbers that represent every row included in a cycle, from
            start to finish
        n_chg <- sum(data_visit$charger_status[cycle_rows] == "Charging") #
            Counts the number of charging rows in the cycle
        n_dchg <- sum(data_visit$charger_status[cycle_rows] == "Discharging") #
            Counts the number of discharging rows in the cycle

        if (n_chg >= min_chg_rows && n_dchg >= min_dchg_rows) { #Checks if the
            cycle has enough charging and discharging rows (3 of each was the
            minimum set limit)
          cycle_count <- cycle_count + 1 #Counts cycles in the dataset and
              assigns unique cycle numbers
          cycle_nr_vec[cycle_rows] <- cycle_count #Stores the current cycle
              number in a vector for every row in the cycle
          i <- chosen_end_i + 1 #Goes to the next row after the detected cycle
              ends
          next #Skips to the next part of the loop
        }
      }
      i <- i + 1 #If no cycle has been found until where the code currently is,
          then check the next row and so on
    }
    data_visit$cycle_nr <- cycle_nr_vec #Add the vector with the cycle number or
        NA back into the original data
    return(data_visit) #Returns the full data for the visit, with the new column
        (cycle_nr)
```

```
}

#Detecting the cycles
df <- df %>% #filter the dataframe
  group_by(charger_id, visit_id) %>% #groups dataset by charger and visit id
  group_modify(~ find_cycles_in_visit(.x, soc_leeway, soc_transition_threshold
      , min_chg_rows, min_dchg_rows)) %>% #For each group run the find_cycles_
      in_visit function, .x refers to all rows from each combination of
      charger id and visit id, then all of these combinations are chekced for
      cycles within the limitations set
  ungroup() #ungroups data

#Adding the delta_t and delta_t_watt with the 30 minute limit.
df <- df %>% #edit dataframe
  group_by(charger_id, visit_id, cycle_nr) %>% #group by charger id visit id
      and cycle number
  arrange(ev_datetime, .by_group = TRUE) %>% #Sorting rows chronologically
  mutate(
    delta_t = as.numeric(difftime(lead(ev_datetime), ev_datetime, units = "
        secs")), #calculates the time difference in seconds, between the
        current and the next row. lead makes it so the duration of this row
        lasts until the next row starts.
    delta_t = case_when( #adjusts delta_T
      is.na(delta_t) ~ 1, #If there is not a row after the current row, then
          delta_t is automatically assigned 1
      delta_t > 1800 ~ 1, #If more than 30 minutes pass (1800 seconds) between
           the current and next row, then delta_t is also set to 1 (meaning it
           lasts for 1 second) in order to avoid overestimating energy
          transfer in time gaps.
      TRUE ~ delta_t #For any other instance, the normal delta_t is used (time
           difference between current and next row in second)
    ),
    delta_t_watt = power_wattage * delta_t #Delta_t_watt is defined as the
        power_wattage * delta_t
  ) %>%
  ungroup() #ungroups data

#Calculating the RTE
df_cycles <- df %>% #edit dataframe
  filter(!is.na(cycle_nr)) %>% #only keeps rows that belong to an identified
      cycle
  group_by(charger_id, visit_id, cycle_nr) %>% #groups by charger_id visit_id
      and cycle number
  summarise( #calculating each cycles:
    energy_in = sum(delta_t_watt[charger_status == "Charging"], na.rm = TRUE),
         #calculating energy input
    energy_out = sum(abs(delta_t_watt[charger_status == "Discharging"]), na.rm
         = TRUE), #calculating energy output
    RTE = if_else(energy_in == 0, NA_real_, (energy_out / energy_in) * 100), #
        calculating the RTE
    .groups = "drop" #Removes grouping after being done with summary
  )

#Merges the RTE results back into the original dataset
df <- df %>%
  left_join(df_cycles, by = c("charger_id", "visit_id", "cycle_nr")) #takes
      the summary table (df_cycles) and merges it into the full dataset

#Printing the results
print(df_cycles) #Prints the summary table (df_cycles) in the "console"
avg_rte <- mean(df_cycles$RTE, na.rm = TRUE) #Calculates average RTE across
    all cycles in the dataset
```

```
  cat("Average RTE across all complete cycles:", round(avg_rte, 2), "%\n") # Prints the average RTE with 2 decimals.
  return(df) #Returns the full dataset
}



#Applying the function to the vehicle datasets and getting the vehicles average RTE
################################################################################

#40

RTE_cycle_40_aas_new <- calc_rte_cycles_new(nissan_leaf_40kwh_aas) #Uses the created cycle identification function, on the nissan_leaf_40kwh_aas dataset, creating a new dataset called RTE_cycle_aas_new

RTE_cycle_40_1_new <- calc_rte_cycles_new(nissan_leaf_40_1) #Uses the created cycle identification function, on the nissan_leaf_40_1, creating a new dataset called RTE_cycle_40_1

RTE_cycle_40_2_new <- calc_rte_cycles_new(nissan_leaf_40_2) #Uses the created cycle identification function, on the nissan_leaf_40_2, creating a new dataset called RTE_cycle_40_2

RTE_cycle_40_3_new <- calc_rte_cycles_new(nissan_leaf_40_3) #Uses the created cycle identification function, on the nissan_leaf_40_3, creating a new dataset called RTE_cycle_40_3


#62

RTE_cycle_62_1_new <- calc_rte_cycles_new(nissan_leaf_62_1) #Uses the created cycle identification function, on the nissan_leaf_62_1, creating a new dataset called RTE_cycle_62_1

RTE_cycle_62_2_new <- calc_rte_cycles_new(nissan_leaf_62_2) #Uses the created cycle identification function, on the nissan_leaf_62_2, creating a new dataset called RTE_cycle_62_2

RTE_cycle_62_3_new <- calc_rte_cycles_new(nissan_leaf_62_3) #Uses the created cycle identification function, on the nissan_leaf_62_3, creating a new dataset called RTE_cycle_62_3

RTE_cycle_62_4_new <- calc_rte_cycles_new(nissan_leaf_62_4) #Uses the created cycle identification function, on the nissan_leaf_62_4, creating a new dataset called RTE_cycle_62_4



################################################################################
#First have to make a new dataset with all the cycles from all the vehicle with the RTE calculations included
#When calculating deviation and RTE based on temperature and SoC classes, all data has to be combined. Therefore each dataset is given a dataset id, as multiple datasets have visit id 1  and cycle id 1.


RTE_cycle_40_1_new <- RTE_cycle_40_1_new %>% mutate(dataset_id = "RTE_cycle_40_1_new") #Making a new column called "dataset_id" and assigning all rows in RTE_cycle_40_1_new the dataset title in that column.
RTE_cycle_40_2_new <- RTE_cycle_40_2_new %>% mutate(dataset_id = "RTE_cycle_40_2_new") #Making a new column called "dataset_id" and assigning all rows in
```

```
    RTE_cycle_40_2_new the dataset title in that column.
RTE_cycle_40_3_new <- RTE_cycle_40_3_new %>% mutate(dataset_id = "RTE_cycle_40_3
    _new") #Making a new column called "dataset_id" and assigning all rows in
    RTE_cycle_40_3_new the dataset title in that column.
RTE_cycle_40_aas_new <- RTE_cycle_40_aas_new %>% mutate(dataset_id = "RTE_cycle_
    40_aas_new") #Making a new column called "dataset_id" and assigning all rows
     in RTE_cycle_40_aas_new the dataset title in that column.
RTE_cycle_62_1_new <- RTE_cycle_62_1_new %>% mutate(dataset_id = "RTE_cycle_62_1
    _new") #Making a new column called "dataset_id" and assigning all rows in
    RTE_cycle_62_1_new the dataset title in that column.
RTE_cycle_62_2_new <- RTE_cycle_62_2_new %>% mutate(dataset_id = "RTE_cycle_62_2
    _new") #Making a new column called "dataset_id" and assigning all rows in
    RTE_cycle_62_2_new the dataset title in that column.
RTE_cycle_62_3_new <- RTE_cycle_62_3_new %>% mutate(dataset_id = "RTE_cycle_62_3
    _new") #Making a new column called "dataset_id" and assigning all rows in
    RTE_cycle_62_3_new the dataset title in that column.
RTE_cycle_62_4_new <- RTE_cycle_62_4_new %>% mutate(dataset_id = "RTE_cycle_62_4
    _new") #Making a new column called "dataset_id" and assigning all rows in
    RTE_cycle_62_4_new the dataset title in that column.

#Bind them together to one dataset:
Temp_RTE_cycle_new <- bind_rows(RTE_cycle_40_1_new, RTE_cycle_40_2_new, RTE_
    cycle_40_3_new, RTE_cycle_40_aas_new, RTE_cycle_62_1_new, RTE_cycle_62_2_new
    , RTE_cycle_62_3_new, RTE_cycle_62_4_new)



################################################################################
#Calculating overall cycle statistics:
cycle_stats <- Temp_RTE_cycle_new %>%
  filter(!is.na(cycle_nr)) %>% #Only keeping rows in a cycle
  group_by(dataset_id, charger_id, visit_id, cycle_nr) %>% #For each unique
      cycle
  summarise( #do as follows:
    avg_temp = mean(temp, na.rm = TRUE), #calculates average temperature
    temp_range = max(temp, na.rm = TRUE) - min(temp, na.rm = TRUE), #calculate
        the temperature range (difference between highest and lowest)
    avg_soc = mean(soc, na.rm = TRUE), #Calculate average SoC
    soc_range = max(soc, na.rm = TRUE) - min(soc, na.rm = TRUE), #Calculate
        average soc range
    RTE = first(RTE), #Since the RTE is the same for all observations belonging
        to the same cycle, only one is needed
    .groups = "drop" #drops grouping
  )
#Calculating averages for different variables
overall_avg_temp_range <- mean(cycle_stats$temp_range, na.rm = TRUE) #Calculates
     the overall average temperature range (from all 104 cycles)
overall_avg_soc_range <- mean(cycle_stats$soc_range, na.rm = TRUE) #Calculates
    the overall average soc range (from all 104 cycles)
overall_avg_RTE <- mean(cycle_stats$RTE, na.rm = TRUE) #Calculates the overall
    average RTE (from all 104 cycles)

#Counting how many cycles had temperature ranges <= 5 degrees C
Cycles_temp_range_less_than_5_degrees <- cycle_stats %>% #filters to only
    include the cycles with a temperature range less or equal to 5 degrees C
  filter(temp_range <= 5) %>%
  nrow() #counts how many cycles there are
#printing results in the console
print(overall_avg_temp_range)
print(overall_avg_soc_range)
print(overall_avg_RTE)
print(Cycles_temp_range_less_than_5_degrees)

```

```
487
488
489 ##############################################################################
490 #Vehicle plot average RTE per vehicle
491 vehicle_rte_summary <- Temp_RTE_cycle_new %>%
492   filter(!is.na(cycle_nr)) %>% #Only keep observations from cycles
493   distinct(dataset_id, charger_id, visit_id, cycle_nr, .keep_all = TRUE) %>% #
        only keep one row per cycle
494   group_by(dataset_id) %>% #groups data per vehicle
495   summarise(
496     mean_RTE = mean(RTE, na.rm = TRUE), #Calculates average RTE for each vehicle
497     min_RTE = min(RTE, na.rm = TRUE), #Retrieves the lowest RTE value each
          vehicle had
498     max_RTE = max(RTE, na.rm = TRUE), #Retrieves the highest RTE value each
          vehicle had
499     n_cycles = n(), #Counts number of cycles in each vehicle
500     .groups = "drop" #Drops grouping
501   ) %>%
502   mutate(vehicle_label = recode(dataset_id, #Creating a new column called
        vehicle_label  to give the cars more "readable" names.
503                                 "RTE_cycle_40_1_new" = "NL 40 kWh 1",
504                                 "RTE_cycle_40_2_new" = "NL 40 kWh 2",
505                                 "RTE_cycle_40_3_new" = "NL 40 kWh 3",
506                                 "RTE_cycle_40_aas_new" = "NL 40 kWh Aas",
507                                 "RTE_cycle_62_1_new" = "NL 62 kWh 1",
508                                 "RTE_cycle_62_2_new" = "NL 62 kWh 2",
509                                 "RTE_cycle_62_3_new" = "NL 62 kWh 3",
510                                 "RTE_cycle_62_4_new" = "NL 62 kWh 4"
511   ))
512 #Plot
513 ggplot(vehicle_rte_summary, aes(x = vehicle_label, y = mean_RTE)) + #Defining
      which dataset to use and what the x and y axis represent
514   geom_col(fill = "lightgreen", width = 0.6) + #makes the bars, defining the bar
        width
515   geom_errorbar(aes(ymin = min_RTE, ymax = max_RTE), #adding the range lines
        showing the highest and lowest RTE in the class.
516                 width = 0.2, color = "darkgreen") + #Defining the range lines
                      width and colour
517   geom_text(aes(label = paste0("n=", n_cycles)), #Adding the text above the
        plots bars
518             vjust = -1.5, size = 3.5) + #vjust adjusts the height above the bar
519   geom_text(aes(label = paste0(sprintf("%.2f", mean_RTE), "%")), #Adding the
        label showing the mean RTE in each class
520             vjust = -0.3, size = 3.5) + #height adjustment above bar
521   labs(
522     x = "Vehicle", #x-axis label
523     y = "Average RTE [%]" #y-axis label
524   ) +
525   theme_minimal() + #applies minimal theme
526   theme(
527     axis.text.x = element_text(angle = 45, hjust = 1) #adjust labels so they
          become diagonal
528   )
529
530
531
532 #Deviation per temperature class #calculated average temperature per cycle and
      sorted them into classes that way
533 ##############################################################################
534 #Calculating average temperature per cycle
535 cycles_avg_temp <- Temp_RTE_cycle_new %>%
536   filter(!is.na(cycle_nr)) %>% #only includes rows in cycles
```

```
  group_by(dataset_id, charger_id, visit_id, cycle_nr) %>% #finds all unique cycles
  summarise( #for each group (or here for every cycle)
    avg_temp = mean(temp, na.rm = TRUE), #calculates average temperature per group
    .groups = "drop" #drops group
  )
#Marks each cycle with its average temperature
Deviation_temp<- Temp_RTE_cycle_new %>%
  filter(!is.na(cycle_nr)) %>% #only includes rows in a cycle
  left_join(cycles_avg_temp, by = c("dataset_id", "charger_id", "visit_id", "cycle_nr")) %>% #merges the avg temperature into the full dataset (making sure that all unique ids get their respective temperature)
  mutate( #
    abs_power = abs(power_wattage), #calculates absolute value of delivered power (power_wattage)
    abs_requested = abs(requested_charge), #calculates absolute value of requested power (requested_charge)
    temp_class = cut( #assigns each cycle to a temperature class based on their avg_temp
      avg_temp,
      breaks = seq(-25, 25, by = 5), #divides temperatures into different classes, with 5 degrees in each
      include.lowest = TRUE, #includes the lowest temperature
      right = FALSE #makes the intervals/classes left-closed and right open [ >
    )
  )
#Calculating deviation per temperature class in %
Deviation_temp_summary <- Deviation_temp%>%
  group_by(temp_class) %>% #groups by temperature class
  summarise(
    total_actual_power = sum(abs_power, na.rm = TRUE), #sums the absolute power delivered in every temperature class (any missing values are ignored)
    total_requested_power = sum(abs_requested, na.rm = TRUE), #sums the absolute power requested in every temperature class (any missing values are ignored)
    deviation_percent = 100 - ((total_actual_power / total_requested_power) * 100), #calculates the percentage deviation between what is delivered and what is requested, by taking 100%-the percentage that is delivered.
    n_rows = n(), #counts observations in each temperature class
    .groups = "drop" #drops group
  )
print(Deviation_temp_summary) #prints results in console



#Calculating deviation per soc class and calculates deviation for charging and discharging separately
################################################################################
Deviation_soc_split <- Temp_RTE_cycle_new %>%
  filter(!is.na(cycle_nr)) %>% #only includes rows in cycles
  mutate(
    soc_class = cut( #places each soc value into a soc class
      soc,
      breaks = c(21, 31, 41, 51, 61, 71, 81, 92), #Soc ranges
      include.lowest = TRUE, #retrieves the lowest value (here 21%)
      right = FALSE,#says to not include the highes value (92) and that is because the highest we have is 91, but had 91 be used only 90.999 would be included and not 91 so therefore 92 is used although the higest left after filtering is 91
      labels = c("21-31%", "31-41%", "41-51%", "51-61%", "61-71%", "71-81%", "81-91%") #sets the names of the soc classes that will appear in the new soc_class column
```

```
    ),
    abs_power = abs(power_wattage), #calculates absolute value of each row in the power_wattage column
    abs_requested = abs(requested_charge) #calculates absolute value of each row in the requested_charge column
  )
#calculating the deviation
Deviation_soc_status <- Deviation_soc_split %>%
  filter(charger_status %in% c("Charging", "Discharging")) %>% #only includes rows where the vehicle is charging or discharging
  group_by(soc_class, charger_status) %>% #groups by soc class and charger status
  summarise( #for each group (here separately for charging and discharging)
    total_actual_power = sum(abs_power, na.rm = TRUE), #uses the absolute value on the total power that is delivered in each class
    total_requested_power = sum(abs_requested, na.rm = TRUE), #uses the absolute value of the total requested power in each class
    deviation_percent = round(100 - ((total_actual_power / total_requested_power) * 100), 2), #calculates the percentage deviation between what is delivered and what is requested, by taking 100%-the percentage that is delivered. 2 decimals
    n_rows = n(), #counts how many observations each class has
    .groups = "drop" #drops grouping
  )
#printing the results
print(Deviation_soc_status)



#Calculating average RTE per battery capacity, where each cycle is considered equally
################################################################################
avg_rte_batcap <- Temp_RTE_cycle_new %>% #filter dataset Temp_RTE_cycle new, and after it is filtered the filtered data becomes a new dataset called avg_rte_batcap
  filter(!is.na(cycle_nr)) %>% #only keeping rows that belong to an identified cycle
  distinct(dataset_id, charger_id, visit_id, cycle_nr, .keep_all = TRUE) %>% #From each cycle, only a single row is kept
  group_by(battery_capacity_Wh) %>% #groups data by the battery capacity
  summarise(
    average_RTE = format(round(mean(RTE, na.rm = TRUE), 2), nsmall = 2), #Calculates average RTE per battery capacity
    n_cycles = n(), #Counts how many cycles that contributes to each average
    .groups = "drop" #removes grouping
  )
print(avg_rte_batcap) #prints the results



#Calculating average RTE per temperature class based on the cycle where the average cycle temp is used and then all data belonging to that cycle is sorted into the belonging temperature class
################################################################################
#Creating one row with average temperature per cycle
unique_cycles <- Temp_RTE_cycle_new %>%
  filter(!is.na(cycle_nr)) %>% #only includes rows that belong to identified cycles
  group_by(dataset_id, charger_id, visit_id, cycle_nr) %>% #Finding all the unique cycles (as each combination if these ids/numbers are unique meaning its a "new" cycle)
  summarise( #for each group:
```

```
    RTE = first(RTE),  #Only uses the first RTE in the cycle (as all rows are labeled with the CYCLES RTE, meaning they are all the same and only 1 is needed)
    avg_temp = mean(temp, na.rm = TRUE), #Calculates the average temp for the cycle
    .groups = "drop" #drop grouping
  ) %>%
  mutate( #adds a new column temp_class where a cycle is sorted into its correct class based on their avg_temp
    temp_class = cut(
      avg_temp,
      breaks = seq(-25, 25, by = 5), #creating the 5 degree bins from -25 up to 25 degrees C
      include.lowest = TRUE, #includes -25 in the first class
      right = FALSE #bins are left-closed and right open [ >
    )
  )
#Calculating the average RTE for each temperature class
RTE_temp <- unique_cycles %>%
  group_by(temp_class) %>% #grouped by temp_class
  summarise( #for each group:
    average_RTE = mean(RTE, na.rm = TRUE), #calculate the average RTE per temp class
    average_temp = mean(avg_temp, na.rm = TRUE), #calculate the average temperature of the cycles that were sorted into that temp_class
    n_cycles = n(), #counts amount of cycles included in each temperature class
    .groups = "drop" #ungroups
  ) %>%
  mutate(
    average_RTE = format(round(average_RTE, 2), nsmall = 2), #rounds the average RTE in each class to 2 decimals and forces it to display those 2 decimals
    average_temp = format(round(average_temp, 2), nsmall = 2) #rounds the average temperature in each class to 2 decimals and forces it to display those 2 decimals
  )
print(RTE_temp) #prints results in the console



#Plot for temp RTE
#One observation for each cycle
unique_cycles_plot <- Temp_RTE_cycle_new %>%
  filter(!is.na(cycle_nr)) %>% #Only keeps rows that belong to a cycle
  group_by(dataset_id, charger_id, visit_id, cycle_nr) %>% #unique cycles
  summarise(
    RTE      = first(RTE), #uses the first RTE value
    avg_temp = mean(temp, na.rm = TRUE), #calculates average temperature for the cycle
    .groups  = "drop" #drops grouping
  ) %>%
  mutate(
    temp_class = cut( #creates a new column called temp class
      avg_temp, #uses the avg_temp to sort into the classes
      breaks = seq(-25, 25, by = 5), #making the classes
      include.lowest = TRUE, #includes -25
      right = FALSE #creating left closed, right open intervals [ >
    )
  )
Temp_RTE_summary <- unique_cycles_plot %>%
  group_by(temp_class) %>% #all cycles in the same class stays grouped together
  summarise(
    mean_RTE   = mean(RTE, na.rm = TRUE), #calculate average RTE for each class
```

```
    min_RTE    = min(RTE,  na.rm = TRUE), #Retrieves the lowest RTE value in
        each class
    max_RTE    = max(RTE,  na.rm = TRUE), #Retrieves the highest RTE value in
        each class
    mean_temp  = mean(avg_temp, na.rm = TRUE), #Calculates average temperature
        based on the cycles included in each class
    n_cycles   = n(), #Counts number of cycles in each class
    .groups    = "drop" #Drops grouping
  )
#Plot
ggplot(Temp_RTE_summary, aes(x = temp_class, y = mean_RTE)) +
  geom_col(fill = "lightgreen", width = 0.6) + #creates the bar, defines their
      width
  geom_errorbar(aes(ymin = min_RTE, ymax = max_RTE), #Adds the vertical line in
      each bar that shows the highest and lowest RTE included in each class
                width = 0.2, color = "darkgreen") + #adjusting the "errorbars'"
                    width and color
  geom_text(aes(label = paste0("n=", n_cycles)), #Adds the label stating the
      number of cycles in each class above the bar
            vjust = -2, size = 3.5) + #adjust the labels height and size
  geom_text(aes(label = paste0(sprintf("%.2f", mean_RTE), "%")), #Adds the label
       above the bar stating the average RTE in each class
            vjust = -0.3, size = 3.5) + #adjust the labels height and size
  # Average temperature value inside the bar (no unit)
  geom_text(aes(label = paste0("Avg. temp: ", round(mean_temp, 1))), #Puts the
      label stating the average temperature in each class based on the cycles
      included inside the bar
            vjust = 2.5, size = 3.2, color = "black") + #adjust the labels
                height, size and color
  labs(
    x = "Temperature class in degrees Celsius", #x-axis title
    y = "Average RTE [%]" #y-axis title
  ) +
  theme_minimal() #adds minimal theme



#Calculating RTE per SoC class, using each cycle and calculating its average SoC
    , and then one row from each cycle is put into the class with the average
    soc, and rte of that cycle. So each cycle is weighed equally
################################################################################
#Calculating the average SoC in each cycle:
cycle_avg_soc <- Temp_RTE_cycle_new %>% #edits the Temp_RTE_cycle_new and
    creates new dataset after filtering
  filter(!is.na(cycle_nr)) %>% #Only considers rows that are in a cycle
  group_by(dataset_id, charger_id, visit_id, cycle_nr) %>% #groups by their
      different ids
  summarise(
    avg_soc = mean(soc, na.rm = TRUE), #calculates average SoC for each cycle,
        based on all the rows in that cycle
    RTE = first(RTE),  #Since all the rows in the same cycle are assigned the
        cycles RTE, only one is picked (the first)
    .groups = "drop" #drops grouping
  )
#Assigning the SoC classes:
cycle_avg_soc <- cycle_avg_soc %>% #modifies the cycle_avg_soc dataset
  mutate( #creates a new column (soc_class) (in this case, mutate cal also edit
      an existing column)
    soc_class = cut( #creates a new column called SoC class
      avg_soc, #based on what the avg_soc is calculated to be, it is sorted into
          its respective soc class
      breaks = c(21, 31, 41, 51, 61, 71, 81, 92), #creating the soc classes
```

```
      include.lowest = TRUE, #left closed intervals. If a avg_soc is exactly one
           of the values, for example 21, then it will be included in the 21-31%
           class. it goes to 92 because the highest in the data is 91, but we
          dont want a right open interval with 91 as that only means 90.999 is
          included and not 91. therefore the limit is set for 92, so the class
          becomes [81,91]
      right = FALSE, #makes the intervals right open
      labels = c("21-31%", "31-41%", "41-51%", "51-61%", "61-71%", "71-81%", "
          81-91%") #The labels that will be assigned to each bin
    )
  )
RTE_soc_summary <- cycle_avg_soc %>% #filtering/modifying cycle_avg_soc
  group_by(soc_class) %>% #groups by soc_class
  summarise( #For each soc_class do the following:
    average_RTE = format(round(mean(RTE, na.rm = TRUE), 2), nsmall = 2), #
        Calculate the average RTE in each soc class, forced to use 2 decimals
    n_cycles = n(), #counts how many cycles have been sorted into each soc class
    .groups = "drop" #ungroups
  )
print(RTE_soc_summary) #prints the results



#RTE SoC graph:
#Summarise RTE by SoC class
soc_rte_summary <- cycle_avg_soc %>%
  group_by(soc_class) %>% #groups by soc class (21-31, 31-41 etc-)
  summarise( #In each group:
    mean_RTE = mean(RTE, na.rm = TRUE), #calculate the average RTE in each class
    min_RTE = min(RTE, na.rm = TRUE), #Retrieves the lowest RTE
    max_RTE = max(RTE, na.rm = TRUE), #Retrieves the highest RTE
    n = n(), #counts how many cycles are in each class
    .groups = "drop" #drops the grouping
  )
#Plotting the results
ggplot(soc_rte_summary, aes(x = soc_class, y = mean_RTE)) + #using the soc_rte_
    summary dataset, set the x-axis to the soc_class variable and the y to the
    mean_RTE
  geom_bar(stat = "identity", fill = "lightgreen", width = 0.6) + #creates the
      bar for each soc_class with the height being equal to the mean_RTW,
      identity means that it uses the actual values and not the counts. Colors
      the bar light green and the bars width is set to 0.6
  geom_errorbar(aes(ymin = min_RTE, ymax = max_RTE), width = 0.2, color = "
      darkgreen") + #Adds a vertical line that shows the range of RTEs that have
       been included in each soc_class. ymin=min_RTE and y_max=max_RTE defines
      the top and bottom of the line. width 0.2 is for the top part of the line,
       how long that is, and dark green sets the color for the line
  geom_text(aes(label = paste0("n=", n)), vjust = -2, size = 3.5) + #Adds the
      number of cycles above the bars, showing the amount of cycles (n) that are
       in each soc class, vjust adjusts how high above the bar the label is and
      paste0(n=n) just makes the label look like n=9 etc, and size adjusts the
      font size
  geom_text(aes(label = paste0(format(mean_RTE, digits = 2, nsmall = 2), "%")),
      vjust = -0.3, size = 3.5) + #Adds the label that shows the average RTE per
       SoC class with 2 significant decimals (digits) and forces it to include
      two decimals (nsmall). vjust again adjusts the height above the bar and
      size adjusts the font size
  labs( #creating labels for the x and y axis
    title = "",
    x = "SoC Class [%]", #x-axis label
    y = "Average RTE [%]" #y-axis label
  ) +
  theme_minimal() #applies the minimal theme
```

B Appendix: All registered cycles in all vehicles

Table B.1: Overview over all the RTEs for all the cycles registered in the Ås vehicle.

Car	Visit id	Cycle number	Cycle RTE [%]	SoC range in cycle [%]
nissan_leaf_40kwh_aas	1	1	37.42	76–>91–>76
	1	2	13.22	91–>89–>91
	6	1	12.86	66–>61–>66
	7	1	35.78	66–>72–>69
	7	2	75.26	83–>91–>83
	8	1	78.34	55–>39–>55
	9	1	81.88	78–>48–>78
	10	1	79.02	91–>79–>91
	10	2	82.25	91–>82–>91
	11	1	73.19	69–>66–>69
	11	2	83.24	78–>91–>78
	11	3	87.83	78–>91–>78
	11	4	77.56	88–>66–>88
	11	5	84.67	89–>91–>89
	11	6	107.70	88–>77–>85
	11	7	106.92	73–>76–>73
	11	8	84.32	72–>53–>72
	11	9	69.36	73–>91–>76
	11	10	83.18	76–>85–>76
	11	11	84.97	75–>65–>75
	11	12	75.64	76–>91–>77
	12	1	100.21	75–>76–>75
	12	2	86.00	73–>55–>73
	12	3	71.29	75–>91–>77
	13	1	23.83	63–>67–>66
	13	2	79.32	66–>91–>66
	14	1	76.14	76–>91–>76
	14	2	83.92	75–>53–>75
	17	1	80.91	55–>33–>55
	17	2	79.14	77–>91–>77
	17	3	54.66	77–>83–>77
	17	4	84.37	76–>63–>76

Table B.2: Overview over all the RTEs for all the cycles registered in first of the 40 kWh vehicles.

Car	Visit id	Cycle number	Cycle RTE [%]	SoC range in cycle [%]
nissan_leaf_40_1	1	1	85.85	91–>48–>91
	1	2	80.10	80–>69–>80
	1	3	89.25	82–>36–>82

Table B.3: Overview over all the RTEs for all the cycles registered in the second 40 kWh vehicle.

Car	Visit id	Cycle number	Cycle RTE [%]	SoC range in cycle [%]
nissan_leaf_40_2	1	1	78.52	77–>91–>77
	2	1	110.83	76–>91–>77

Table B.4: Overview over all the RTEs for all the cycles registered in the third 40 kWh vehicle.

Car	Visit id	Cycle number	Cycle RTE [%]	SoC range in cycle [%]
nissan_leaf_40_3	1	1	44.14	52–>58–>52

Table B.5: Overview over all the RTEs for all the cycles registered in the first 62 kWh vehicle.

Car	Visit id	Cycle number	Cycle RTE [%]	SoC range in cycle [%]
nissan_leaf_62_1	1	1	117.75	29–>36–>29
	1	2	91.59	29–>52–>29
	1	3	78.00	28–>21–>28
	1	4	84.98	51–>66–>51
	1	5	88.07	52–>63–>52
	1	6	84.28	51–>31–>51
	1	7	83.28	59–>76–>59
	1	8	93.86	59–>71–>59
	1	9	88.19	58–>41–>58
	1	10	83.67	69–>85–>69
	1	11	89.86	70–>80–>70
	1	12	87.88	69–>51–>69
	1	13	86.60	78–>91–>78
	1	14	98.97	78–>89–>78
	1	15	87.01	77–>58–>77
	1	16	84.86	78–>91–>78
	1	17	89.10	79–>89–>79
	1	18	88.87	78–>58–>78
	1	19	87.40	79–>91–>79
	1	20	86.62	90–>59–>90
	1	21	100.21	91–>91–>91
	1	22	92.62	90–>78–>90
	2	1	79.29	71–>89–>71
	2	2	91.67	79–>91–>79
	2	3	82.63	80–>89–>80
	2	4	90.81	79–>59–>79
	2	5	73.53	80–>91–>82
	3	1	79.86	62–>71–>62
	3	2	91.54	61–>41–>61
	3	3	81.19	67–>84–>67
	3	4	83.38	70–>79–>70
	3	5	99.87	67–>51–>67

Table B.6: Overview over all the RTEs for all the cycles registered in the second 62 kWh vehicle.

Car	Visit id	Cycle number	Cycle RTE [%]	SoC range in cycle [%]
nissan_leaf_62_2	1	1	91.85	36–>64–>36
	1	2	98.29	73–>62–>73
	1	3	88.92	73–>42–>73
	1	4	122.86	75–>86–>75
	1	5	87.49	73–>62–>73
	1	6	67.46	76–>91–>79
	1	7	82.97	79–>90–>79
	1	8	89.90	78–>59–>78
	2	1	139.09	85–>79–>85
	2	2	84.65	85–>53–>85
	2	3	68.36	88–>91–>88
	2	4	131.44	85–>79–>85
	3	1	2.51	85–>91–>86
	4	1	83.63	91–>58–>91
	4	2	120.52	91–>78–>89
	4	3	80.57	89–>70–>89
	4	4	96.36	90–>91–>90
	4	5	87.22	89–>77–>89

Table B.7: Overview over all the RTEs for all the cycles registered in the third 62 kWh vehicle.

Car	Visit id	Cycle number	Cycle RTE [%]	SoC range in cycle [%]
nissan_leaf_62_3	1	1	91.45	29–>52–>29
	1	2	81.63	28–>22–>28
	1	3	95.06	52–>66–>52
	1	4	89.54	52–>62–>52
	1	5	82.89	51–>31–>51
	1	6	91.70	61–>77–>61

Table B.8: Overview over all the RTEs for all the cycles registered in the fourth 62 kWh vehicle.

Car	Visit id	Cycle number	Cycle RTE [%]	SoC range in cycle [%]
nissan_leaf_62_4	1	1	95.44	91–>50–>91
	1	2	87.81	86–>77–>86
	1	3	83.87	86–>57–>86
	1	4	26.28	88–>91–>91
	2	1	83.46	78–>91–>78
	2	2	90.60	78–>88–>78
	3	1	82.45	75–>90–>75
	3	2	88.45	75–>85–>75
	3	3	89.77	73–>56–>73
	3	4	87.04	79–>91–>79

Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway