Martin A. Ruby, BSc.

# Design and Implementation of an Adaptive Peak Shaving Algorithm on an ARM-based Embedded System

**MASTER'S THESIS**

to achieve the university degree of

Diplom-Ingenieur, DI

Master's degree programme: ICE - Information and Computer Engineering

submitted to

**Graz University of Technology**

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger

Advisor

Dipl.-Ing. Dr.techn. Kurt Gschweitl

Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger

Institute of Technical Informatics

Graz, January 2021

# Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

| | |
|---|---|
| Date | Signature |

# Abstract

Many industrial companies cause fluctuating loads in everyday operations, such as launching production plants, heating furnaces, or starting pumping processes. These load spikes, so-called peaks, are relevant for both grid stability and electricity procurement costs and are therefore recorded by the power measurement mechanism.

At any given time, the power consumption in a network and the power plants' production must be exactly the same. The higher the power demand is at any one time, the greater the load on the distribution networks gets. For this reason, energy suppliers penalize high power consumption with higher costs. In addition to the energy consumed, a power-related charge is applied. Therefore, the grid operator determines the demand charges based on the maximum average power consumption over a given time interval (usually 15 minutes). In Austria, this mainly affects companies and industry. Utilizing so-called peak shaving, the power drawn from the grid can be reduced. An energy storage system buffers power peaks and does not draw them from the grid. In this way, the power-related costs can be reduced.

In the present work, different approaches for peak shaving were tried, evaluated, and analyzed based on an available power profiles over the period of a year. When selecting the algorithms, it was also taken into account that they can be executed on an ARM-based embedded system which limits the computing power and memory capacity.

In the end, a working, universally applicable algorithm was developed, consisting of both a rigid analytical part and a flexible, responsive part. With its help, the power-related electricity costs could be reduced by over 35%.

# Zusammenfassung


Viele Industrieunternehmen verursachen schwankende Lasten im Alltagsbetrieb, beispielsweise beim Anfahren von Produktionsanlagen, beim Aufheizen von Öfen oder beim Starten von Pumpvorgängen. Diese Lastspitzen, die sogenannten Peaks, sind sowohl für die Netzstabilität als auch für die Strombezugskosten relevant und werden daher durch eine kontinuierliche Leistungsmessung erfasst. Daraus ermittelt der Netzbetreiber anhand des maximalen mittleren Leistungsbezugs über ein Mittelungsintervall von üblicherweise 15 Minuten die Netznutzungsentgelte.

“Peak Shaving” bezeichnet das Glätten von Lastspitzen (Peaks) bei industriellen und gewerblichen Stromverbrauchern, um das Netz zu entlasten und die Stromkosten zu senken. In der vorliegenden Arbeit wird ein elektrischer Energiespeicher verwendet, um Lastspitzen zu puffern und nicht vom Netz zu beziehen. Damit ist es möglich, die vom Netzbetreiber vorgeschriebenen leistungsbezogenen Kosten deutlich zu senken.

Auf Basis von zur Verfügung gestellten Datensätzen über den Zeitraum eines Jahres wurden verschiedene Ansätze des Peak Shavings evaluiert und optimiert. Dabei wurde stets darauf geachtet, dass die notwendigen Rechenoperationen auf einem ARM-basierten eingebetteten System ausgeführt werden können, dessen Rechenleistung entsprechend eingeschränkt ist.

Es resultiert ein universell einsetzbarer Algorithmus, der die Vorteile eines starren analytischen Teils und einer flexiblen reaktiven Komponente vereint. Durch dessen Anwendung konnten in Simulationen die leistungsbezogenen Stromkosten um über 35% gesenkt werden.

# Contents

# List of Figures

# List of Tables

# 1 Introduction and Motivation

Renewable energy production plays an increasingly important role in today's world. Due to the growing population and the associated steadily increasing energy demand, the electricity network operators' infrastructure is also under more significant strain and sooner or later reaches its limits.

The use of peak shaving smoothes the power demand and thus relieves the power distribution networks. Peak shaving brings advantages for the network operator and the end customer and not at least the environment. By changing the amount of power drawn from the grid over time, electricity costs are lowered for the end customer. Reducing the power peaks also protects the environment since the network operator often covers short-term demand peaks by switching on gas turbines.

There are different ways to perform peak shaving. However, using an energy storage system in the form of rechargeable batteries is most reasonable, as long as one cannot or does not want to impact the electrical loads. Furthermore, the investment costs for battery storage systems have been decreasing significantly for several years, making their use attractive from an economic perspective as well. In many cases, an existing photovoltaic system already provides a battery storage unit that can be additionally used for peak shaving without any extra investment costs.

Power generation and power consumption are in a sensitive balance in the power grid. At any given time, demand must exactly meet production. It is obvious that irregular power profiles impede this balance. For this reason, larger electricity consumers such as corporate or industrial customers (private households are excluded in Austria) are charged a power-demand-dependent component on their electricity bill in addition to the energy consumed, which penalizes irregular load profiles. The use of peak shaving flattens the load profile from the grid operator's perspective and thus eases the load on the distribution grid. This results in financial advantages for the end customer due to the reduction of demand-related electricity costs.

The aim of this thesis is to design an algorithm that implements peak shaving and thus effectively reduces power-related electricity costs. It has to be considered that all computational operations are performed on an embedded system, and therefore the computing power is limited. For the development of the algorithm, the records of a smart meter of one facility in Styria were provided. The data set contains the consumed energy of one year resolved into quarter-hourly time intervals. Based on this data, a generally applicable strategy should be developed that is independent of this data.

# 2 Related Work

## 2.1 Background

Due to the steadily increasing population, industry, and commerce, the demand for energy is increasing steadily. [Rah+13] For the energy supply to work, production and demand must be in exact balance at any given point in time. The electricity demand varies throughout the day, which leads to load peaks from time to time. Especially the industry and large electricity consumers often have strongly fluctuating load profiles. For example, the heating of furnaces, the start-up of production machines, or similar processes require high power within a relatively short time. These so-called load peaks put a strain on the infrastructure distributing electricity from the power plant to the end-user, which is why they are penalized by higher costs. Power consumption is measured and averaged over a time interval of typically 15 minutes. The maximum of these averages per billing period is used for the billing of the demand price. These demand costs can account for up to 50% of the total electricity bill.[OCB07] A detailed explanation of how the electricity bill is composed can be found in chapter 2.1.3 on page 6.

Avoiding the load peaks, therefore offers great potential for cost savings. 'Peak Shaving' or 'Peak Load Shaving' denotes the process of flattening the load curve by reducing the peak amount of load and shifting it to times of lower load. [NKS08] According to Uddin et al. [Udd+18], there are three basic methods to achieve this goal:

- **Demand Side Management (DSM):** Load peaks are avoided by shutting down consumer processes. The serious disadvantage of this method is that the production process is affected and slowed down by switching off or throttling systems.

- **Integration of Energy Storage System (ESS):** When electricity demand is high (peak hours) the load peaks are not drawn from the grid but from a company's own battery storage (Battery Energy Storage System (BESS)). During off-peak hours the BESS gets charged again.

- **Integration of Electric Vehicle (EV):** Instead of a battery storage owned by the company, the ESS from an electric vehicle is used for load leveling.

DSM is not considered in this thesis because peak shaving should not restrict the customer's daily processes and power demands. The integration of an Electric Vehicle (EV) would be possible. However, the capacity of the battery is constrained, which is why this work only deals with the integration of an external energy storage.

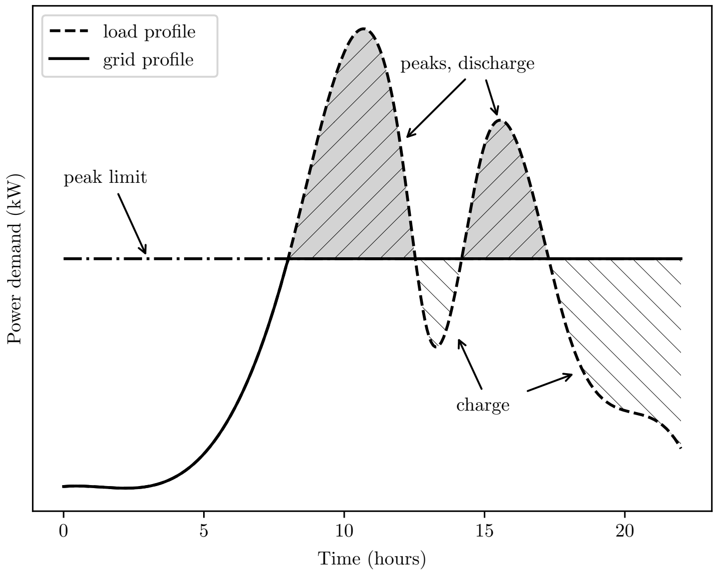



Figure 2.1: Basic principle of peak shaving with BESS.

Figure 2.1 shows a simplified load profile and illustrates the idea of BESS based peak shaving. If the load demand is higher than a specific threshold, the so-called 'peak limit', the power is drawn from BESS instead of the grid. However, the BESS is charged in off-peak hours if necessary. Mind the difference between load and grid profile.

### 2.1.1 Benefits of Peak Shaving

Peak Shaving has many benefits, which can generally be grouped into three categories: [Udd+18]

- **Benefits for the grid operator**

A main challenge for the grid operator is to keep a balance between electricity generation and demand. [JP15] If and only if the production meets the electricity demand, a high power quality concerning stability and voltage is given. [Udd+18] To measure the efficiency of power consumption, a Load Factor (LF) can be calculated. It is defined as

$$LF = \frac{P_{AVG}}{P_{Peak}} \cdot 100\,\% \qquad (2.1)$$

where $P_{AVG}$ denotes the average power and $P_{Peak}$ the maximum power demand in a given time interval. The higher the Load Factor (LF) is, the better gets the power plant's economic feasibility. [MM05] The highest LF is achieved when the power demand is constant, that means no peaks occur.

Besides, the power losses in the transmission line are lower if peaks are avoided since $P_{loss} = I^2 \cdot R$ and a decrease of the current $I$ means a larger decrease of the power loss $P_{loss}$ while resistant $R$ stays constant. [Udd+18]

The infrastructure for power transmission must be designed to meet the maximum projection demand (peaks). If the peaks are lower, the costs for transmission lines decrease. Furthermore, the life span of the transmission and distribution system increases.

- **Benefits for the end user**

In peak hours (mostly daytime), the end-user is usually billed a higher energy price than in off-peak hours (e.g., night time). Thus shifting power demand from daytime to nighttime results in a lower electricity bill. Furthermore, one can save on connection costs and the charges for the distribution system. [Udd+18]

Industry and companies are billed demand charges added to the energy cost. These demand charges are slightly high and can be reduced with peak shaving.

In summary, the end-user can profit from a reduction in electricity costs through peak shaving. Furthermore, he or she reduces $CO_2$ emissions and thus helps to protect the environment. This is described in more detail in the following.

- **Carbon Dioxide emission reduction**

In practice, peak loads are often supplied by fuel-powered generators. This causes higher $CO_2$ emissions, mainly because the generators are often switched on and off. Peak shaving ensures more efficient power production and thus reduces the emission of harmful carbon dioxide. [KL15]

### 2.1.2 Basic Concept of Grid connected Battery Storage

In most cases a photovoltaic (PV) system is available in addition to the battery storage. Along with peak shaving, it brings further potential for cost reduction. The PV modules generate direct current (DC), the battery storage also works with direct current. The consumers, whether they are household customers or industry, need alternating current (AC). Therefore the existing direct current must be converted into alternating current. Basically, there are two ways to achieve this and interconnect the individual components: DC-coupling and AC-coupling. The two methods are briefly explained below.

#### AC Coupling

The direct current produced by the PV modules and provided by the battery is converted into alternating current by an inverter each. Both components are coupled on the AC side. The consumers can now use the converted alternating current. A significant advantage of this method is that one or more battery storage units can easily be added to existing systems if required. The sketched wiring is shown in Figure 2.2.

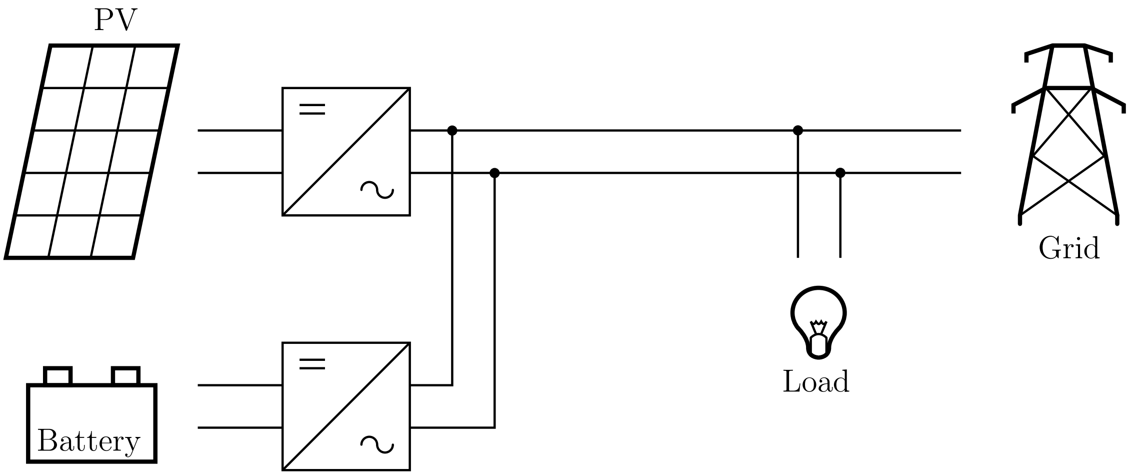



Figure 2.2: AC Coupling of PV, battery and grid.

#### DC Coupling

PV and battery are coupled on the DC side. The solar power can be directly stored in the battery (DC to DC). This avoids the losses of the DC-AC-DC conversion, which occur during AC coupling. Therefore the efficiency is higher. The used inverters are often combined in one device, the so-called hybrid inverter. Less space is required because the separate battery inverter is no longer needed. The overall system is, therefore, more cost-effective. The disadvantage is that the system is dependent on the hybrid inverter. One can only use batteries that are compatible with the hybrid inverter. Besides, the charge and discharge power is limited by the hybrid inverter. For existing systems, the storage system can only be retrofitted by replacing the existing inverter. [iEne] (See also Figure 2.3)

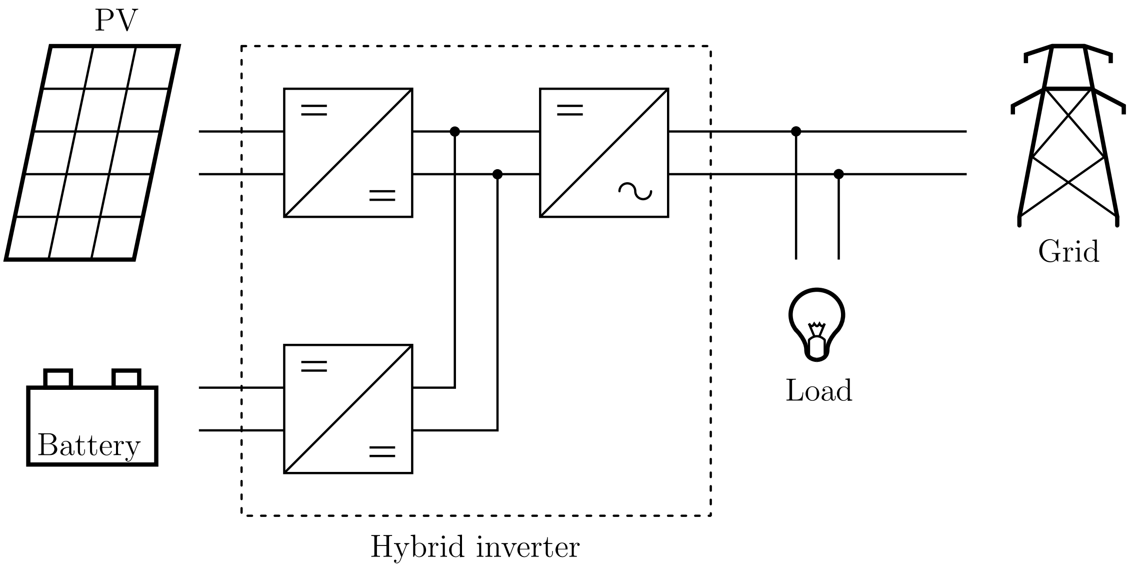



Figure 2.3: DC Coupling of PV, battery and grid.

### 2.1.3 Contents of Electricity Bill

The transmission and distribution of electrical energy takes place as efficiently and economically as possible via power grids operated at different voltages. The transmission grid supplies the large distributors' electrical energy from the large power plants via a high-voltage grid. It also serves the supraregional exchange of electricity. The distribution network, in turn, is the network to which customers are connected. Distribution refers to the transport of electrical energy with medium or low voltage to supply electricity to customers. [E-C20a] Operators are allowed to charge their customers for the services they offer. According to E-Control, the system charges are a composition of the following components: [E-C20c]

- **System Utilization Charge**: Charges for maintaining the system operators network. Usually calculated by demand (energy and power).
- **Charges for System Losses**: Costs for grid losses that occur when distributing the electricity.
- **Metering Charge**: Costs for installing and operating metering equipment.

- **System Provision Charge**: Charges to build and expand different network levels.
- **System Service Charge**: Charges for controlling offset load variations utilizing secondary control.
- **System Admission Charge**: Costs for connecting a facility to electricity network for the first time.
- **Supplementary Service Charges**: Extra charges for payment reminders, extra meter readings, etc.

The available measurement datasets were collected in Weiz in Styria. Connection to the electricity grid is located in the distribution network on grid level 7. Austrian electricity fees valid for 2020 are listed in table 2.1.

Table 2.1: Power Grid Fee and Power Loss Fee valid for Austria 2020 [E-C20b]. (LP...Demand Cost, SHT...Summer High Tariff, SNT...Summer Low Tariff, WHT...Winter High Tariff, WNT...Winter Low Tariff). All prices in ct/kWh except LP, which is in ct/kW.

| Province | Power Grid Fee | | | | | Power Loss Fee | | | |
|---|---|---|---|---|---|---|---|---|---|
| | LP | SHT | SNT | WHT | WNT | SHT | SNT | WHT | WNT |
| Burgenland | 4.536 | 3.03 | 3.03 | 3.03 | 3.03 | 0.291 | 0.291 | 0.291 | 0.291 |
| Carinthia | 7.500 | 3.30 | 1.81 | 4.14 | 1.91 | 0.393 | 0.393 | 0.393 | 0.393 |
| Klagenfurt | 5.640 | 2.71 | 2.28 | 3.20 | 2.28 | 0.286 | 0.286 | 0.286 | 0.286 |
| Lower Austria | 3.000 | 3.15 | 3.15 | 3.15 | 3.15 | 0.231 | 0.231 | 0.231 | 0.231 |
| Upper Austria | 4.116 | 3.41 | 3.25 | 3.63 | 3.33 | 0.331 | 0.331 | 0.331 | 0.331 |
| Linz | 4.200 | 2.35 | 1.30 | 2.35 | 1.30 | 0.247 | 0.247 | 0.247 | 0.247 |
| Salzburg | 4.104 | 2.24 | 2.24 | 2.24 | 2.24 | 0.228 | 0.228 | 0.228 | 0.228 |
| Styria | 4.368 | 3.84 | 3.09 | 3.84 | 3.09 | 0.315 | 0.315 | 0.315 | 0.315 |
| Graz | 3.348 | 3.47 | 2.59 | 3.47 | 2.59 | 0.384 | 0.384 | 0.384 | 0.384 |
| Tyrol | 3.948 | 2.25 | 1.60 | 2.25 | 1.60 | 0.322 | 0.322 | 0.322 | 0.322 |
| Innsbruck | 4.716 | 3.20 | 2.36 | 3.20 | 2.36 | 0.323 | 0.323 | 0.323 | 0.323 |
| Vorarlberg | 3.444 | 1.53 | 1.53 | 1.53 | 1.53 | 0.271 | 0.271 | 0.271 | 0.271 |
| Vienna | 4.548 | 1.95 | 1.95 | 1.95 | 1.95 | 0.414 | 0.414 | 0.414 | 0.414 |
| Kleinwalsertal | 8.616 | 5.38 | 5.38 | 5.38 | 5.38 | 0.301 | 0.301 | 0.301 | 0,301 |

Depending on the season and time, different energy prices apply. Between 6 a.m. and 10 p.m., the high tariff applies; from 10 p.m. to 6 a.m., the low tariff applies. Besides, a distinction is made between summer (April to September) and winter (October to March). Thus the price groups shown in table 2.1 are: SHT ('Sommerhochtarif', summer high tariff), SNT ('Sommerniedertarif', summer low tariff), WHT ('Winterhochtarif', winter high tariff), WNT ('Winterniedertarif', winter low tariff). The demand price is referred to as LP ('Leistungspreis').

To calculate the demand price, the withdrawn power is measured and averaged over a time interval of 15 minutes. The monthly maximum of these values is multiplied by a twelfth of the valid demand price and charged in the electricity bill. By reducing the maximum load using peak shaving, electricity costs can be significantly reduced.

## 2.2 State of the Art

According to Uddin et al., three aspects are essential in order to operate peak shaving in a practical way: First, the optimal operation of ESS, second, the sizing of the battery storage, and third, the economic feasibility. [Udd+18] There are several approaches and algorithms in the literature dealing with peak shaving. This chapter presents an overview of various selected concepts.

Lucas and Chondrogiannis proposed a simple method that defines two limits - one upper and one lower. If the power demand exceeds the upper limit, the battery is discharged; if it falls under the lower limit, the battery is charged. Moreover, the State of Charge (SOC) of the battery is taken into account. The battery is only charged if SOC is below 90%. Correspondingly, the battery is only discharged as long as the SOC is above 10%. This ensures that the battery is not damaged by deep discharge or overcharge. [LC16]

Leadbetter and Swan follow a similar strategy by setting a maximum grid electricity demand ('peak limit'). If the power demand exceeds the peak limit, the electricity from the grid is only drawn up to the limit. The battery supplies the rest. If the inverter is sized sufficiently, the maximum grid electricity demand remains at the limit defined before. During the night, when the power demand is minimal, the battery is fully charged. To ensure a longer lifetime and safe operation conditions, the battery is only used between a SOC-range of 10% to 85%. [LS12]

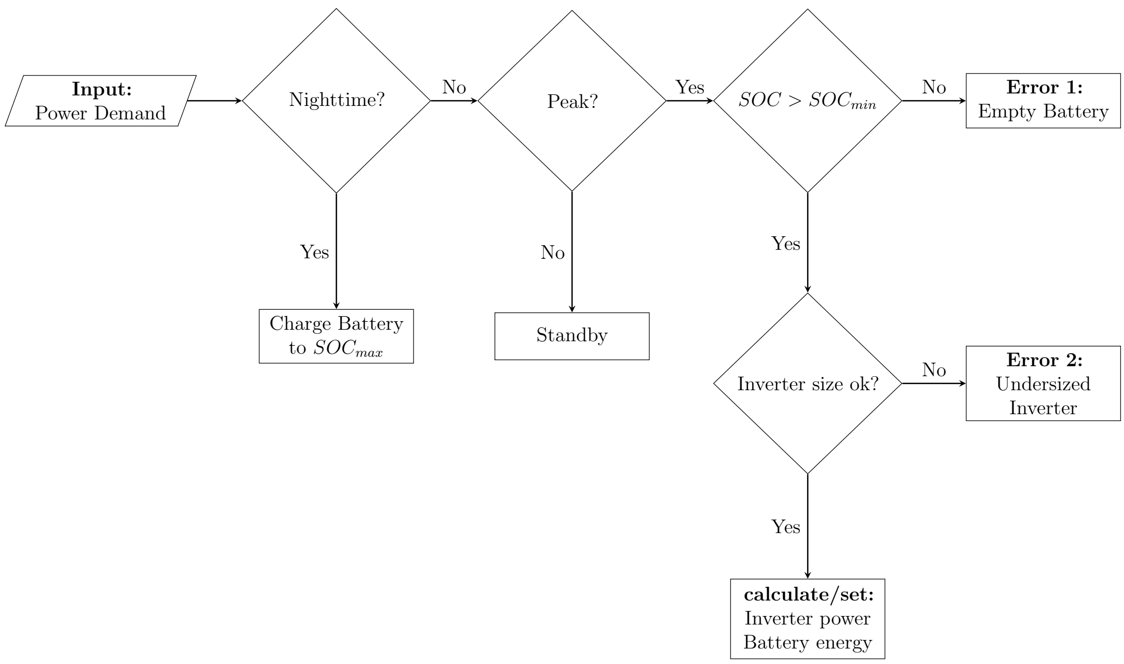



Figure 2.4: Simplified Process Flow Diagram from Leadbetter and Swan. [LS12]

Figure 2.4 shows how the proposed algorithm works. The system works in one of three different states: charging, standby, or discharging. If the SOC falls below the minimum value or the inverter is undersized, an error occurs. [LS12]

According to Leadbetter and Swan, defining the grid demand limit is the deciding factor. It is determined by statistical analysis of historical load data finding the top one to two percent of load peaks as they are the most expensive.

Concerning the dimensioning of the battery and the inverter, the three parameters peak limit, battery size, and inverter size, were varied in certain ranges. The number of occurring error states was counted. In the end, the final parameters were chosen in a way that no errors occur. It is observed that doubling the battery size increases the peak shaving capability by only 25%. [LS12]

The data used was collected from households in Canada. In their simulation, the authors achieved a peak demand reduction of 42% to 49% in all Canadian regions except Quebec (peak demand reduction of 28%). However, different parameters were required for all regions, suggesting that the load profile strongly influences the result. [LS12]

Oudalov, Cherkaoui, and Beguin proposed a BESS sizing strategy for peak shaving application maximizing the economic benefit by optimizing an objective function

$$Benefit|_{T_e} = Savings(Size, \text{operating schedule})|_{T_e} - Cost(Size, \text{operating schedule})|_{T_e} \tag{2.2}$$

where $T_e$ denotes the evaluation time and functions $Savings$ and $Cost$ are both dependent on battery size (capacity and power) and the 'operating schedule' which is subject of the used peak shaving algorithm. The objective function 2.2 can be simplified to a linear function of the battery power ($P_B$), $\Delta T$ denoting the time where the power demand exceeds the peak limit and some more parameters:

$$Benefit = P_B \cdot f(\Delta T, T_e, \ldots). \tag{2.3}$$

The benefit 2.3 then gets maximized by varying all possible input values. This assumes that the load profile is known in advance. [OCB07]

Oudalov, Cherkaoui, and Beguin also introduced an algorithm based on Dynamic Programming (DP) to find the optimal operating strategy for peak shaving. It depends on the customer load curve, electricity fees, battery sizing, and a threshold defining the peak limit. The simulation showed that this method is profitable for peaks with a length of less than about one hour. The annual electricity costs are reduced by around four percent using the DP algorithm and the BESS sizing strategy on simulation data from an industry customer. [OCB07]

In their work, Rahimi et al. presented a simple approach for peak load shaving, which is independent of the system topology and its impedances. With a sliding window of length $\tau$, the algorithm calculates the load profile average. This is used as a reference for the next utilization period $up$ ($up > \tau$). According to a utilization factor ($uf$) with $0 \leq uf \leq 1$, which defines the percentage of battery usage over time, a peak limit is derived. This method reasonably compensates errors in load forecasting due to the sliding window and the repeated calculation of the average power demand. The aim is to adapt the load curve to its mean value as closely as possible. Simulation with different battery sizes resulted in an average load reduction of about two to six percent. [Rah+13]

A method to perform peak shaving in a grid-connected photovoltaic battery system is documented in [BMH18]. This method aims to reduce daily peaks to a defined target power. As the load changes from day to day, the peak threshold gets changed based on historical load data, the average SOC during a given time interval, and the current time. During the day, the battery is used to flatten the load profile while charged in the night hours with constant power. An average load reduction of about 15% is achieved using this method at Helmholtz Institute Ulm (HIU). [BMH18]

While other methods are based on complex arithmetic operations, Ananda-Rao et al. offers a strategy implemented on a microcontroller and therefore can operate with few resources. Peak shaving is done with two limits defined in advance, one upper and one lower bound. [Ana+16] Similar to the algorithm in [LC16], the battery is charged if power demand is less than the lower bound and respectively discharged when power demand exceeds the upper bound. Furthermore, a simple SOC monitoring by the cyclic reading of the battery voltage is implemented to prevent the BESS from under-discharge and overcharge. Unfortunately, no real-world simulation data were provided to show potential savings through peak shaving using this method. [Ana+16]

# 3 Design

This thesis aims to develop an algorithm that allows effective peak shaving, resulting in cost reduction on the electricity bill. This algorithm shall be executable on an embedded system based on a BeagleCore. For this reason, complex, computationally, and memory-intensive processes have to be avoided. For the development, the smart meter load profiles were provided by Innovationszentrum Weiz [1] for a whole year. With the help of these data sets, the algorithm is developed and tested.

In the following sections, the general use cases and requirements are explained. The data sets are analyzed, and a simulation environment is prepared to test and compare different approaches.

## 3.1 General Setup

The setup consists of three essential elements, which are shown in Figure 3.1. The building blocks are a smart meter, which captures the current power demand and provides it via different interfaces, the inverter, which monitors the battery via the battery controller and regulates its input and output power, and the EEM, which is the central control unit and executes the peak shaving algorithm. In addition, the EEM provides a web interface that allows a user to monitor all processes.

The EOS Energy Manager runs a software called `macchina.io` [OO], which provides special interfaces for different applications, so called 'bundles'. All bundles have to follow a specific structure to use the provided interfaces. Therefore, the peak shaving algorithm must meet the specifications of a bundle. Detailed information about `macchina.io` and bundles can be found in Section 4.3.1 on page 52.

[1] `http://www.innovationszentrum-weiz.at/`

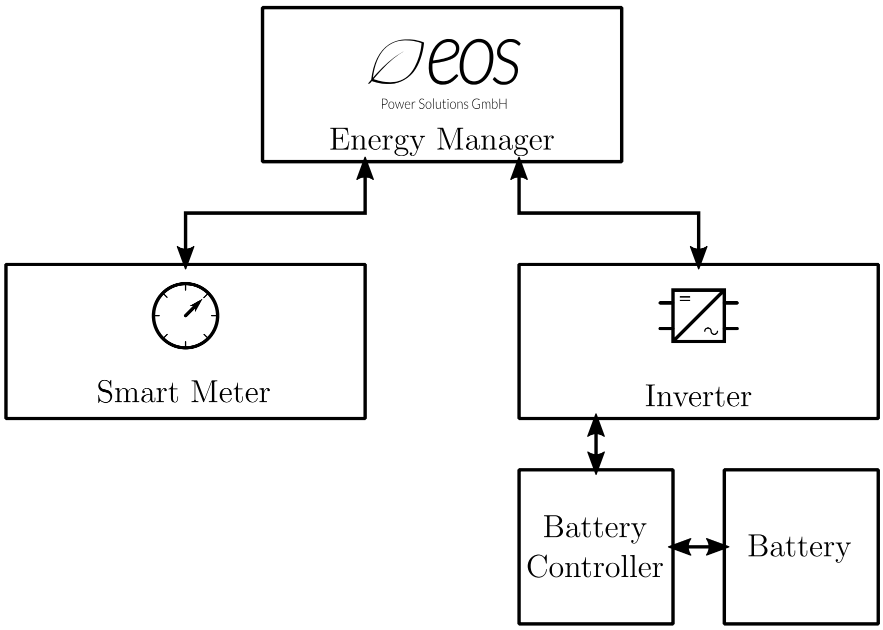



Figure 3.1: General setup of sensors and actuators.

## 3.2 Use Cases

In the use case diagram (Figure 3.2) the three already mentioned basic components (EEM, smart meter, inverter+battery) can be seen. A user has to activate the Peak Shaving Bundle to execute the algorithm. At regular intervals, the EEM reads the current power demand via the smart meter and processes the data received. Additionally, the current battery status is read. Based on this data, a peak limit is calculated, which is valid and applied for the next time interval. This results in a power that is drawn from the battery via the inverter and supplied to the load circuit.

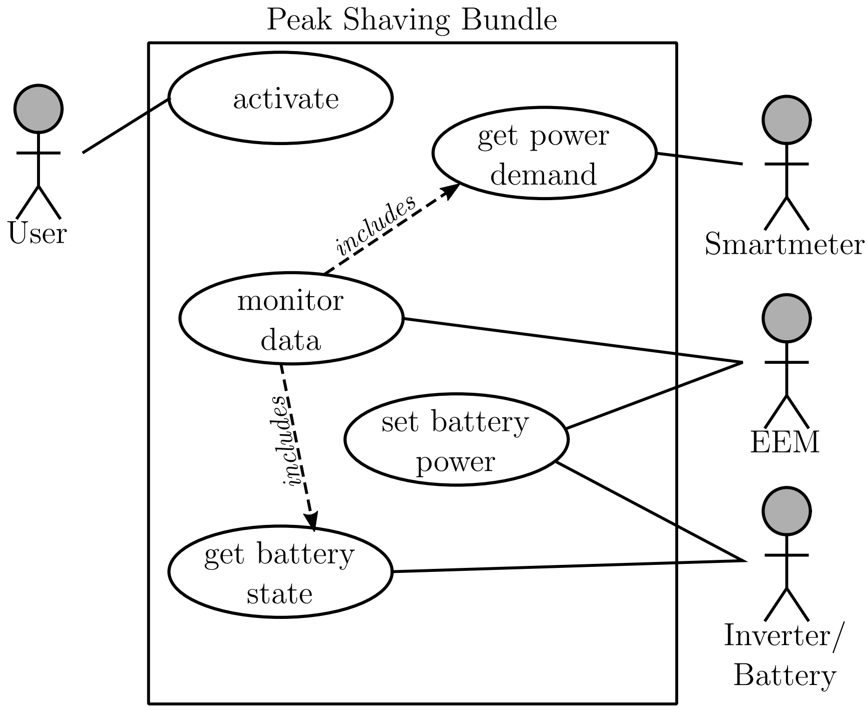



Figure 3.2: Use Case Diagram: Peak Shaving Application.

## 3.3 Requirements

The goal is to create a so-called bundle for the SDK `macchina.io`, which executes the developed peak shaving algorithm. Since the software must be executable on an embedded system, and the computing power is limited, computationally intensive operations must be omitted. Power demand profile data over the period of one year was provided for the development. However, the algorithm should work independently of the provided dataset.

## 3.4 Data Analysis

The energy consumption data for 2018, measured using a smart meter, have been provided for research purposes. The energy consumption in kilowatt-hours (kWh) was averaged over fifteen-minute time intervals and stored with a time stamp. The energy values are now converted into power values by means of the correlation

$$W = P \cdot t \;\;\Rightarrow\;\; P = \frac{W}{t}. \tag{3.1}$$

$W$ denotes the measured energy consumption in kWh, $t$ the length of the time interval in hours (h) and $P$ represents the power demand in kilowatt (kW). Thus the data is now available as shown in Table 3.1.

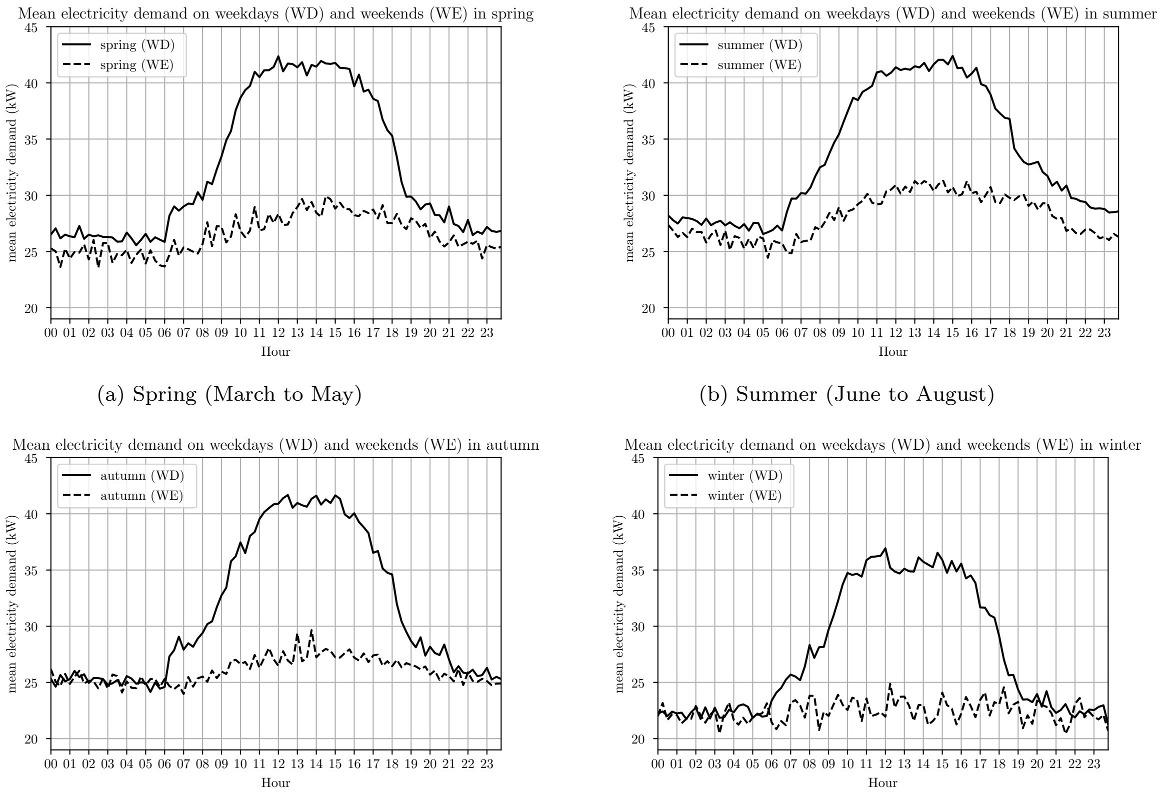
Table 3.1: Available dataset consisting of timestamp, energy- and power demand.

| Timestamp | Energy (kWh) | Power (kW) |
|---|---|---|
| 2018-01-01 00:15 | 7.59 | 30.36 |
| 2018-01-01 00:30 | 5.49 | 21.96 |
| 2018-01-01 00:45 | 8.67 | 34.68 |
| 2018-01-01 01:00 | 5.01 | 20.04 |
| 2018-01-01 01:15 | 8.79 | 35.16 |
| ⋮ | ⋮ | ⋮ |
| 2018-12-31 22:45 | 4.86 | 19.44 |
| 2018-12-31 23:00 | 4.14 | 16.56 |
| 2018-12-31 23:15 | 3.87 | 15.48 |
| 2018-12-31 23:30 | 3.87 | 15.48 |
| 2018-12-31 23:45 | 4.11 | 16.44 |

An initial overview shows what is living up to expectations: energy consumption during the night hours is significantly lower than during the day. In principle, the data sets can be assigned to two basic patterns. On weekdays, the energy demand increases rapidly in the morning, remains relatively constant throughout the day, and decreases again towards evening. On weekends or holidays, the morning's increase is not clearly visible as energy consumption remains at about the same level throughout the day. It is noticeable that there is a not negligible difference in the seasons. While the electricity demand is relatively similar in spring, summer, and autumn, significantly less energy is required during winter. Figure 3.3 shows the averaged energy demand over the four seasons and the differences between working day and weekend. It can be seen that the electricity demand starts to rise from 6 a.m. on and is back at night level from about 10 p.m. This also corresponds to the electricity tariffs listed in Table 2.1 on page 7.

## 3.5 Electricity Bill

In order to evaluate the savings of different peak shaving algorithms, it is first necessary to calculate the electricity cost of the existing data without using peak shaving. Since peak shaving only changes the variable costs, ground fees and other similar charges remain the same and are therefore not considered in the calculations.




(a) Spring (March to May)

(b) Summer (June to August)

(c) Autumn (September to November)

(d) Winter (December to February)

Figure 3.3: Mean electricity demand in different seasons showing differences between weekdays and weekends.

### 3.5.1 Electricity Cost Calculation

In Table 3.2 the charges used for the calculation are shown. The variable electricity costs are calculated from the sum of the energy rate, demand cost and loss compensation fee.

Table 3.2: Variable electricity charges valid for Styria. [Dig20]

| Energy Cost (ct/kWh) | | | | Loss Fee (ct/kWh) | Demand Cost (ct/kW) |
|---|---|---|---|---|---|
| Summer | | Winter | | | |
| High tariff | Low tariff | High tariff | Low tariff | | |
| 3,84 | 3,09 | 3,84 | 3,09 | 0,315 | 4368 |

#### Energy Rate

The energy consumed is measured and stored by smart meter. The consumed kilowatt-hours are billed according to season (summer/winter) and time (high/low tariff).

#### Demand Rate

The demand price in Table 3.2 is indicated as an annual price but is charged monthly on a pro-rata basis. The load curve over time is recorded by the smart meter. In time intervals of 15 minutes the energy consumed ($W$) is determined as

$$W = \int P(t) \cdot dt, \tag{3.2}$$

by integrating the power values ($P(t)$) over time ($t$) as stated in Equation 3.2. Then the statistical power

$$\overline{P} = \frac{W}{t} \tag{3.3}$$

for this period ($t = 0.25\,h$) is calculated by dividing the calculated energy by the period length as shown in Equation 3.3. These calculated quarter-hourly power values are collected per month. The highest value is used for calculating the demand charge by multiplying it with a twelfth of the annual demand price rate.

### 3.5.2 Electricity Cost without Peak Shaving

Table 3.3 shows the variable electricity charges for the given dataset. It can be seen that power demand costs account for about 20% of the total costs. Each month the highest peak (stated in Table 3.3 in column 'max. Power (kW)') is considered for the calculation of the demand charges. This is where peak shaving becomes valuable in order to reduce costs. Due to physical constraints, demand price cannot be reduced to zero. The overall savings achieved by using peak shaving are less than 20% of the total costs.

Table 3.3: Electricity cost without peak shaving.

| Month | Energy (kWh) | max. Power (kW) | Energy Fee (€) | Loss Fee (€) | Demand Fee (€) | Total (€) |
|---|---|---|---|---|---|---|
| 1 | 18 161.10 | 61.04 | 659.65 | 57.21 | 222.19 | 939.05 |
| 2 | 19 875.09 | 71.04 | 721.89 | 62.61 | 258.59 | 1043.09 |
| 3 | 20 449.71 | 52.72 | 740.11 | 64.42 | 191.90 | 996.43 |
| 4 | 23 193.00 | 62.52 | 843.06 | 73.06 | 227.57 | 1143.69 |
| 5 | 24 159.12 | 59.60 | 876.98 | 76.10 | 216.94 | 1170.02 |
| 6 | 23 031.93 | 60.28 | 834.77 | 72.55 | 219.42 | 1126.74 |
| 7 | 23 594.40 | 59.32 | 857.05 | 74.32 | 215.92 | 1147.29 |
| 8 | 23 990.13 | 55.92 | 868.82 | 75.57 | 203.55 | 1147.94 |
| 9 | 23 351.70 | 56.88 | 846.68 | 73.56 | 207.04 | 1127.28 |
| 10 | 22 355.22 | 56.84 | 811.74 | 70.42 | 206.90 | 1089.06 |
| 11 | 19 542.66 | 56.72 | 709.25 | 61.56 | 206.46 | 977.27 |
| 12 | 18 351.66 | 50.92 | 663.75 | 57.81 | 185.35 | 906.91 |
| Σ | 260 055.72 | - | 9433.75 | 819.19 | 2561.83 | 12 814.77 |

## 3.6 Simulation Environment

In order to develop and evaluate different algorithms, a simulation environment was designed, which makes it possible to read the available data sets and prepare them in such a way that easily exchangeable algorithms can use them. The individual algorithms receive the data sets in a special format, perform simulated peak shaving based on the data and store the results in a way allowing the simulation environment to evaluate them. The expected electricity costs are calculated and printed in tabular form. For a better overview, an interactive report is generated. Besides the cost table, a chart shows the curves of all relevant data over the simulated time period. This includes not only the power demand but also the battery power, the charge level and the resulting power drawn from the grid. Depending on the requirements, individual variables can be easily added or removed in the report. Figure 3.4 shows the flow of the data in the simulation environment.

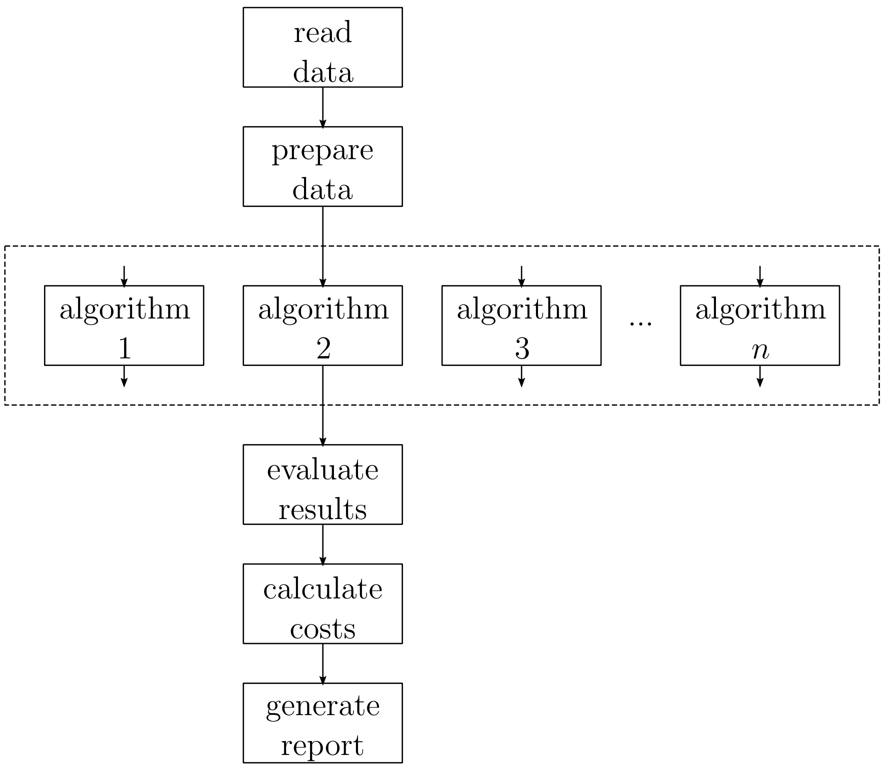



Figure 3.4: Process diagram of simulation environment.

# 4 Implementation

In this chapter, the implementation of the simulation environment and the individual algorithms is explained in detail. Furthermore, it explains how an algorithm can be integrated into the EOS Energy Manager and which limitations have to be considered concerning the embedded system's hardware and software. At the end of this chapter, the finished peak shaving bundle, including its visualization, is presented.

## 4.1 Simulation Environment

The simulation environment is programmed in Python (version 3.7.6) and uses the packages and dependencies shown in Table 4.1.

Table 4.1: Used python packages for simulation environment.

| Name | Version | Short description |
|---|---|---|
| numPy | 1.18.1 | Open source package for scientific computing. [Har+20] |
| matplotlib | 3.1.3 | 2D graphics package for application development, interactive scripting and publication-quality image generation. [Hun07] |
| pandas | 1.0.1 | Flexible open source data analysis and manipulation tool. [tea20] |

The data provided is available as an Excel spreadsheet (`*.xlsx`) and was exported into CSV format (`*.csv`) for better programmatic processing. Pandas natively support the reading of CSV files, whose content is then available to the programmer as a so-called `DataFrame`, the primary data structure used in Pandas. That is a tabular data structure allowing searching, indexing, grouping, and more efficiently.

The simulation environment essentially consists of three modules: the simulation control unit, the peak shaving algorithm, and the utility module. These are briefly described in the following subsections.

### 4.1.1 Utility Module

The utility module provides functions that are necessary for the internal calculations. Recurrently called functions are calculateSOC and eta. `CalculateSOC` calculates, as the name suggests, the current charge level of the battery based on the available energy and the maximum capacity. The function `eta` calculates an overall efficiency for the battery and the inverter based on the inverter size and the current inverter power. The efficiency curve shown in Figure 4.1 was used for this purpose. It was taken from the data sheet of an inverter manufacturer and used as an approximate guide value. [Fro20]

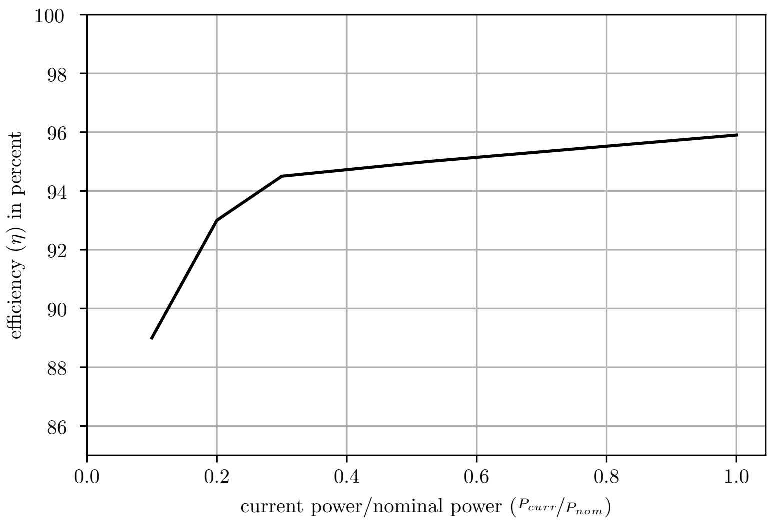



Figure 4.1: Battery/Inverter efficiency graph. [Fro20]

### 4.1.2 Algorithm Module

The algorithm module contains the logic and the calculations for peak shaving. It consists of the two large function blocks `simulate` and `simulateFrame`.

#### Simulate

The function `simulate` contains the essential logic, which determines the needed inverter power from the given peak limit and the current power demand and calculates as well as manages the remaining energy in the battery. Additionally, error checks are performed. Two errors can occur. On the one hand, it is possible that the battery has not stored enough energy anymore to cover the demand ($Err_{01}$), on the other hand, the necessary inverter power can be higher than the maximum output power supported by the inverter ($Err_{02}$). After each call, a data set is returned consisting of the energy still available in the battery, and the two error indicators $Err_{01}$ and $Err_{02}$. This function has to be called in chronological order for each existing data set.

#### Simulate Frame

The `simulateFrame`-function contains the outer logic and the calculations around the peak limit. It iterates through the given dataset and calls the `simulate`-function for each dataset, evaluating the return values. For the analysis, all values and parameters necessary for the algorithm such as power demand, peak limit, available battery energy, SOC, inverter power, grid power, and similar are stored in a separate DataFrame in order to be able to track the calculations and decisions made as a result. However, the recorded values are highly dependable on the used algorithm, so the resulting DataFrame looks slightly different for various algorithms.

### 4.1.3 Simulation Control Module

This module controls the simulation process and manages the simulation data. Here all physical conditions like battery capacity, inverter size, minimum and maximum charge of the battery, and suchlike are defined, which are not changed during the whole simulation process. In the following, the run of the simulation starts by calling `simulateFrame`. The prepared data sets and all parameters are passed. At the end of a complete run, which can take between a few minutes or even hours depending on the complexity of the used algorithm, the DataFrame containing the statistics is ready. With this statistical data, the function `calculateCost` is called, which calculates the expected electricity costs based on the price composition described in Section 3.5 on page 16 and returns them broken down as a tabular DataFrame showing all components separately. From the two generated DataFrames, a report is created, which contains an interactive graphic of all graphs showing the statistical data as well as a list of the costs. This report can be opened and viewed with a browser.

## 4.2 Algorithms

In the course of this thesis, various algorithms, some of them based on or extending approaches from literature research, will be tested and evaluated. The resulting cost savings measure the performance of each method. The better a peak shaving algorithm works, the lower the final electricity costs compared to the previously calculated invoice amount without applying peak shaving.

In general, the algorithms presented here can be divided into two major categories: Static and adaptive approaches. Static approaches are very often described in the literature. They analyze an existing past load profile and determine a peak limit that is valid and used for future profiles. The peak limit is not changed anymore and is therefore 'static'. Adaptive or dynamic approaches are less frequently treated in the literature. They are characterized by the ability to change the peak limit over time and adapt it to the current demand and load profile.

At the end of this chapter, a combined algorithm is presented that combines both an analytical and a responsive part and thus allows the determination of the peak limit for each month 'live', *i.e.* only depending on data collected at runtime and without having to analyze large amounts of data in advance.

In order to obtain the most comparable results of the algorithms, they are all tested based on the same specifications. 'Sufficiently large' values are chosen for battery capacity and inverter size.

**Battery Capacity** $E_{Batt} = 233\,kWh$ used within a range of $SOC_{min} = 1\%$ and $SOC_{max} = 99\%$

**Inverter Power (nominal)** $P_{inv,max} = 88\,kW$

Both types of algorithms are described in detail in the following sections.

### 4.2.1 Static Algorithms

For static algorithms, the peak limit is determined mathematically in advance based on past load data and then applied. Therefore it is not changed anymore. This has the great advantage that the demand costs can be calculated beforehand, making it easier to estimate the investment costs and the payback period. Research on the internet shows that there are some providers in Europe who offer peak shaving equipment based on static algorithms. Usually, a previous year's load profile is sent to the company, analyzed, and a peak limit, including the necessary hardware specification (battery capacity, inverter size), is created. Subsequently, all evaluated static algorithms are explained and evaluated based on their potential to reduce electricity costs.

**First Simulation Approach**

Based on the algorithm presented by Leadbetter and Swan in [LS12] an optimal solution should be found for the provided load profile from Weiz. Figure 4.2 shows the slightly adapted flowchart of the used algorithm.

The condition `Nighttime?` checks if the time of the current data sample is between 10 p.m. and 6 a.m. and charges the battery if that is the case. Thereby, the current SOC is considered, and the battery is charged up to $SOC_{max}$. If the current time is outside of the night range, the algorithm checks for peaks. Assuming that a peak limit is defined in advance, a peak occurs if the current power demand $P_{dem}$ exceeds the peak limit $P_{lim}$ ($P_{dem} > P_{lim}$). As long as no peak is present, the algorithm ends, and the process stays in idle mode. However, otherwise after checking if the battery is not empty and therefore no error is thrown (`Error 01: Battery empty`), the necessary inverter power ($P_{inv}$) is calculated using equation 4.1.

$$P_{inv} = P_{dem} - P_{lim} \tag{4.1}$$

In any case, the calculated inverter power must be lower than the maximum inverter power ($P_{inv,max}$). If not, the process enters a failure state (`Error 02: Undersized Inverter`).

To find the optimal peak limit, simulation is done multiple times, varying the peak limit in a range between 26 $kW$ and 40 $kW$. Table 4.2 shows the results for the different peak limits used including price components.

Table 4.2: Results for different peak limits.

| $P_{lim}$ (kW) | #Err01 (.) | #Err02 (.) | Energy Cost (€) | Loss Fee (€) | Demand Cost (€) | Total (€) |
|---|---|---|---|---|---|---|
| 26 | 13454 | 0 | 9436.80 | 823.46 | 2598.51 | 12 858.77 |
| 28 | 10221 | 0 | 9435.85 | 825.05 | 2521.64 | 12 782.54 |
| 30 | 6358 | 0 | 9433.18 | 826.71 | 2428.18 | 12 688.07 |
| 32 | 3202 | 0 | 9429.23 | 827.68 | 2295.40 | 12 552.31 |
| 34 | 1263 | 0 | 9426.82 | 827.59 | 2179.21 | 12 433.62 |
| 36 | 283 | 0 | 9426.66 | 826.56 | 2007.97 | 12 261.19 |
| 38 | 0 | 0 | 9427.07 | 824.80 | 1766.44 | 12 018.31 |
| 40 | 0 | 0 | 9428.20 | 823.08 | 1811.13 | 12 062.41 |

Figure 4.3 shows the plot of peak limit over the total cost. It is clearly visible that the peak limit resulting in the lowest cost is 38 $kW$. At this limit, neither Error 01 nor Error 02 occurs. A closer look at the evaluation of the electricity costs in Table 4.3 with that peak limit shows a saving of € 796.46 compared to the electricity bill without peak shaving. However, it is noticeable that despite the peak limit of 38 $kW$, the highest peaks in most months are to some extent significantly higher.

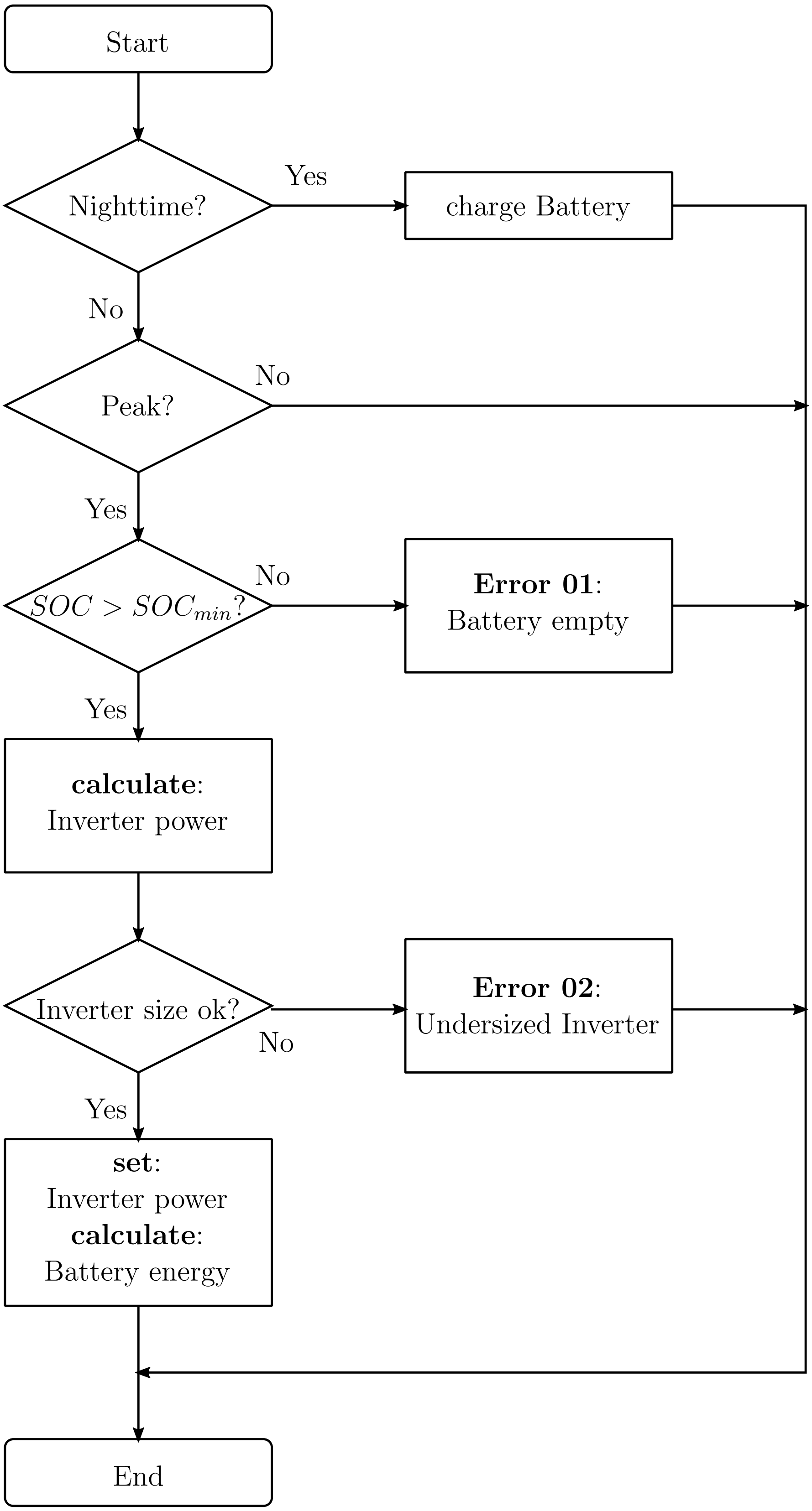



Figure 4.2: Flowchart of Algorithm 1. [LS12]

The evaluation of the data shows that the high peaks all occur at night. Occasionally, peaks also occur at night, although the general power demand during the night is relatively low. Since the algorithm used does not react to peaks between 10 p.m. and 6 a.m. but only charges the battery, these peaks are used to calculate the power price. Due to these circumstances, the algorithm must be adapted in such a way that it also reacts to possible peaks during the night.

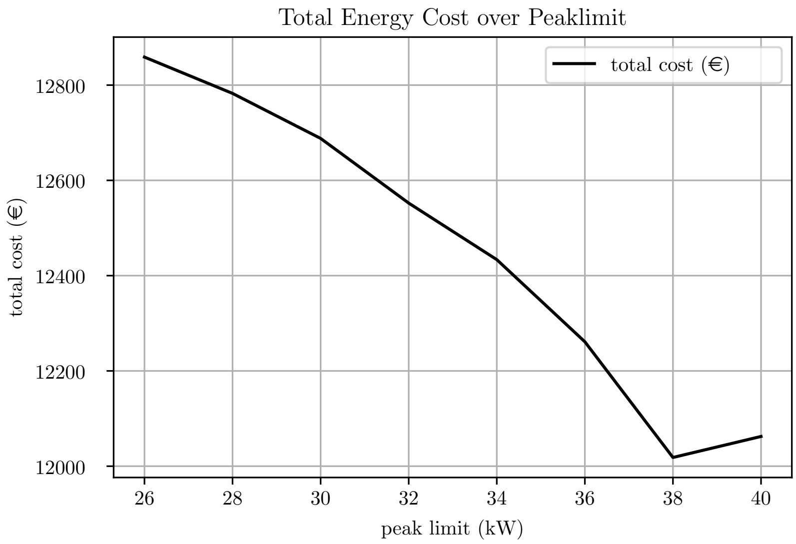



Figure 4.3: Total energy cost over peaklimit (simulation 1).

Table 4.3: Evaluation of electricity cost using first simulation approach with a peak limit of 38 $kW$.

| Month | Energy (kWh) | max. Power (kW) | Energy Fee (€) | Loss Fee (€) | Demand Fee (€) | Total (€) |
|---|---|---|---|---|---|---|
| 1 | 18 206.69 | 41.28 | 659.30 | 57.35 | 150.26 | 866.91 |
| 2 | 20 010.96 | 51.12 | 720.34 | 63.03 | 186.08 | 969.45 |
| 3 | 20 535.31 | 39.12 | 740.71 | 64.69 | 142.40 | 947.80 |
| 4 | 23 365.46 | 40.92 | 837.88 | 73.60 | 148.95 | 1060.43 |
| 5 | 24 452.45 | 40.56 | 877.03 | 77.03 | 147.64 | 1101.70 |
| 6 | 23 244.19 | 38.28 | 836.58 | 73.22 | 139.34 | 1049.14 |
| 7 | 23 748.42 | 38.00 | 855.24 | 74.81 | 138.32 | 1068.37 |
| 8 | 24 104.77 | 38.04 | 867.43 | 75.93 | 138.47 | 1081.83 |
| 9 | 23 659.02 | 38.00 | 848.73 | 74.53 | 138.32 | 1061.58 |
| 10 | 22 502.81 | 38.28 | 810.69 | 70.88 | 139.34 | 1020.91 |
| 11 | 19 628.92 | 38.00 | 709.46 | 61.83 | 138.32 | 909.61 |
| 12 | 18 381.99 | 43.68 | 663.68 | 57.90 | 159.00 | 880.58 |
| $\Sigma$ | 261 840.99 | - | 9427.07 | 824.80 | 1766.44 | 12 018.31 |
| $\Sigma$ (orig.) | 260 055.72 | - | 9433.75 | 819.19 | 2561.83 | 12 814.77 |
| $\Delta$ | 1785.27 | - | $-6.68$ | 5.61 | $-795.39$ | $-796.46$ |

### Adaption Night Peaks

To further reduce costs, the algorithm is adapted so that the peak detection also works at night. If a peak occurs between 10 p.m. and 6 a.m., power above the peak limit is supplied from the battery. When the power demand during night hours is below the peak limit, the battery is charged to $SOC_{max}$. According to expectations, this should reduce the maximum power relevant for billing for each month to the specified peak limit.

The simulation shows that the best peak limit is 38$kW$ again and that the adjustment can save about € 100 more. The total savings in electricity costs now amount to € 902.80. The exact breakdown of the costs can be seen in Table 4.4. Owing to the algorithm's adaptation, the previously defined peak limit is no longer exceeded and used to calculate the demand costs for each individual month.

Table 4.4: Evaluation of electricity cost using adapted first simulation approach with a peak limit of 38 $kW$.

| Month | Energy (kWh) | max. Power (kW) | Energy Fee (€) | Loss Fee (€) | Demand Fee (€) | Total (€) |
|---|---|---|---|---|---|---|
| 1 | 18 209.12 | 38.0 | 659.38 | 57.36 | 138.32 | 855.06 |
| 2 | 20 016.08 | 38.0 | 720.50 | 63.05 | 138.32 | 921.87 |
| 3 | 20 534.83 | 38.0 | 740.69 | 64.68 | 138.32 | 943.69 |
| 4 | 23 366.05 | 38.0 | 837.90 | 73.60 | 138.32 | 1049.82 |
| 5 | 24 452.14 | 38.0 | 877.02 | 77.02 | 138.32 | 1092.36 |
| 6 | 23 245.11 | 38.0 | 836.61 | 73.22 | 138.32 | 1048.15 |
| 7 | 23 747.83 | 38.0 | 855.22 | 74.81 | 138.32 | 1068.35 |
| 8 | 24 104.76 | 38.0 | 867.43 | 75.93 | 138.32 | 1081.68 |
| 9 | 23 659.02 | 38.0 | 848.73 | 74.53 | 138.32 | 1061.58 |
| 10 | 22 502.74 | 38.0 | 810.69 | 70.88 | 138.32 | 1019.89 |
| 11 | 19 628.92 | 38.0 | 709.46 | 61.83 | 138.32 | 909.61 |
| 12 | 18 382.21 | 38.0 | 663.69 | 57.90 | 138.32 | 859.91 |
| $\Sigma$ | 261 848.81 | - | 9427.32 | 824.81 | 1659.84 | 11 911.97 |
| $\Sigma$ (orig.) | 260 055.72 | - | 9433.75 | 819.19 | 2561.83 | 12 814.77 |
| $\Delta$ | 1793.09 | - | $-6.43$ | 5.62 | $-901.99$ | $-902.80$ |

**Further Thoughts** To further reduce costs, the recurring patterns of the dataset must be exploited. The data analysis clearly shows that the average energy demand strongly depends on the following criteria (see Figure 3.3 on page 17 for example).

1. **Time of Day:** The energy demand is significantly higher during the day (between around 6 a.m. and 10 p.m.) than at night.

2. **Day of Week:** Much more energy is required on weekdays than on weekends or holidays.

3. **Season:** In Winter (December to February) the average energy demand is lower than in the other months.

The variable adjustment of the peak limit according to time of day does not bring any advantages in terms of peak shaving. Since the maximum of a month is used for the calculation of the relevant demand price, a change of the limit over individual days does not affect the total costs. The consideration of weekdays can be interesting for charging the battery and will be discussed in more detail in a later section.

Individual regulation of the peak limit according to the season of the year offers further savings potential. Significantly during the winter months, the limit can be lowered over an entire billing period, which is directly noticeable on the electricity bill. Therefore, a variable configuration of the peak limit is a possible extension. This can make it possible to lower the limit further in certain months.

### Season Limit

According to the findings of the above extension, a second, lower limit is now introduced. This so-called winter limit will be applied in the period from December to February and is intended to further reduce power demand costs in the months in which relatively less energy is required. In addition, it is tried to extend the winter limit in terms of timespan as much as possible.

The basic peak limit for the simulation is again assumed to be 38 $kW$. Different winter limits are evaluated. Figure 4.4 shows that the best winter limit is 34 $kW$. Additionally, Table 4.5 shows the exact results for all simulation runs. It is noticeable that with a winter limit of 34 $kW$, four empty battery errors occur despite the lowest total price. However, the winter limit of 36 $kW$ differs only slightly in the total price. This corresponds to an error time of one hour due to the 15-minute average interval.

Table 4.5: Results for different winter limits from December to February. 38 $kW$ is used as peak limit for the other months.

| $P_{lim,winter}$ (kW) | #Err01 (.) | #Err02 (.) | Energy Cost (€) | Loss Fee (€) | Demand Cost (€) | Total (€) |
|---|---|---|---|---|---|---|
| 26 | 1791 | 0 | 9431.46 | 826.12 | 1915.80 | 12 173.38 |
| 28 | 975 | 0 | 9429.81 | 826.29 | 1899.64 | 12 155.74 |
| 30 | 472 | 0 | 9428.05 | 826.24 | 1719.68 | 11 973.97 |
| 32 | 165 | 0 | 9426.91 | 826.01 | 1692.75 | 11 945.67 |
| 34 | 4 | 0 | 9426.82 | 825.64 | 1636.11 | 11 888.57 |
| 36 | 0 | 0 | 9427.00 | 825.15 | 1638.00 | 11 890.15 |
| 38 | 0 | 0 | 9427.32 | 824.81 | 1659.84 | 11 911.97 |

This result already looks promising. However, the winter limit can be extended to the adjacent months. The simulation shows that the winter limit of 34 $kW$ can be successfully applied from October to March, which reduces the power costs for three

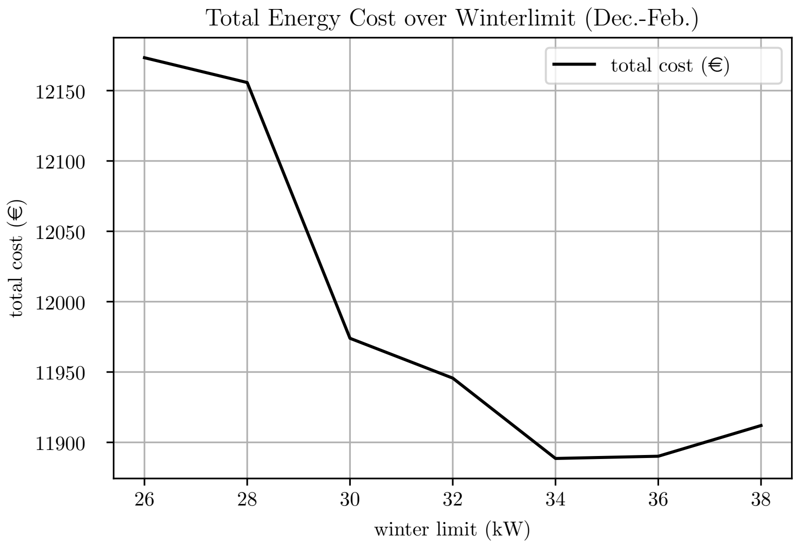



Figure 4.4: Total energy cost over winter limit from December to February using 38 $kW$ as limit for the other months.

more months. Figure 4.5 shows clearly this time that 34 $kW$ is the winter limit, which leads to the lowest total price.

Table 4.6: Results for different winter limits from October to March. 38 $kW$ is used as peak limit for the other months.

| $P_{lim,winter}$ (kW) | #Err01 (.) | #Err02 (.) | Energy Cost (€) | Loss Fee (€) | Demand Cost (€) | Total (€) |
|---|---|---|---|---|---|---|
| 26 | 4833 | 0 | 9437.57 | 826.55 | 2120.65 | 12 384.77 |
| 28 | 2611 | 0 | 9433.53 | 827.40 | 2080.91 | 12 341.84 |
| 30 | 1002 | 0 | 9429.17 | 827.89 | 1883.05 | 12 140.11 |
| 32 | 273 | 0 | 9426.22 | 827.44 | 1707.75 | 11 961.41 |
| 34 | 4 | 0 | 9425.97 | 826.59 | 1592.43 | 11 844.99 |
| 36 | 0 | 0 | 9426.52 | 825.58 | 1616.16 | 11 868.26 |
| 38 | 0 | 0 | 9427.32 | 824.81 | 1659.84 | 11 911.97 |

As Table 4.6 shows, the four occurring errors (`Error 01 - Battery empty`) remain. Table 4.7 shows that with the help of the winter limit over six months, a total cost saving of € 969.78 is possible. It can be seen that the battery appears to get empty in February because the specified peak limit of 34 $kW$ has been exceeded. A more detailed analysis shows that the highest peak here was reached on February 22nd at 17:00 (see Figure 4.6). At this time, the SOC of the battery was at one percent ($SOC_{min}$). Therefore this peak could not be supplied from the battery and is visible on the electricity bill. If the power demand had been higher at that time, the bill would have been more expensive. The battery could not be fully charged overnight. As one can see in Figure 4.7, which shows a whole week from Monday to Sunday,

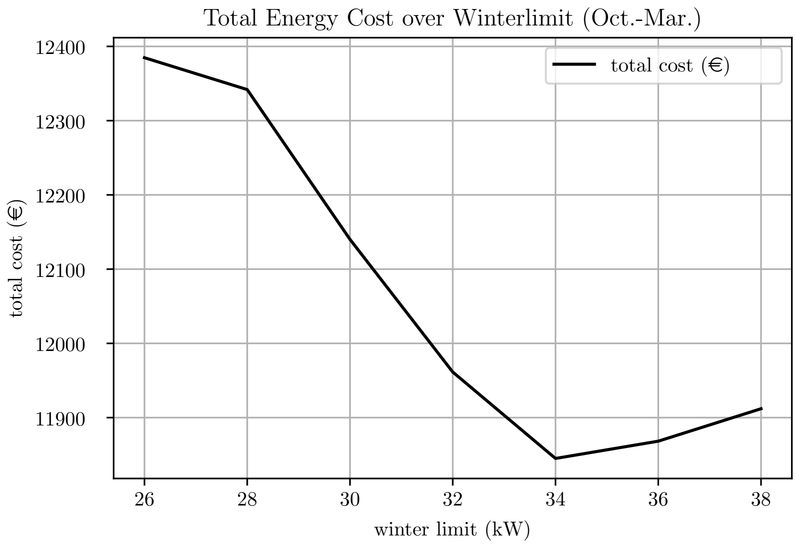



Figure 4.5: Total energy cost over winter limit from October to March using 38 $kW$ as limit for the other months.

the battery is discharged much more and thus could not be sufficiently charged during the night hours. Only at the weekend, when there is no high power demand for a more extended period of time, the charge level recovers.

What turns out during the simulation data analysis is that even during the day, there are time intervals with a lower power demand than the peak limit. These could be used in addition for charging the battery in order to prevent the battery from becoming empty during operation.

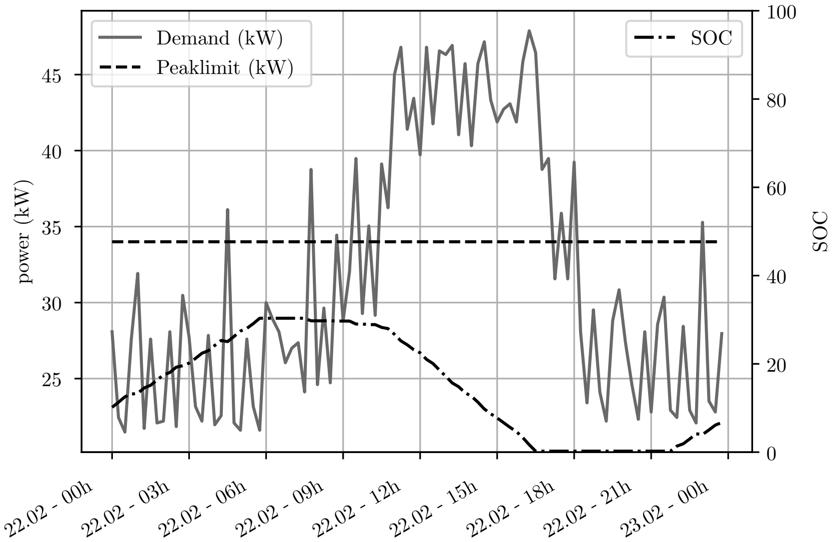



Figure 4.6: Detailed output of the peak shaving algorithm for $22^{nd}$ of February.

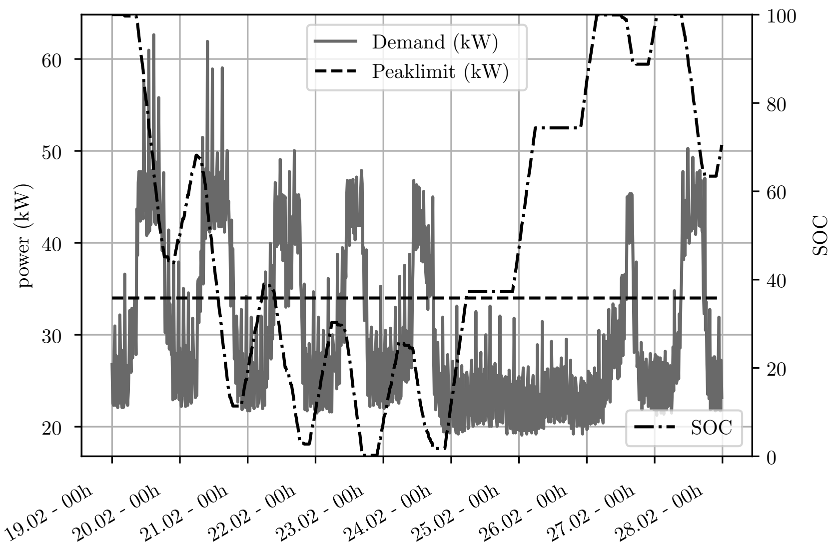



Figure 4.7: Detailed output of the peak shaving algorithm for $19^{th}$ to $28^{th}$ of February.

Table 4.7: Evaluation of electricity cost using adapted first simulation approach with a peak limit of 38 $kW$.

| Month | Energy (kWh) | max. Power (kW) | Energy Fee (€) | Loss Fee (€) | Demand Fee (€) | Total (€) |
|---|---|---|---|---|---|---|
| 1 | 18 234.85 | 34.00 | 657.69 | 57.44 | 123.76 | 838.89 |
| 2 | 20 129.57 | 39.48 | 719.46 | 63.41 | 143.71 | 926.58 |
| 3 | 20 682.54 | 34.00 | 742.74 | 65.15 | 123.76 | 931.65 |
| 4 | 23 364.38 | 38.00 | 837.85 | 73.60 | 138.32 | 1049.77 |
| 5 | 24 452.34 | 38.00 | 877.03 | 77.02 | 138.32 | 1092.37 |
| 6 | 23 245.49 | 38.00 | 836.62 | 73.22 | 138.32 | 1048.16 |
| 7 | 23 748.42 | 38.00 | 855.24 | 74.81 | 138.32 | 1068.37 |
| 8 | 24 104.44 | 38.00 | 867.42 | 75.93 | 138.32 | 1081.67 |
| 9 | 23 659.02 | 38.00 | 848.73 | 74.53 | 138.32 | 1061.58 |
| 10 | 22 583.14 | 34.00 | 808.60 | 71.14 | 123.76 | 1003.50 |
| 11 | 19 768.61 | 34.00 | 710.82 | 62.27 | 123.76 | 896.85 |
| 12 | 18 435.74 | 34.00 | 663.77 | 58.07 | 123.76 | 845.60 |
| $\Sigma$ | 262 408.54 | - | 9425.97 | 826.59 | 1592.43 | 11 844.99 |
| $\Sigma$ (orig.) | 260 055.72 | - | 9433.75 | 819.19 | 2561.83 | 12 814.77 |
| $\Delta$ | 2352.82 | - | $-7.78$ | 7.40 | $-969.40$ | $-969.78$ |

### Day Charging

The algorithm is now adapted so that charging is also possible during the day. The flow chart in Figure 4.9 shows the process workflow. If the power demand lies below the peak limit, the difference between these two (see equation 4.2) is used for charging the battery up to a maximum SOC defined by `SOC_day_charge_limit_max`, provided that the SOC has fallen below a previously defined threshold value (`SOC_day_charge_limit_min`).

$$P_{charge} = P_{lim} - P_{dem} \tag{4.2}$$

During the night hours the charging to $SOC_{max}$ is done as the energy price is lower than during the day. This behavior is implemented as a state machine. The two contained states `IDLE` and `CHARGE` and their relations are illustrated in Figure 4.8. The state machine is executed only during the daytime and only under the condition that no peak is present.

The simulation results show that the peak limits (base and winter limit) of 38 $kW$ and 34 $kW$ respectively, which already resulted in the previous simulations, still lead to the lowest overall price. However, the battery is used better now, which has the consequence that even in February a peak limit of 34 $kW$ works without any overruns. A total saving of € 987.85 is reached, as Table 4.8 makes evident.

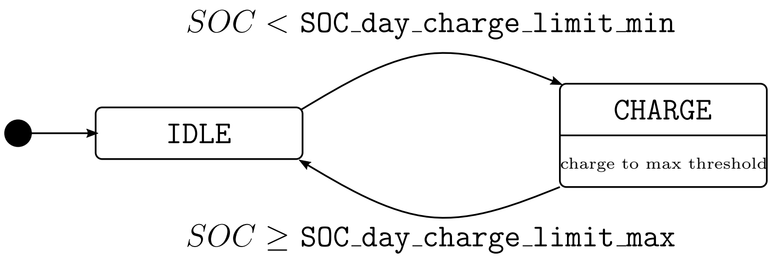



Figure 4.8: State diagram for day charging.

Table 4.8: Evaluation of electricity cost using day charging with a peak limit of 38 $kW$ and winter limit of 34 $kW$.

| Month | Energy (kWh) | max. Power (kW) | Energy Fee (€) | Loss Fee (€) | Demand Fee (€) | Total (€) |
|---|---|---|---|---|---|---|
| 1 | 18 234.85 | 34.0 | 657.69 | 57.44 | 123.76 | 838.89 |
| 2 | 20 130.60 | 34.0 | 719.86 | 63.41 | 123.76 | 907.03 |
| 3 | 20 683.41 | 34.0 | 742.77 | 65.15 | 123.76 | 931.68 |
| 4 | 23 365.72 | 38.0 | 838.15 | 73.60 | 138.32 | 1050.07 |
| 5 | 24 513.75 | 38.0 | 879.55 | 77.22 | 138.32 | 1095.09 |
| 6 | 23 182.67 | 38.0 | 834.68 | 73.03 | 138.32 | 1046.03 |
| 7 | 23 748.42 | 38.0 | 855.24 | 74.81 | 138.32 | 1068.37 |
| 8 | 24 104.41 | 38.0 | 867.42 | 75.93 | 138.32 | 1081.67 |
| 9 | 23 659.29 | 38.0 | 848.74 | 74.53 | 138.32 | 1061.59 |
| 10 | 22 585.27 | 34.0 | 809.18 | 71.14 | 123.76 | 1004.08 |
| 11 | 19 768.00 | 34.0 | 710.80 | 62.27 | 123.76 | 896.83 |
| 12 | 18 435.40 | 34.0 | 663.76 | 58.07 | 123.76 | 845.59 |
| $\Sigma$ | 262 411.79 | - | 9427.84 | 826.60 | 1572.48 | 11 826.92 |
| $\Sigma$ (orig.) | 260 055.72 | - | 9433.75 | 819.19 | 2561.83 | 12 814.77 |
| $\Delta$ | 2356.07 | - | −5.91 | 7.41 | −989.35 | −987.85 |

### 4.2.2 Adaptive Algorithms

While static algorithms do not change the peak limit after it has been set, adaptive algorithms dynamically adapt the peak limit to the current conditions. As a result, the peak limit is changed several times per month and adapts to the current power demand. Basically, two approaches can be distinguished: Analytical and responsive methods. Both are explained in this section. Besides, a combined algorithm is presented at the end, which unites the advantages of both approaches.

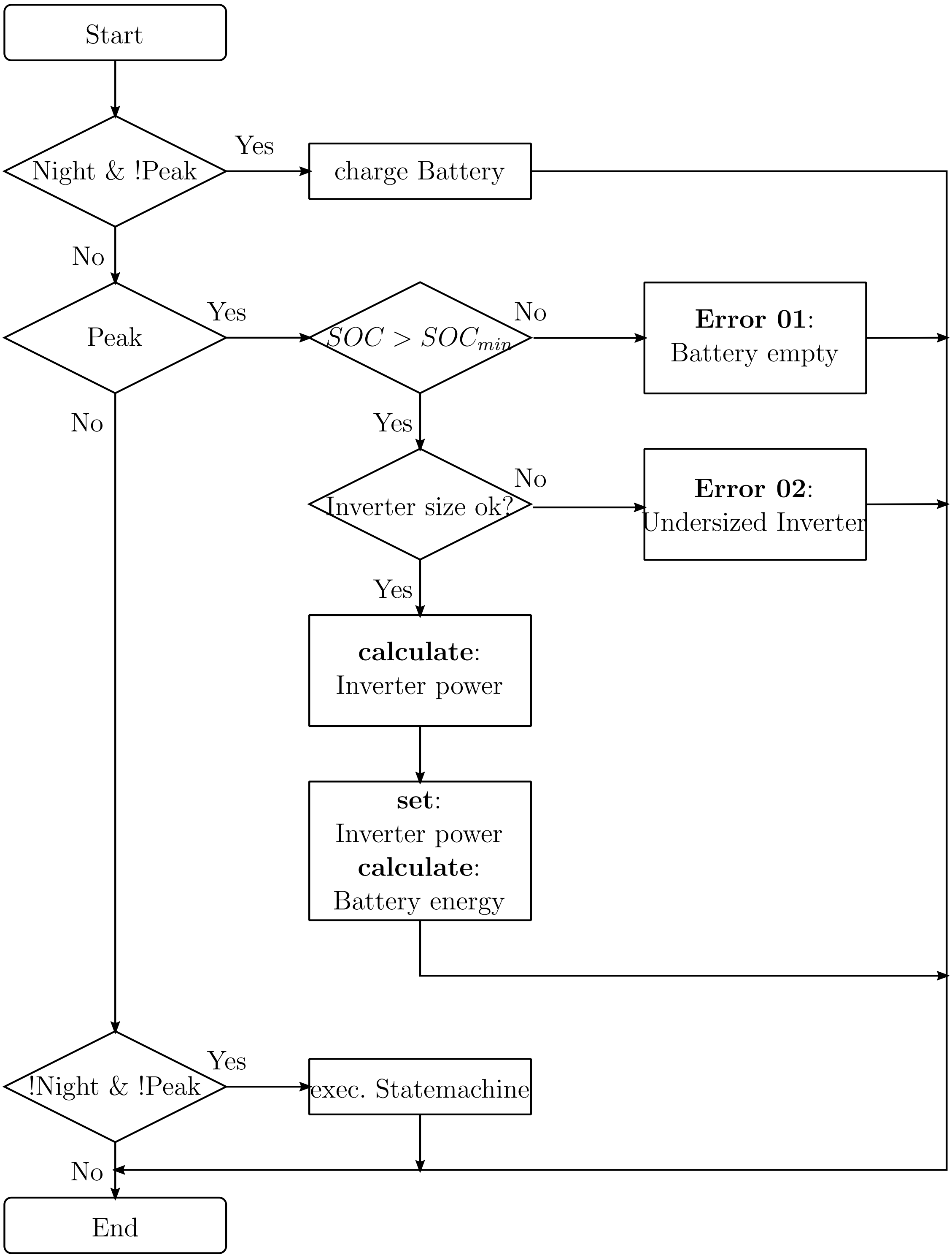



Figure 4.9: Flowchart day charging.

### Analytic Approach (v1)

The analytical method's goal is to calculate a peak limit from the current power demand and from the data recorded in the past time, which is valid for the future. Of course, this assumes that the general power demand does not change or only very slightly compared to the data used for the calculation.

The energy price calculation states that the day from 6 to 22 o'clock with 16 hours is twice as long as the night, which lasts eight hours, from 10 to 6 o'clock accordingly. The calculation of the peak limit is based on the following two fundamental premises. On the one hand, the battery should be charged as fully as possible during the night hours; on the other hand, it should be used as efficiently as possible during the day. The trade-off lies in the peak limit. The faster the battery should be charged, the higher the peak limit has to be. Nevertheless, this also results in a higher power price. A higher peak limit also means that less energy is drawn from the battery during the day, which in turn does not fully utilize the battery.

The battery has a nominal storage capacity of $E_B^{nom}$ in kWh and is operated between $SOC_{min}$ and $SOC_{max}$. This allows the actual usable energy to be calculated with

$$E_B^{use} = E_B^{nom} \cdot (SOC_{max} - SOC_{min}). \tag{4.3}$$

**Fully charge the battery at night** During the night, the following applies to each time interval $\Delta t$: The energy difference required to charge the battery to $SOC_{max}$ is

$$\Delta E = E_B^{use} - E_B \tag{4.4}$$

where $E_B$ is the energy currently stored in the battery. Per time unit $\Delta t = 1/4\, h$, the energy

$$E_{\Delta t} = \frac{\Delta E}{t_{night,remaining}/\Delta t} = \frac{\Delta E \cdot \Delta t}{t_{night,remaining}} \tag{4.5}$$

must therefore be recharged (Equation 4.5). $t_{night,remaining}$ denotes the remaining time in night mode and is calculated based on the current time $t$ as

$$t_{night,remaining} = (6 - t)\ mod\ 24. \tag{4.6}$$

$E_{\Delta t}$ can then be used to calculate

$$P_{\Delta t} = \frac{E_{\Delta t}}{\Delta t} = \frac{\Delta E}{t_{night,remaining}}, \tag{4.7}$$

which is the power that is required to charge the battery.

This results in a theoretical peak limit

$$P_{lim,th} = P_{demand,\Delta t} + P_{\Delta t}. \tag{4.8}$$

Since peaks also occur at night, calculating the theoretical peak limit according to equation 4.8 would produce even larger peaks. This has to be avoided. Therefore the theoretical peak limit is calculated as

$$P_{lim,th} = \overline{P}_{night,n} + P_{\Delta t}, \tag{4.9}$$

based on an average of the power demand of the last $n$ nights ($\overline{P}_{night,n}$).

**Utilize Battery during day time** The day mode is valid for 16 hours. Based on the current time $t$, the remaining time until the beginning of the night can be calculated with

$$t_{day,remaining} = (22 - t)\ mod\ 24. \tag{4.10}$$

For the maximum possible energy draw $E_{B,\Delta t}$ from the battery per time unit $\Delta t$, the following applies:

$$E_{B,\Delta t} = \frac{E_B^{nom}}{t_{day,remaining}/\Delta t} = \frac{E_B^{nom} \cdot \Delta t}{t_{day,remaining}}. \tag{4.11}$$

This results in the possible power withdrawal

$$P_{\Delta t} = \frac{E_{\Delta t}}{\Delta t} = \frac{\Delta E}{t_{day,remaining}}. \tag{4.12}$$

The theoretic peak limit for the current time interval $\Delta t$ is

$$P_{lim,th} = P_{demand,\Delta t} - P_{\Delta t}. \tag{4.13}$$

The actual peak limit is the maximum of the theoretical limits calculated in the current month. Of course, the theoretical limits of the night hours are also taken into account. To avoid immediately increasing the peak limit for a short peak, even if this would not be necessary at all, it is only adjusted if the remaining time until the next mode change (day/night) in percent is smaller than the current battery charge level $SOC$.

Table 4.9 clearly shows that the analytical method's annual savings do not reach those of the static algorithms. However, it is interesting to note that some months also get by with a significantly lower peak limit than previously assumed with the static methods. In general, it can be said that the algorithm is already working quite well but reacts too slowly to increasing power demands. This means that the peak limit is only increased when the battery charge level is already shallow. Therefore, it cannot be avoided that the battery gets empty, which results in the high peaks. This needs to be improved.

Table 4.9: Evaluation of electricity cost using the analytic approach (v1).

| Month | Energy (kWh) | max. Power (kW) | Energy Fee (€) | Loss Fee (€) | Demand Fee (€) | Total (€) |
|---|---|---|---|---|---|---|
| 1 | 18 355.80 | 30.36 | 663.22 | 57.82 | 110.51 | 831.55 |
| 2 | 20 250.85 | 34.30 | 729.56 | 63.79 | 124.84 | 918.19 |
| 3 | 20 773.56 | 41.28 | 747.15 | 65.44 | 150.26 | 962.85 |
| 4 | 23 442.58 | 53.04 | 844.94 | 73.84 | 193.07 | 1111.85 |
| 5 | 24 411.47 | 42.19 | 883.47 | 76.90 | 153.58 | 1113.95 |
| 6 | 23 241.79 | 60.84 | 838.56 | 73.21 | 221.46 | 1133.23 |
| 7 | 23 818.45 | 59.88 | 859.29 | 75.03 | 217.96 | 1152.28 |
| 8 | 24 073.25 | 56.40 | 866.64 | 75.83 | 205.30 | 1147.77 |
| 9 | 23 591.94 | 64.71 | 854.38 | 74.31 | 235.56 | 1164.25 |
| 10 | 22 668.42 | 48.96 | 815.04 | 71.41 | 178.21 | 1064.66 |
| 11 | 19 684.08 | 42.12 | 713.82 | 62.00 | 153.33 | 929.15 |
| 12 | 18 549.21 | 30.00 | 668.28 | 58.43 | 109.20 | 835.91 |
| $\Sigma$ | 262 861.40 | - | 9484.35 | 828.01 | 2053.28 | 12 365.64 |
| $\Sigma$ (orig.) | 260 055.72 | - | 9433.75 | 819.19 | 2561.83 | 12 814.77 |
| $\Delta$ | 2805.68 | - | 50.60 | 8.82 | $-508.55$ | $-449.13$ |

### Analytic Approach (v2)

To enable the algorithm to react faster to approaching peaks and increase the limit earlier and therefore less high, the threshold value used is adapted. A linear function is introduced that maps the current time at night to a threshold charge level of $SOC_{min}$ to 75%. As soon as the current charge level falls below this threshold value, the peak limit is set to the theoretical peak limit calculated from version 1 if it is higher than the previous monthly maximum. With this extension the battery is charged to at least 75% every morning, so there should be no problems during the day. An additional extension considers holidays which have to be provided as a list before. Thus it is possible not to increase the limit on holidays, because here, as on weekends, only very few peaks are to be expected and the battery can also be charged during the day.

The cost-saving precipitates here, as Table 4.10 shows, already twice as high as before. Compared with the results from Table 4.9 the highest peaks in all months are in a moderate range, which lies only slightly over the statically fixed limits. In some months, the static limit is even undercut.

Overall, it can be said that the algorithm approximates the optimal limits quite well but still reacts too slowly to an increasing power graph. If the limit is then increased, it is immediately raised by a relatively large amount. This could be improved in a next version.

Table 4.10: Evaluation of electricity cost using the analytic approach (v2).

| Month | Energy (kWh) | max. Power (kW) | Energy Fee (€) | Loss Fee (€) | Demand Fee (€) | Total (€) |
|---|---|---|---|---|---|---|
| 1 | 18 319.84 | 31.83 | 663.48 | 57.71 | 115.86 | 837.05 |
| 2 | 20 217.45 | 36.89 | 730.22 | 63.68 | 134.30 | 928.20 |
| 3 | 20 710.88 | 37.23 | 745.46 | 65.24 | 135.50 | 946.20 |
| 4 | 23 440.46 | 38.19 | 845.08 | 73.84 | 139.00 | 1057.92 |
| 5 | 24 357.12 | 47.38 | 882.96 | 76.72 | 172.48 | 1132.16 |
| 6 | 23 219.71 | 38.13 | 838.31 | 73.14 | 138.80 | 1050.25 |
| 7 | 23 772.84 | 40.70 | 861.18 | 74.88 | 148.18 | 1084.24 |
| 8 | 24 124.55 | 40.10 | 871.65 | 75.99 | 145.99 | 1093.63 |
| 9 | 23 539.43 | 40.96 | 851.24 | 74.15 | 149.08 | 1074.47 |
| 10 | 22 560.60 | 41.22 | 816.18 | 71.07 | 150.06 | 1037.31 |
| 11 | 19 778.28 | 34.86 | 714.10 | 62.30 | 126.90 | 903.30 |
| 12 | 18 547.80 | 30.00 | 668.23 | 58.43 | 109.20 | 835.86 |
| $\Sigma$ | 262 588.96 | - | 9488.09 | 827.15 | 1665.35 | 11 980.59 |
| $\Sigma$ (orig.) | 260 055.72 | - | 9433.75 | 819.19 | 2561.83 | 12 814.77 |
| $\Delta$ | 2533.24 | - | 54.34 | 7.96 | $-896.48$ | $-834.18$ |

The most prominent is the limit in May, which is significantly above the expected limit. A close analysis shows that the sharp increase in the limit at the beginning of May could possibly have been avoided if the limit had been set a little higher the first time it fell below the threshold. In Figure 4.10 the two relevant points where SOC undercuts the limiter function are marked.

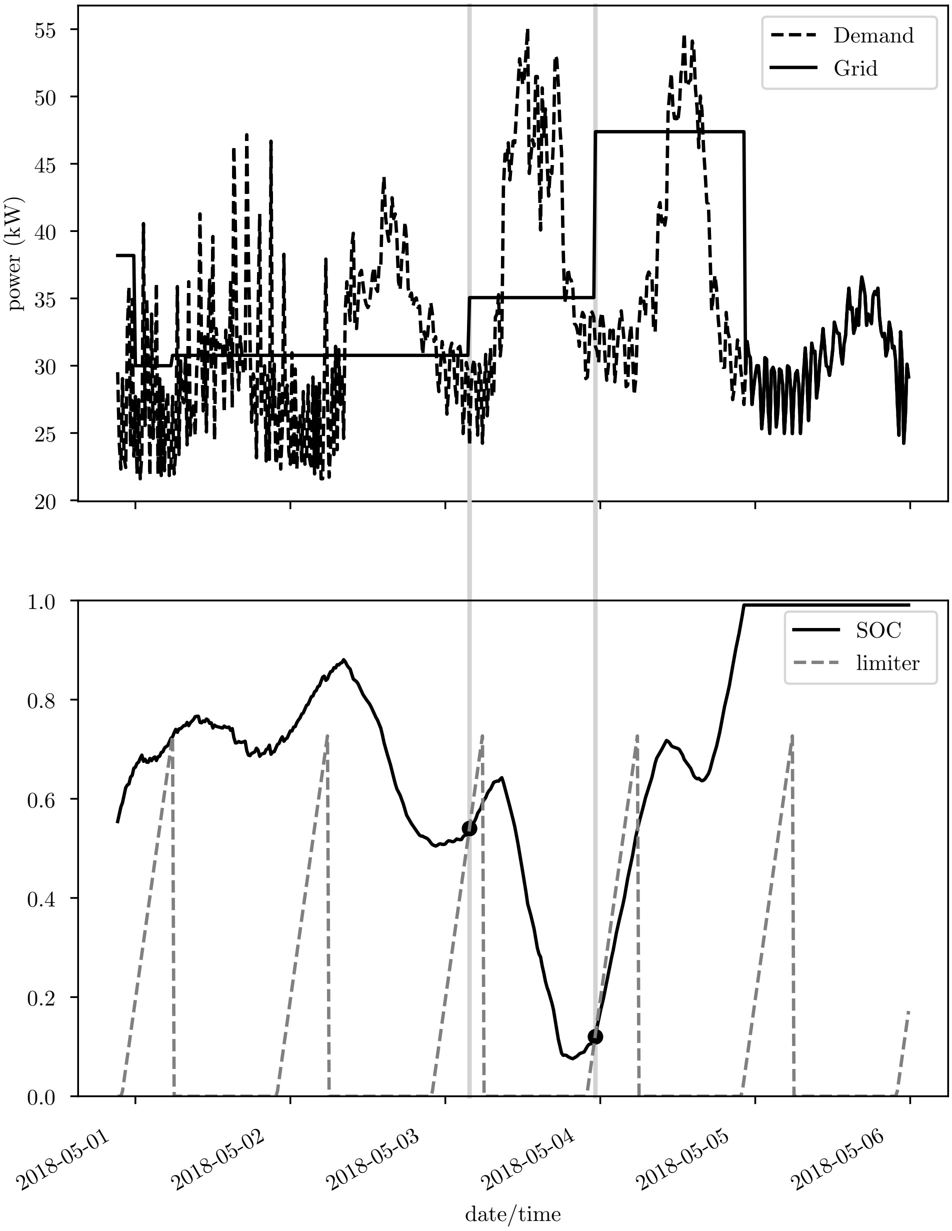



Figure 4.10: Simulation details for the beginning of May indicating the points where SOC undercuts the limiter function and therefore the limit is raised.

### Analytic Approach (v3)

To avoid that the peak limit depends on the current demand so heavily but can still react to the performance profile, a moving average is used in this approach. Therefore several days or nights are considered, and the power demand is averaged. Only values of working days are used. Weekends and holidays are excluded from averaging in order not to falsify the results. These two average values, one for the day and one for the night, are used for the peak limit calculations. The size of the sliding window for averaging can be specified by the number of days or nights to be considered and can be defined by two parameters `number_of_nights_for_mean` and `number_of_days_for_mean`. Two threshold values are set for the limiting function, which is a linear function and restricts the battery charge level ($SOC$).

$$SOC_{day\,start} = 70\% \tag{4.14}$$

$$SOC_{day\,end} = 20\% \tag{4.15}$$

These parameters are defined in equations 4.14 and 4.15 so that at the beginning of each day the SOC must be at least 70% and at the end of each day the SOC must be above 20% to keep the current peak limit active. Otherwise, the limit is increased. The limiter function used for this is a linear function as defined in Equation 4.16 and 4.17 with $t$ as the current time and $nightlen$ as the length of the night in hours (8).

$$x = \left( t.hour + \frac{t.minute}{60} - 22 \right) \; mod\, 24 \tag{4.16}$$

$$limit(x) = \frac{(SOC_{day\,start} - SOC_{day\,end})}{nightlen} \cdot x + SOC_{day\,end} \tag{4.17}$$

During night hours, similar to the last version of the algorithm, the energy difference $\Delta E$ that is needed to charge the battery to the defined minimum at the start of the day is calculated using Equation 4.18.

$$\Delta E = E_B^{nom} \cdot SOC_{day\,start} - E_B \tag{4.18}$$

This is then turned into a power value valid for each time step of $\Delta t = 1/4\, h$.

$$P_{\Delta t} = \frac{\Delta E}{nightlen} \tag{4.19}$$

Where $nightlen$ denotes the length of a night in hours (8).

With the help of the moving average $\overline{P}_{night}$ calculated for the last `number_of_nights_for_mean` nights (excluding weekends and holidays), a theoretic peak limit is determined.

$$P_{lim,th} = \overline{P}_{night} + S \cdot P_{\Delta t} \tag{4.20}$$

$S$ is defined as a security factor $\geq 1$. This limit is then applied if it is above the current limit, the $SOC$ lies under the limiter function, and the current date is neither a holiday nor a Saturday. Otherwise, it is just ignored.

During the day, the battery usage factor $\beta$ is calculated from an initially defined restriction factor $\alpha$, with which a grid usage factor $\gamma$ is then defined in Equation 4.23.

$$\alpha = 70\% \tag{4.21}$$

$$\beta = \frac{E_B \cdot \alpha}{\overline{P}_{day} \cdot daylen} \tag{4.22}$$

With $\overline{P}_{day}$ as the moving average for the day values and $daylen$ as the length of a day in hours (16).

$$\gamma = 1 - \beta \tag{4.23}$$

The theoretical peak limit $P_{lim,th}$ then depends on the calculated grid usage factor $\gamma$ and the average day power demand $\overline{P}_{day}$ as stated in Equation 4.24.

$$P_{lim,th} = \overline{P}_{day} \cdot \gamma \tag{4.24}$$

Between 6 a.m. and 10 p.m. the limiter function is kept at minimum to provide a free usage of the battery.

$$limiter(x) = SOC_{day\ end} \tag{4.25}$$

Several simulations with different parameter constellations have shown that the highest total savings are achieved with the following values:

`number_of_nights_for_mean` $= 1$
`number_of_days_for_mean` $= 3$
$\alpha = 0.7$
$S = 1.0$
$SOC_{day\ start} = 70\%$
$SOC_{day\ end} = 20\%$

These values lead to a simulation result, which is broken down in Table 4.11.

By averaging the power values, a more stable course of the peak limit is achieved, but there are scenarios in which the algorithm has problems to adjust the limit in time. If the power demand increases steadily for several days in a row, the battery will reach its limits. As an example, the simulation details for the beginning of July are shown in Figure 4.11. Here, the average power demand increases significantly over five consecutive days. The peak limit is not increased until July 5, which causes the battery to become almost empty. The grey curve shows the calculated theoretic peak limit ($P_{lim,th}$) which only rises slowly due to the averaging. July 1st is a Sunday, so the values are not taken into account in the average calculation and

Table 4.11: Evaluation of electricity cost using the analytic approach (v3).

| Month | Energy (kWh) | max. Power (kW) | Energy Fee (€) | Loss Fee (€) | Demand Fee (€) | Total (€) |
|---|---|---|---|---|---|---|
| 1 | 18 293.61 | 33.68 | 664.27 | 57.62 | 122.61 | 844.50 |
| 2 | 20 182.41 | 38.73 | 728.51 | 63.57 | 140.98 | 933.06 |
| 3 | 20 562.90 | 35.45 | 743.61 | 64.77 | 129.04 | 937.42 |
| 4 | 23 424.47 | 39.97 | 846.22 | 73.79 | 145.50 | 1065.51 |
| 5 | 24 386.49 | 41.84 | 882.50 | 76.82 | 152.28 | 1111.60 |
| 6 | 23 155.10 | 42.51 | 837.00 | 72.94 | 154.72 | 1064.66 |
| 7 | 23 714.74 | 43.31 | 860.05 | 74.70 | 157.63 | 1092.38 |
| 8 | 24 147.19 | 38.36 | 872.06 | 76.06 | 139.63 | 1087.75 |
| 9 | 23 619.29 | 37.89 | 850.74 | 74.40 | 137.90 | 1063.04 |
| 10 | 22 555.05 | 39.64 | 816.94 | 71.05 | 144.31 | 1032.30 |
| 11 | 19 681.10 | 35.24 | 713.44 | 62.00 | 128.28 | 903.72 |
| 12 | 18 505.47 | 31.92 | 667.87 | 58.29 | 116.20 | 842.36 |
| $\Sigma$ | 262 227.81 | - | 9483.21 | 826.01 | 1669.08 | 11 978.3 |
| $\Sigma$ (orig.) | 260 055.72 | - | 9433.75 | 819.19 | 2561.83 | 12 814.77 |
| $\Delta$ | 2172.09 | - | 49.46 | 6.82 | $-892.75$ | $-836.47$ |

the theoretical limit does not change. Since the power demand is lower again on July 6, the charge level can recover. If this hadn't happened and the power demand had increased again, the battery would have become empty and the whole power would have been drawn from the grid.

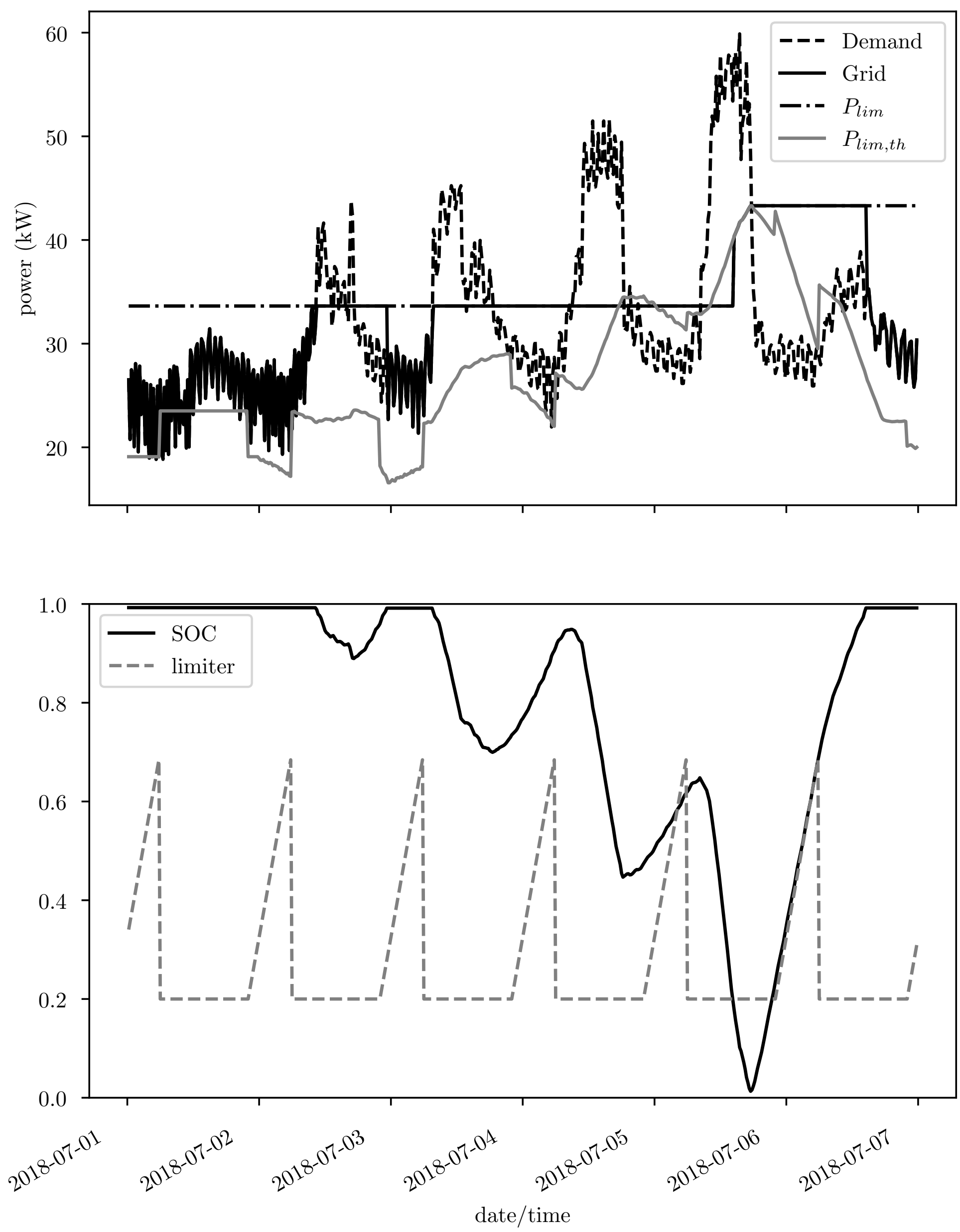



Figure 4.11: Simulation details for the beginning of July.

#### Responsive Approach

A purely responsive approach, which only determines a peak limit based on the current SOC values, is not appropriate and leads to immensely high peak limits in the individual accounting periods. However, various simulations have shown that responsive methods have the potential to support analytical approaches. One method that is promising for this is SOC monitoring. Here the battery charge level is monitored, and the future course is approximated. The predicted curve can be used to determine the time at which the battery will be empty. This information can then be used to adjust the peak limit if necessary. The approximation is made as a linear function because the simulation results show that the battery's discharge curve is approximately a straight line.

For this purpose, the SOC values are recorded at regular 15-minute intervals and stored and analysed over a longer period of, for example, one hour.

$$\underbrace{[SOC_{t_{n-m}},\ \ldots,\ SOC_{t_{n-1}},\ SOC_{t_n}]}_{\text{in } \Delta t \text{ time}}$$

For linearization, the first and last values of the time series are of particular interest. The difference between those two values represents the absolute change of the charge level of the battery.

$$\Delta SOC = SOC_{t_n} - SOC_{t_{n-m}} \tag{4.26}$$

If $\Delta SOC$ is negative, a further calculation for the desired application makes sense and it can be used to model the trend as a linear function

$$SOC(t) = \frac{\Delta SOC}{\Delta t} \cdot (t - t_n) + SOC_{t_n}. \tag{4.27}$$

To calculate the time of emptying, Equation 4.27 is set to zero and transformed to $t$:

$$0 = \frac{\Delta SOC}{\Delta t} \cdot (t - t_n) + SOC_{t_n} \tag{4.28}$$

$$\Rightarrow t = t_n - \frac{SOC_{t_n} \cdot \Delta t}{\Delta SOC} \tag{4.29}$$

### Combined Algorithm

The combined algorithm is based on analytical calculations with past profile data. For better stability and an application on changing load profiles, a responsive component is additionally integrated. The SOC monitoring adjusts the peak limit if the analytical path would react too slowly, thus preventing the battery from running empty, which would lead to high costs.

The algorithm's analytical part runs in cycles from 10 p.m. (beginning of the night) to 9:45 p.m. (end of the day). At the beginning of each cycle, an optimal peak limit can be calculated for the past cycle based on the demand data. This limit allows the same amount of energy to be charged into the battery during the night (or off-peak hours) consumed during the day (or peak hours). For this purpose, all the power values which occurred are summed up

$$P_{sum} = \sum_{t}^{t \in [22:00;22:00)} P_t, \tag{4.30}$$

the energy is calculated therefrom, and a power value

$$P_{lim} = \frac{P_{sum} \cdot \Delta t}{24\ h} \tag{4.31}$$

is determined from the total energy, at which a day-night-balance exists.

Since this limit is optimal for the past cycle, this does not necessarily mean that it will also keep the balance for the coming cycle. Therefore additional parameters have to be included. If at the end of a cycle the battery charge level is less than 50%, which means that significantly more energy has been drawn than has been recharged, the peak limit is increased by a charge amount

$$P_{load} = \frac{\Delta E}{24\ h}, \tag{4.32}$$

where $\Delta E$ denotes the energy difference calculated as

$$\Delta E = \frac{E_B^{nom} \cdot (SOC_{max} - SOC)}{\texttt{days\_to\_load}}. \tag{4.33}$$

For this purpose, a parameter called `days_to_load` is introduced, that determines after how many days, on which the day-night-balance is assumed, the missing energy should be recharged. The amount added to the limit is calculated with

$$P_{lim} = P_{lim} + P_{load}. \tag{4.34}$$

This recalculated limit is applied if it is greater than the previous peak limit and the SOC is less than 50% or the sum of the power values is more than 10% higher than in the previous n days considered.

To support the analytical part, two responsive components are involved. On the one hand, the optimal peak limit for the current cycle, which can only be calculated precisely at its end, is approximated in order to be able to react earlier to increasing power demand, if required. This process is called "Attention Mode". On the other hand, the "SOC monitoring" ensures that the battery does not become empty by estimating the future course of the charge level.

**Attention Mode** In each time interval of 15 minutes, the sum of all power values that occurred in the current cycle is available. Based on this, an approximate limit can be calculated using the current time

$$P_{lim,\,app} = \frac{P_{sum} \cdot \Delta t}{t_{cycle}}, \tag{4.35}$$

which comes as close as possible to the optimal limit at the end of the cycle using $t$ as the current time and

$$t_{cycle} = \left( t.hour + \frac{t.minute}{60} - 22 \right) \; mod \; 24 \tag{4.36}$$

as the elapsed time in the current cycle.

If this calculated limit is greater than the current peak limit, it will be added to a list. This list is filled as long as the predicted limits lie above the current one and is reset when the calculated limit lies below it. Attention mode is activated under the condition of a higher calculated limit if the SOC drops below 50% or the estimated limit is significantly higher than the current limit. If the Attention Mode is active, the approximated limits stored in the list are averaged. If the Attention Mode has been running for more than one hour (that corresponds to 4 values), the average is calculated. This mean value is then used as the current peak limit. In extreme cases, if the SOC falls below 20%, the averaged limit is used earlier. The Attention Mode is terminated if the calculated limit drops below the current limit or the power demand decreases below the current limit, which means there is currently no peak.

**SOC Monitoring** As described above, the SOC monitoring reacts to a falling trend line of the battery charge level. It calculates a possible peak limit, which prevents the battery from running empty before the night starts. During the night hours, the battery can be recharged. The detailed calculations are explained under 'Responsive Approach' on page 46.

The flow chart in Figure 4.12 shows the workflow of the algorithm. At the beginning of each month, the peak limit is reset. The total sums of the last n working days are considered, and the highest optimal limit of these days is used as the base value for the coming month. During the course of a month, the peak limit can only rise and

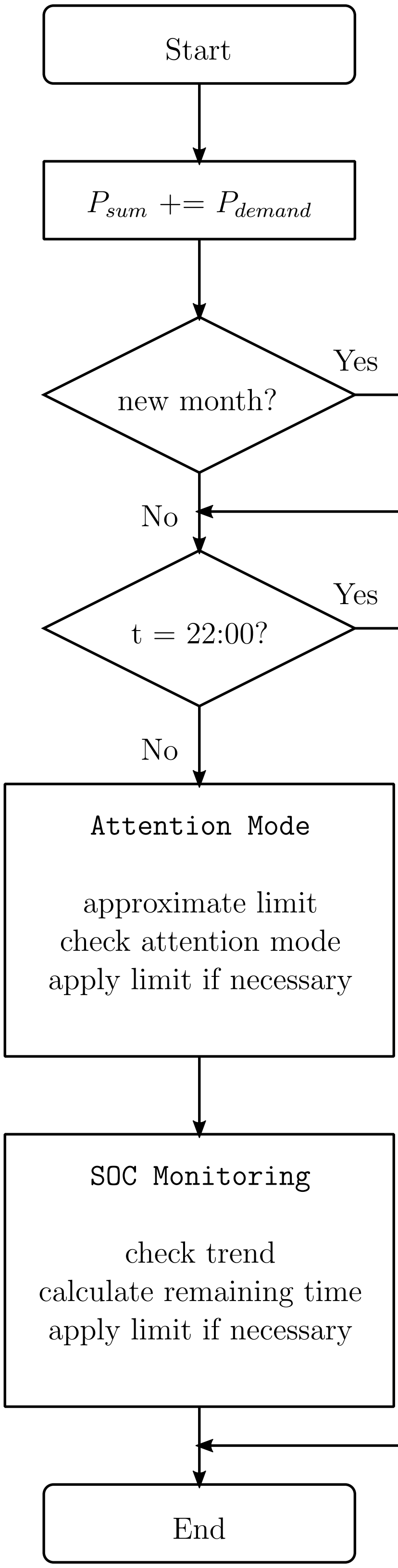



Figure 4.12: Flowchart for combined analytic and responsive algorithm.

not fall, so it is essential to reset the limit to a minimum value at the beginning of each month. At 10 p.m., when one cycle ends and the next begins, all necessary calculations are accomplished. The optimum peak limit of the previous cycle can now be determined and, if it is higher than the previous limit, will be applied. Also, all required statistical evaluations are determined and stored for further use. For all other times, the algorithm is in the middle of a cycle and applies the responsive methods. First, the Attention Mode is evaluated, then the SOC Monitoring follows. Both are described in detail above. After running the simulation, the following result is obtained.

Table 4.12: Evaluation of electricity cost using combined approach.

| Month | Energy (kWh) | max. Power (kW) | Energy Fee (€) | Loss Fee (€) | Demand Fee (€) | Total (€) |
|---|---|---|---|---|---|---|
| 1 | 18 373.14 | 30.00 | 663.82 | 57.88 | 109.20 | 830.90 |
| 2 | 20 272.54 | 36.65 | 729.73 | 63.86 | 133.41 | 927.00 |
| 3 | 20 630.30 | 32.20 | 743.33 | 64.99 | 117.21 | 925.53 |
| 4 | 23 487.72 | 41.09 | 846.55 | 73.99 | 149.57 | 1070.11 |
| 5 | 24 404.04 | 40.06 | 881.12 | 76.87 | 145.82 | 1103.81 |
| 6 | 23 071.98 | 40.93 | 835.04 | 72.68 | 148.99 | 1056.71 |
| 7 | 23 740.78 | 43.94 | 860.64 | 74.78 | 159.94 | 1095.36 |
| 8 | 24 218.48 | 37.91 | 872.51 | 76.29 | 137.99 | 1086.79 |
| 9 | 23 615.28 | 37.33 | 850.62 | 74.39 | 135.88 | 1060.89 |
| 10 | 22 639.78 | 37.06 | 817.40 | 71.32 | 134.90 | 1023.62 |
| 11 | 19 832.67 | 30.96 | 714.00 | 62.47 | 112.69 | 889.16 |
| 12 | 18 502.66 | 32.01 | 668.16 | 58.28 | 116.52 | 842.96 |
| $\Sigma$ | 262 789.37 | - | 9482.92 | 827.79 | 1602.12 | 11 912.83 |
| $\Sigma$ (orig.) | 260 055.72 | - | 9433.75 | 819.19 | 2561.83 | 12 814.77 |
| $\Delta$ | 2733.65 | - | 49.17 | 8.60 | $-959.71$ | $-901.94$ |

Table 4.12 shows that the highest peak limit in almost all months is below the statically calculated optimum. In some cases, even significantly below. That leads to a saving of € 901.94 in relation to the original data. Above all, the fact that the limit is determined "live", which means only based on the past demand curve, shows that peak shaving is absolutely possible with this algorithm without having analyzed large amounts of data beforehand.

## 4.3 Integration in EOS Energy Manager

This section describes the hardware and software behind the EOS Energy Manager (EEM). It also explains how the peak shaving algorithm can be integrated into the existing system as a so-called bundle so that it runs in parallel with the other applications on the system.

### 4.3.1 The Embedded System

The EOS Energy Manager (EEM) is an embedded system based on an ARM Cortex-A8. The surrounding board offers, as Figure 4.13 shows, 16 GPIOs, 4 of them supporting PWM, 6 Modbus interfaces, and an Ethernet connection. Additionally, it provides an SD-Card interface, which is the system memory. Thus the board is ideally equipped for applications in the energy sector. The power values required for peak shaving are recorded by a smart meter and made available via a Modbus interface. The system runs Debian Linux as the operating system and a software called `macchina.io` on top.

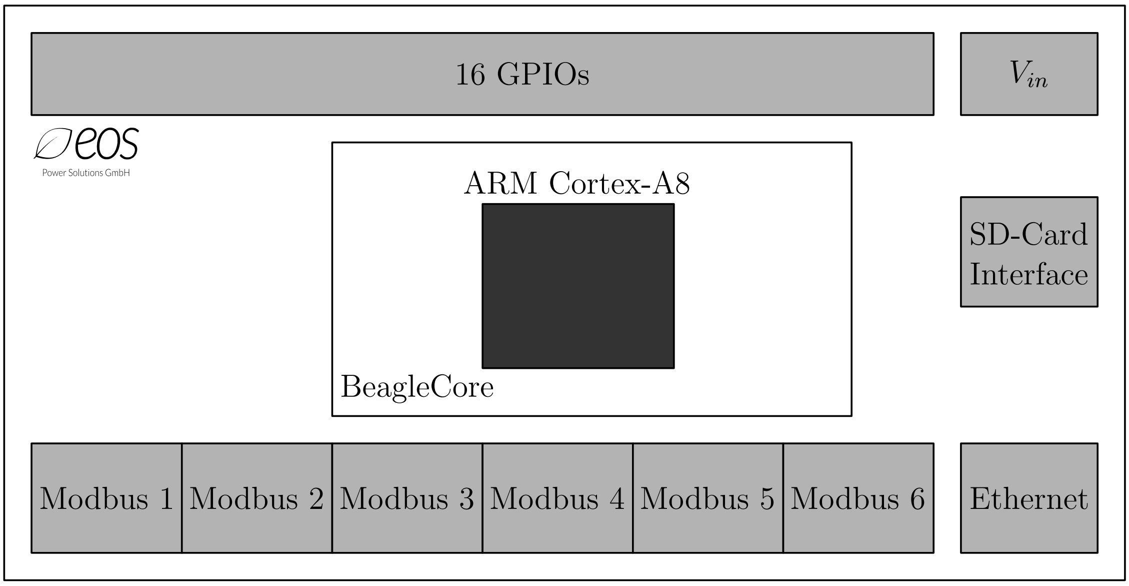



Figure 4.13: Sketch of EOS Energy Manager (EEM) circuit board with interfaces.

### macchina.io

Macchina.io is an IoT Software Development Kit (SDK) for creating applications for embedded Linux systems in C++ and JavaScript. [OO] Due to its modular design, it is highly customizable. The base of macchina.io consists of different frameworks like the Open Service Platform (OSP) and libraries, optimized for use in embedded Linux systems. [mac20] The OSP enables the highly modular design through a robust plug-in and service model. According to the documentation, this framework offers an environment based on so-called bundles and services. A bundle is a summary of executable code, configuration, and resource data that can offer various services that extend the primary system's functional scope. Each of these services is registered in a particular service registry and is therefore also available to other bundles globally. A bundle can be dynamically installed, started, stopped, and removed without requiring a system-wide restart. [mac20]

### Peak Shaving Bundle

To meet the specification of a bundle, the directory structure shown in Figure 4.14 must be followed. The bundle consists of two main components: The algorithm and the visualization. The algorithm is implemented in C++ and is located in the folder `src`. Additionally, a bundle activator must be implemented, which is the interface to the OSP. The folder `bundle/webapp` contains all files and subdirectories necessary for the visualization. The visualization is based on web technologies like HTML, CSS, and JavaScript.

```
PeakShaving
├─ bundle
│  ├─ webapp
│  │  └─ html, css, js files
│  ├─ bundle.properties
│  └─ extensions.xml
├─ src
│  ├─ BundleActivator.cpp
│  └─ further C++ code
├─ Makefile
└─ PeakShaving.bnldspec
```

Figure 4.14: Directory structure for a valid bundle.

The file `PeakShaving.bndlspec` contains important metadata in xml format for the bundle to run. It must be built precisely to specification so that the bundle works correctly. The exact specification can be found in the documentation of macchina.io. [mac20] In contrast, `extensions.xml` contains the necessary information about the web interface. Besides icons and UI-related things, the server directories for this bundle can be defined here.

**BundleActivator** To make a bundle executable, it must implement two methods in the file `BundleActivator.cpp`: `start()` and `stop()`. (See Figure 4.17.) In the method `start()` the runtime environment and context are provided and must be stored for further interactions with the OSP. This gives the bundle access to the ServiceRegistry where all services from the system or other bundles are registered and can be managed. In addition, the context provides essential interfaces for development, such as a logger. If a bundle is terminated, the method `stop()` is invoked automatically. Here it must be ensured that all stored references are adequately released, and the bundle is brought into a safe state that allows a clean restart.

### Visualisation

The visualization of the algorithm is realized with web technologies. `macchina.io` already offers some interfaces for this purpose. The power demand, grid consumption, peak limit, and SOC are displayed graphically in a diagram for the current overview. This allows the current status of the peak shaving and the necessary components to be seen at a glance. In addition to the diagram, the last calculated status and some statistical key figures are output textually. Figure 4.15 shows a screenshot of the visualization.

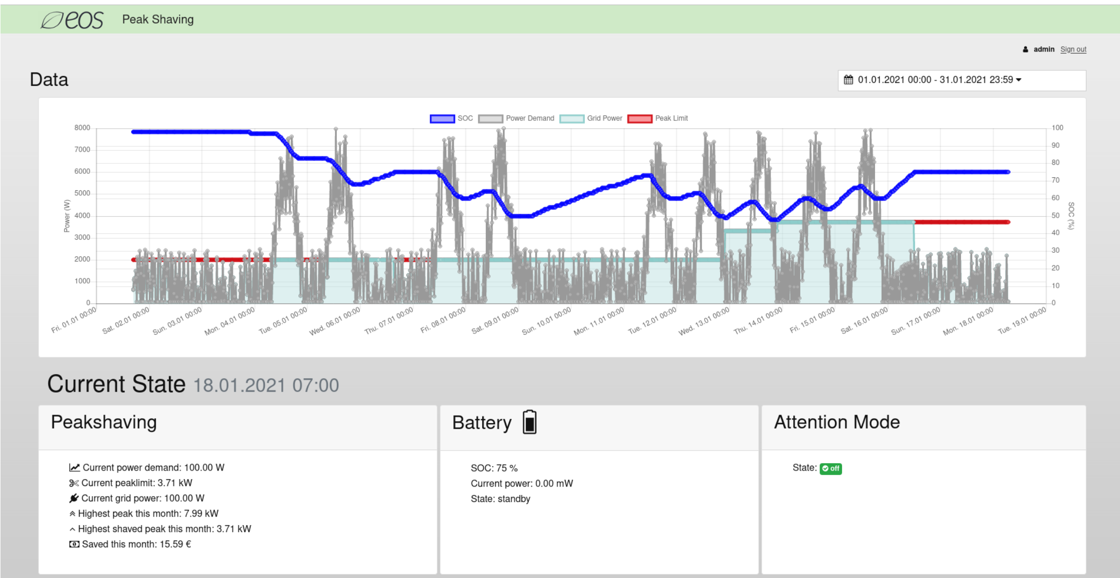



Figure 4.15: Screenshot of web based user interface.

## 4.4 Software architecture

This subsection describes the implementation of the peak shaving algorithm based on Kruchten's "The 4+1 View Model of Architecture" [Kru95]. The software architecture is described from four different perspectives: logical view, process view, physical view, and development view. In addition, the interaction of all components is illustrated in a so-called scenario.

### 4.4.1 Logical view and classes

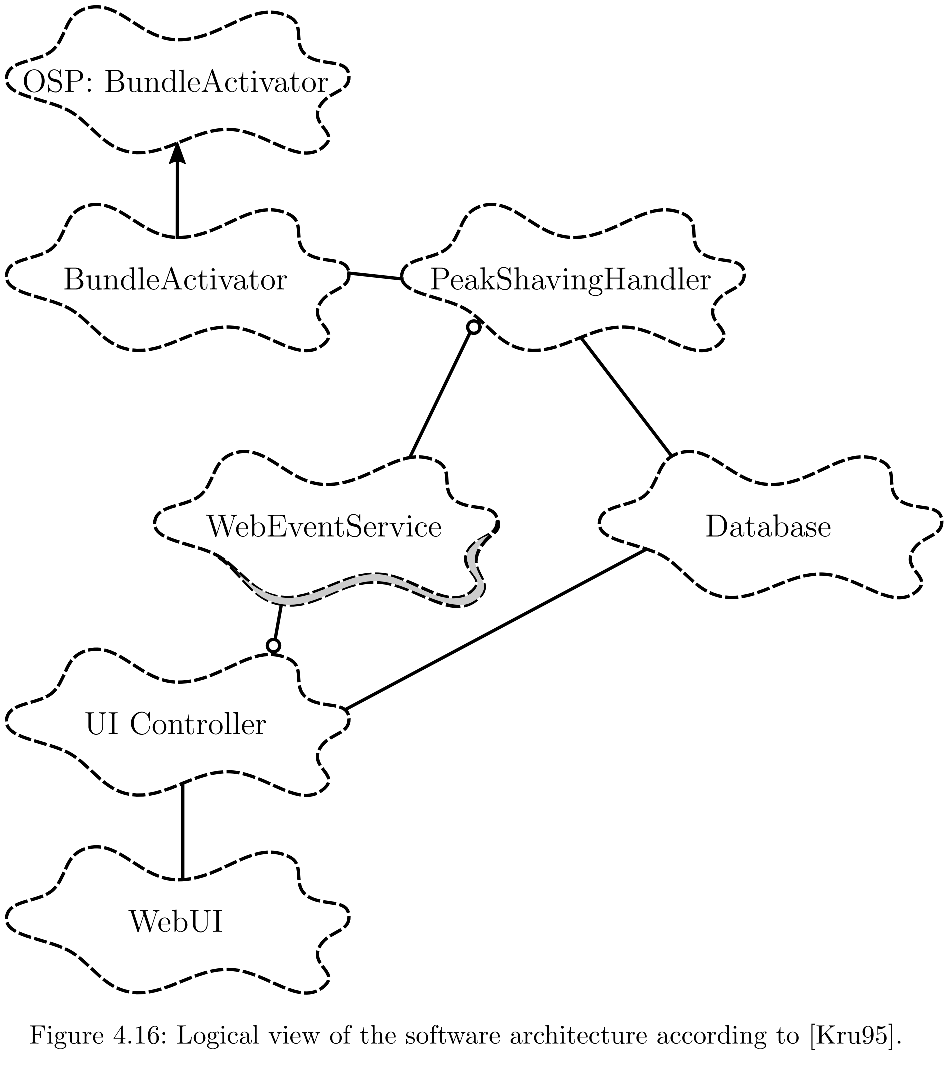



Figure 4.16: Logical view of the software architecture according to [Kru95].

Figure 4.16 shows the logical structure of the software. Each node in the graph represents a class in the software. All classes developed for this work are listed and explained in more detail below.

### BundleActivator

The `BundleActivator` class inherits from `OSP:BundleActivator` and is responsible for building and initializing all components of the peak shaving bundle. When running the `start`-method, all user-defined parameters placed in a config-file get loaded with the help of `PreferencesService` which is provided by OSP. Besides the IDs of all sensors required for peak shaving, the configuration file also contains other important parameters, such as the SOC limits. The class holds references for all necessary sensors and builds an instance of the `PeakShavingHandler`. Any time a sensor value updates, the `PeakShavingHandler` gets informed via the `handleSOC()`-method or the `handlePowerDemand()`-method. Figure 4.17 shows the (simplified) UML class diagram for the `BundleActivator`. Some utility functions are excluded from the graphic for better clarity.

| **BundleActivator : OSP::BundleActivator** |
|---|
| - _pContext : BundleContext<br>- _pPrefs : PreferencesService<br>- _pListener : DeviceListener<br>- _SOCSensorId : string<br>- _demandSensorId : string<br>- _batteryPowerSensorId : string<br>- _batteryPowerRequestensorId : string<br>- _pSOCSensor : ISensor*<br>- _pDemandSensor : ISensor*<br>- _pBatteryPowerSensor : ISensor*<br>- _pBatteryPowerRequestSensor : ISensor*<br>- _peakshavingHandler : PeakShavingHandler |
| + start(pContext : BundleContext *) : void<br>+ stop(pContext : BundleContext *) : void<br># handlePowerDemand(demand : double) : void<br># handleSoc(soc : double) : void |

Figure 4.17: UML class diagram for BundleActivator.

### PeakShavingHandler

After a `PeakShavingHandler` has been built by the constructor, which, in addition to the `BundleContext`, takes all configuration values relevant for the algorithm summarized in a structure as parameters, the references to the `BatteryPowerSensor` and the `BatteryPowerRequestSensor` must be set with the corresponding setter methods in the `PeakShavingHandler`. In the context of `macchina.io`, instances of sensors can be both sensors in the classical sense and actuators. `BatteryPowerSensor` represents the Modbus interface to the inverter, which passes on the values sent to

| **PeakShavingHandler** |
|---|
| - _pContext : BundleContext* |
| - _pWebEventService : WebEventService* |
| - _pMailService : MailService* |
| - _databasePath : string |
| - _pDatabase : Database* |
| - _pBatteryPowerSensor : ISensor* |
| - _pBatteryPowerRequestSensor : ISensor* |
| - _timer : Timer |
| - _storageCapacity : int |
| - _socMin : int |
| - _socMax : int |
| - _batteryLowThreshold : int |
| - _batteryCriticalThreshold : int |
| - _batterySafetzLimit : int |
| - _maxInverterPower : double |
| - _invertBatterySensors : bool |
| - _dayStartHour : int |
| - _dayEndHour : int |
| - _dayLen : double |
| - _nightLen : double |
| - _initialisationLimit : double |
| - _deltaT : double |
| - _numberOfDazsForMean : int |
| - _numberOfDazsToLoad : int |
| - _socMonitorTime : int |
| - _powerDemand : double |
| - _soc : double |
| - _powerBattery : double |
| - _powerGrid : double |
| - _currentPeakLimit : double |
| - _currentMonth : int |
| - _lastDay : int |
| - _currentDayIsHoliday : bool |
| - _PSum : double |
| - _workdaySums : vector<double> |
| - _attentionMode : bool |
| - _attentionValues : vector<double> |
| - _attentionCounter : int |
| - _attentionModeThreshold : double |
| - _attentionModeApplyDelay : double |
| - _socValues : vector<double> |
| - _numberOfSocValues : int |
| - _PValuesValues : vector<double> |
| - _numberOfPValues : int |
| ... |

Figure 4.18: UML class diagram for PeakShavingHandler (part I).

| **PeakShavingHandler** |
|---|
| ... |
| + PeakShavingHanlder(pContext : BundleContext*, params : struct Params) : void<br>+ PeakShavingHanlder(params : struct Params) : void<br>+ handlePowerDemand(demand : double) : void<br>+ setSOC(soc : double) : void<br>+ setPowerDemand(powerDemand : double) : void<br>+ setBatteryPowerSensor(sensor : ISensor*) : void<br>+ setBatteryPowerRequestSensor(sensor : ISensor*) : void<br>+ setWebEventService(wes WebEventService*) : void<br>+ notifyWebEvent() : void<br># checkHoliday(date : DateTime) : bool<br># setInverterPower(pInverter : double) : void<br># approximatelyEqual( : double, b : double, epsilon : double) : bool<br>- _getDateKeyString(int d, int m, int y) : string<br>- _getDateKeyString(Poco::DateTime d) : string<br>- _getEaster(int year) : DateTime |

Figure 4.19: UML class diagram for PeakShavingHandler (part II).

the battery controller. `BatteryPowerRequestSensor` provides the desired inverter power as a value for verification.

Every 15 minutes the `PeakShavingHandler` receives the current power demand and the method `setPowerDemand(double demand)` is invoked. This method handles the new demand value and calculates the peak limit valid for the next time interval. Because the holidays of the current year are also relevant for the calculation, they are calculated dynamically at the program's runtime. The `checkHoliday()` method checks whether a passed date falls on a holiday. Based on the demand value and all stored information, the next time interval's battery power is calculated and transmitted to the inverter. After the battery power has been sent to the inverter, a timer is started which checks after one minute whether the desired power has been applied correctly and sends a warning by e-mail in the event of an error. Subsequently, all values calculated by the algorithm and some intermediate results are stored in a database. After writing the data to the database, an update event is sent (with a few seconds delay) via the available `WebEventService`, which informs other services such as the user interface about the new records. The time delay is used to ensure the completion of the data transaction.

### Database

| Database |
| --- |
| - _mutex : FastMutex<br>- _logger : Logger<br>- _insertData : struct InsertData<br>- _insertStatement : Statement<br>- _getLatestPeaklimitStatement : Statement<br>- _latestPeaklimit : struct TimestampValue<br># _scheduler : TimerTaskScheduler |
| + Database(path : string)<br>+ insert(insertData : struct InsertData) : void<br>+ getLatestPeaklimit() : struct TimestampValue |

Figure 4.20: UML class diagram for Database.

The `Database` class (see Figure 4.20) manages all database related features. When a `Database` object is created, it automatically checks if a peak shaving database already exists and, if not, generates a new SQLite database. The location can be specified in the configuration file. If no preferences are provided regarding the storage location, a database named `peakshaving.sqlite` will be created in the data directory. The database itself consists of several tables which are explained in the following.

**Table 'peakshaving'** stores all values relevant for the peak shaving algorithm, which are updated at 15-minute calculation intervals. The structure of the database table is shown in Table 4.13.

Table 4.13: Structure of the database table peakshaving.

| Name | Datatype |
| --- | --- |
| id | INTEGER (primary key) |
| timestamp | INTEGER |
| p_demand | REAL |
| p_limit | REAL |
| soc | REAL |
| attention_mode | BOOL |
| error_empty_battery | BOOL |
| error_undersized_inverter | BOOL |

For querying the latest peak limit from the database a statement is prepared:

```
SELECT timestamp, p_limit FROM peakshaving ORDER BY
    timestamp DESC LIMIT 1
```

For inserting a set of new values the following statement is prepared:

```
INSERT INTO peakshaving (timestamp, p_demand, p_limit,
    soc, attention_mode, error_empty_battery,
    error_undersized_inverter) VALUES (?, ?, ?, ?, ?, ?, ?)
```

**Table 'workdaysums'** holds the total sums of the working days updated at the end of each cycle at 22:00. To save memory space, only as many values are stored in this table are necessary for calculating the peak limit. The required number is given by parameter `number_of_days_for_mean`. See Table 4.14 for the table structure. To

Table 4.14: Structure of the database table workdaysum.

| Name | Datatype |
|---|---|
| id | INTEGER (primary key) |
| timestamp | INTEGER |
| sum | REAL |

limit the number of stored values to the most recent `number_of_days_for_mean`, the following statement is executed after inserting a new row into the database.

```
DELETE FROM workdaysums WHERE id <= (SELECT id FROM
    (SELECT id FROM workdaysums ORDER BY timestamp DESC
    LIMIT 1 OFFSET ?) foo)
```

The value of `number_of_days_for_mean` set by the user is used as the parameter in this SQL statement.

**Table 'powersum'** stores the sum of the power values of the current cycle. This allows the algorithm to continue working with the previous data even if the system is restarted. After a reboot, the sum value is read from the database and used, as long as it is not older than 15 minutes. Otherwise, it is discarded and the algorithm starts with a new (incomplete) cycle. Table 4.15 shows the structure of this database table.

**Table 'powervalues'** holds the power data (see Table 4.16). Each time the current power demand changes, the battery power is adjusted so that the grid draw remains approximately constant. All three power values are permanently stored in the table.

Table 4.15: Structure of the database table powersum.

| Name | Datatype |
|---|---|
| id | INTEGER (primary key) |
| timestamp | INTEGER |
| sum | REAL |

Table 4.16: Structure of the database table powervalues.

| Name | Datatype |
|---|---|
| id | INTEGER (primary key) |
| timestamp | INTEGER |
| p_demand | REAL |
| p_grid | REAL |
| p_batt | REAL |

### 4.4.2 Process view

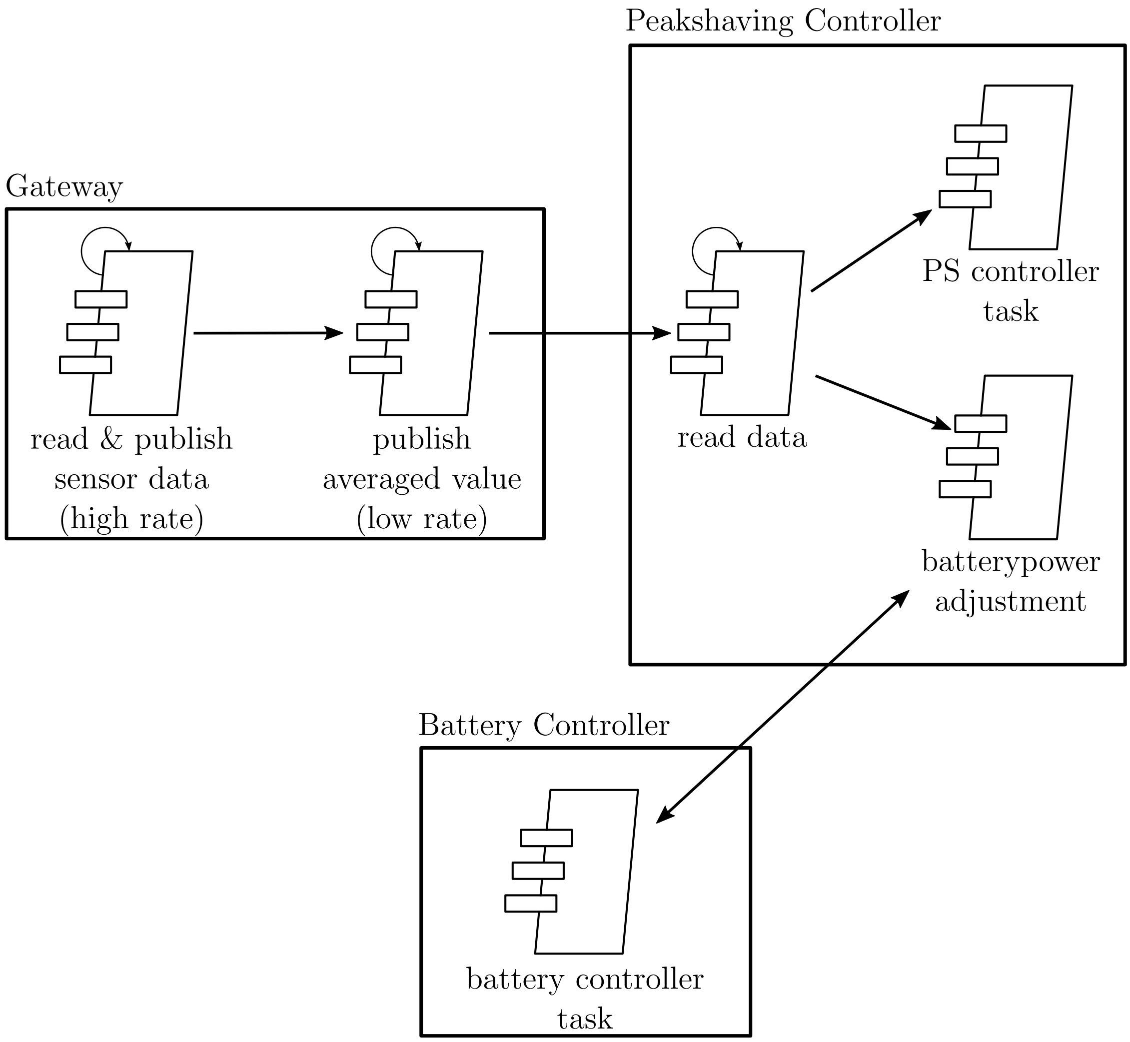



Figure 4.21: Process view of the software architecture according to [Kru95].

The process view according to [Kru95] is shown in Figure 4.21. As already described, there are two independent devices: the gateway and the peak shaving controller. In addition, there is a battery control device that regulates the battery power through the inverter. The battery controller was not developed in the course of this thesis.

The gateway's task is the continuous reading of the smart meter, which is connected to the gateway via a Modbus interface.

For each time interval of 15 minutes, the measured power values are recorded and numerically integrated over time (see process `Read and publish Sensor Data` in Figure 4.21). At the end of each time interval, the value divided by the total time is published on the MQTT broker and thus made available to the peak shaving controller as a new power demand value (see process `Publish Averaged Value` in Figure 4.21).

In order to ensure a continuous adjustment of the battery power to the current power demand, all measured sensor values are also published at the broker in addition to the averaged value (see process `Read and publish Sensor Data` in Figure 4.21).

The `Read Data` process at the Peakshaving Controller is notified by the broker when new data is available and triggers the `PS Controller Task`, which uses the developed algorithm to calculate a peak limit that applies for the next quarter-hour. This, in turn is passed on to the inverter, which regulates the battery power via the `Battery Controller` process.

### 4.4.3 Development view

The development view defines the layers of development. Each layer depends on the lower ones but not on the layers above.

Table 4.17: Development view of the software architecture according to Kruchten. [Kru95]

| |
|---|
| UI, Man-Machine-Interface |
| Inter Process Communication (WebEventService) |
| Peakshaving Control Algorithm |
| Database |
| Hardware I/O (Sensors, actuators) |

As shown in Table 4.17, the lowest layer is the hardware communication. Through the sensors and actuators, the system can interact with its environment. The current power demand is transmitted to the system via the Modbus interface and is ready for further processing. Communication with the inverter, which is the actuator in this case, also takes place via a Modbus interface through which the setpoint of the battery power is specified. It then regulates the actual power to the specified value.

The database offers the possibility to store past sensor values and intermediate results calculated from them permanently. This is essential for the correct determination of the peak limit.

The peak shaving control algorithm determines a peak limit from the current sensor data and the past values stored in the database, which is always valid for the next time interval. This, as well as the intermediate results necessary for the calculation, are stored in the database.

The WebEventService connects the two layers of the algorithm and the user interface.

The UI forms the highest layer and facilitates a user-friendly interaction between the system and the user. It contains a graphical presentation of the data as well as a status display of the entire system with all components.

### 4.4.4 Physical view

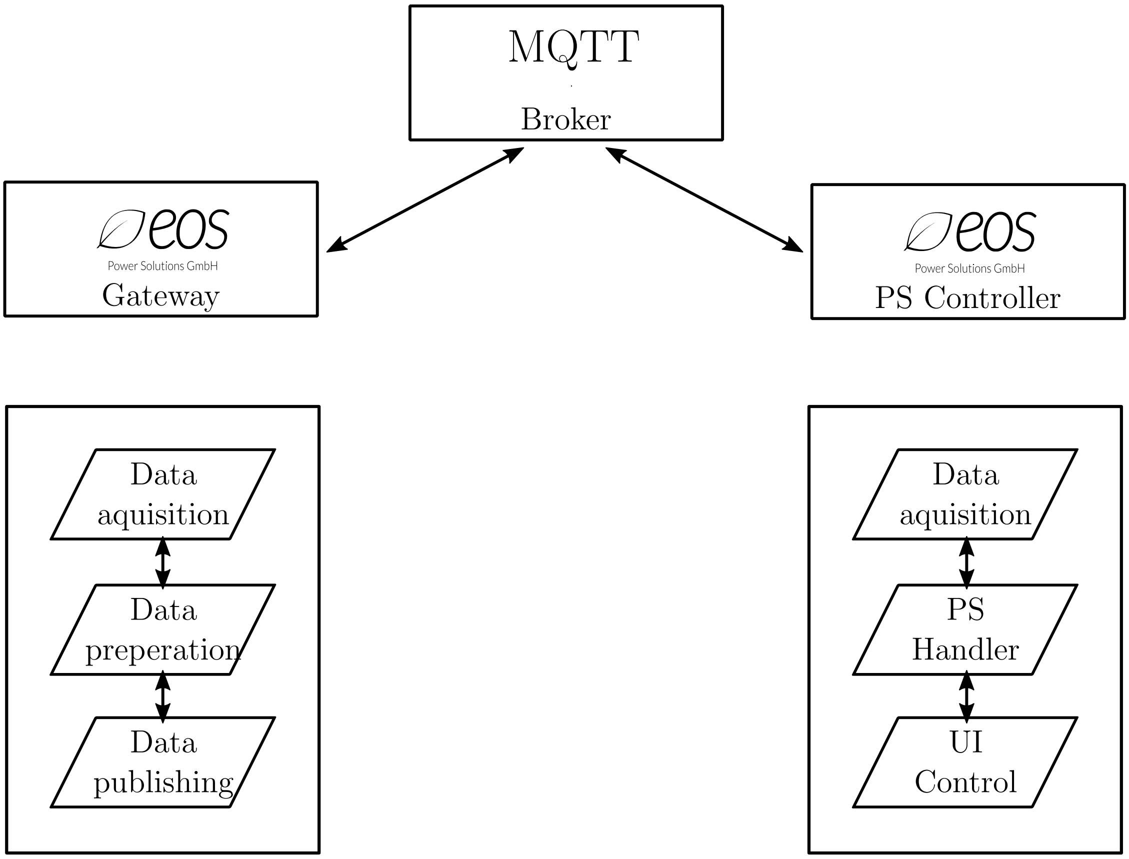



Figure 4.22: Physical view of the software architecture according to [Kru95].

Figure 4.22 shows the physical view of the overall system. Since a direct connection of the Modbus meter to the peak shaving control system is not always possible due to local conditions, a separation into the gateway and peak shaving controller was chosen, as can already be seen in the process view. The gateway handles the acquisition of the meter data and its processing. The peak shaving controller then manages the actual regulation and communication with the inverter. An MQTT broker serves as a bridge between the two devices.

Figure 4.23 shows all components and interfaces of the system for a better overview.

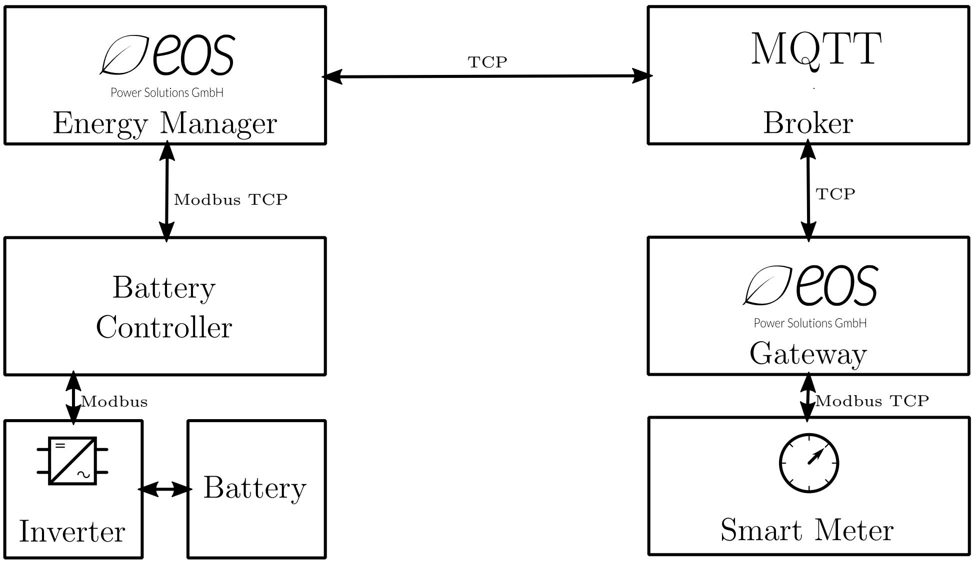



Figure 4.23: Overview of the system composition and the interfaces.

### 4.4.5 Scenario

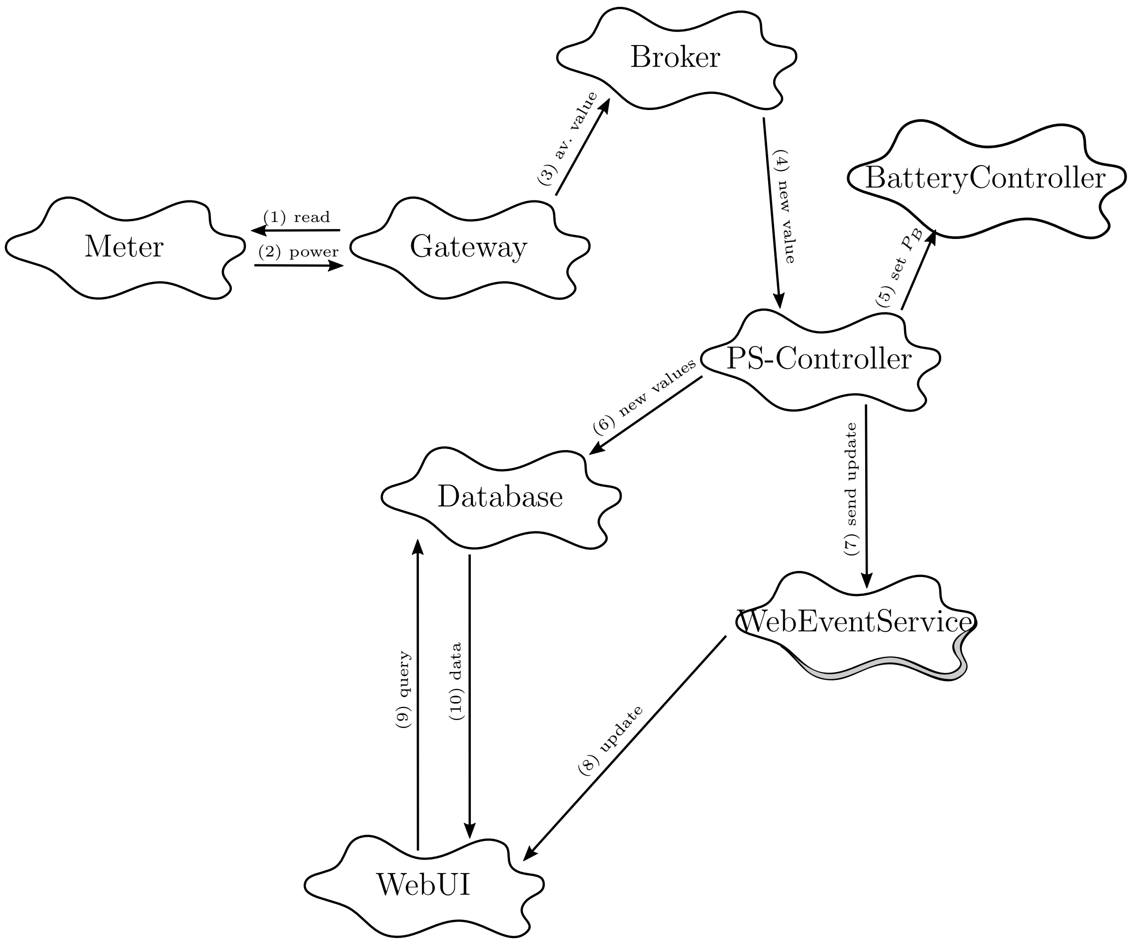



Figure 4.24: Scenario of the software architecture according to [Kru95].

Figure 4.24 shows a typical procedure of the system from the reading of the sensor value to the visualization on the user interface. Since the gateway is the Modbus master, it initiates the reading of the current power value from the meter `(1)`, which then returns the desired sensor value `(2)`. This process is continuously repeated until the defined time interval of 15 minutes has elapsed. All recorded values are then averaged, and the resulting power is transmitted to the broker `(3)`, which immediately informs the peak shaving controller about the presence of a new value `(4)`.

The controller determines the necessary battery power for the next time interval and sends it to the battery controller via the inverter `(5)`. Subsequently, the newly calculated values and the measured power demand are stored in the database `(6)`.

After the data transaction is completed, the WebEventService is informed about the new database entry `(7)`, which automatically causes the user interface to execute an update event `(8)`. In this event, the data requested by the user is retrieved from the database `(9)`, and the returned data records `(10)` are graphically prepared.

A similar scenario occurs when a new (unaveraged) power value is available. The value read by the smart meter is published directly to the broker `(3)`, which informs the peak shaving controller `(4)`. The controller then adjusts the battery power to the current conditions ($P_{Batt} = P_{demand} - P_{limit}$) so that the power drawn from the grid does not exceed the peak limit, which is calculated only using averaged values over a 15-minute interval `(5)`. The new data is written to the database `(6)`, and the WebUI is updated via the WebEventService `(7 -10)`.

# 5 Evaluation

To evaluate the profitability of peak shaving, a cost-benefit calculation is carried out based on the simulation data. This involves varying demand prices and battery storage sizes to achieve the best result and achieve the system components' optimum configuration where the yield is highest.

## 5.1 Cost Function

Battery life of 10 years is assumed for the evaluation. Due to the lack of available data over such a long period, the electricity cost savings are considered to be approximately the same over all years. The battery storage system costs are assumed to be 300 Euro per kilowatt hour, which is currently the standard on the market. The investment costs are offset against the annual savings in electricity costs over the battery life and evaluated for different battery storage sizes. This allows the storage size to be determined at which the yield is highest.

With the current demand price and the presently necessary investment costs for battery storage systems, an economic operation of peak shaving based on the existing performance profiles is not possible. (See solid black line in Figure 5.1.)

It can be assumed that demand prices will rise in the coming years. More and more photovoltaic systems are being connected to the grid, thus stressing the grid infrastructure. In Germany, the current demand prices are already twice as high as in Austria. For this reason, the simulation was also carried out with different demand prices. The results show that, as was to be expected, the yield increases with rising power prices. Figure 5.1 illustrates the expected profits at different demand prices depending on the size of the battery storage.

According to Holland, the average investment costs for battery storage will drop to about € 100 per kilowatt hour by 2023. [Hol19] This creates new opportunities for peak shaving to be more economical. For this reason, further evaluation with different battery storage prices is done.

Figure 5.2 shows the results for a storage price of € 200 per kilowatt hour. At the current low demand price in Austria, however, this is still not profitable. Only from € 100 per kilowatt, an economical operation is possible. With the electricity tariffs in Germany, though, a positive balance at the end of the battery life would already be feasible under these conditions.

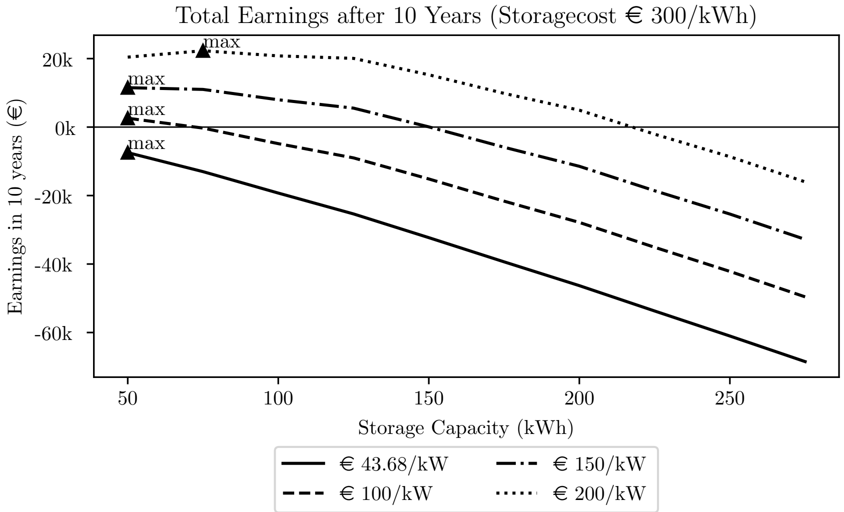



Figure 5.1: Total earnings over battery capacity at battery storage cost of € 300 per kilowatt-hour.

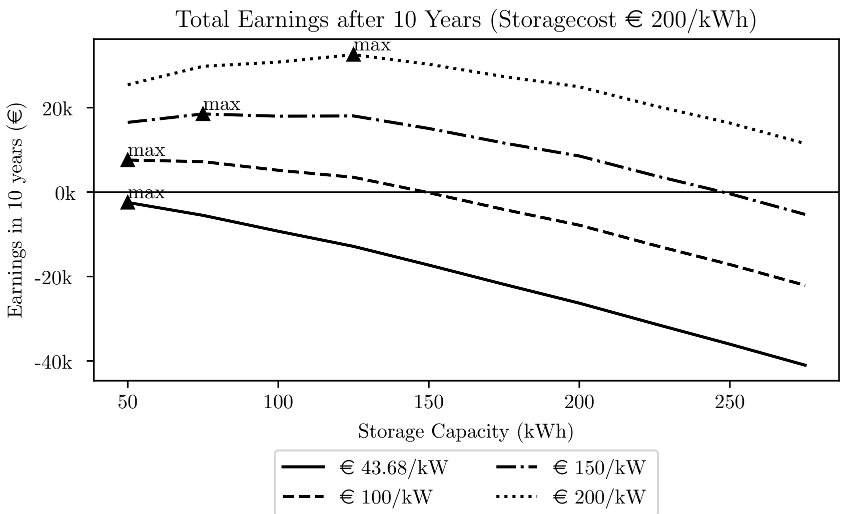



Figure 5.2: Total earnings over battery capacity at battery storage cost of € 200 per kilowatt-hour.

Suppose the battery costs continue to fall, and battery storage is available from 100€ per kilowatt-hour. In that case, an economical operation is possible even at the current demand prices, as shown in Figure 5.3. If the demand price still rises, the operation of peak shaving becomes more and more profitable and leads, depending on the demand price, to a return between 10 000 and over 40 000 Euro at the end of the battery lifetime of 10 years.

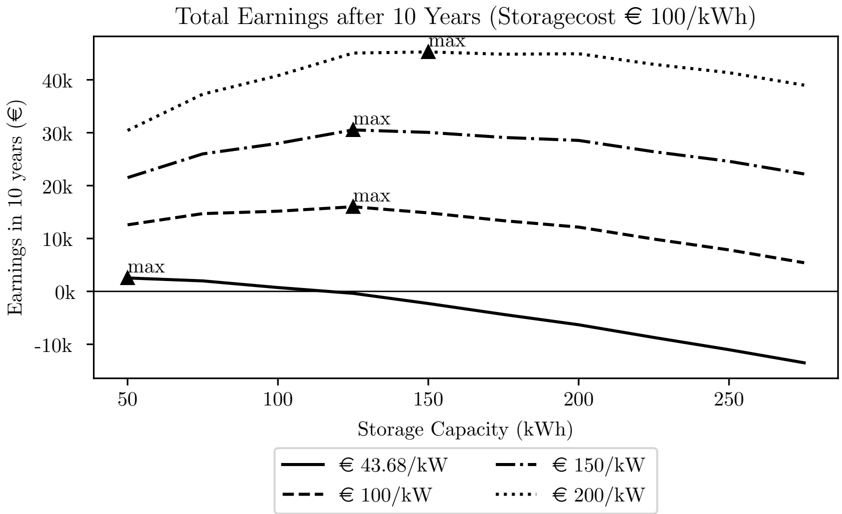



Figure 5.3: Total earnings over battery capacity at battery storage cost of € 100 per kilowatt-hour.

## 5.2 Discussion

The result of this thesis is a working algorithm that, in contrast to most of those found in the literature, computes a dynamic peak limit live, *i.e.* at runtime. This does not require past data models and load profiles, as is the case with the current state of the art. The optimal peak limit calculation for each billing period depends on the storage size, the maximum inverter power, and the power demand. Demand data is collected over the runtime to be used in further computations.

As the evaluation shows, cost recovery or profitable operation is not possible at current power costs and storage prices. Therefore, investing in an energy storage system only for peak shaving reasons is not target-oriented. However, the trend shows a clear direction. Following this trend, both, the investment costs in energy storage will decrease, and the performance costs will increase in the future. These circumstances make peak shaving interesting in the coming years.

Nevertheless, in most cases, energy storage systems are installed in connection with photovoltaic systems and are thus already available. In this case, a cost-saving can be achieved by applying peak shaving since the investment does not have to be amortized purely by peak shaving.

The developed overall system is ready as a prototype and will go into a test run. However, the results of this test phase will not be available for about a year and are therefore not included in this work.

# 6 Conclusion and Outlook

The aim of this thesis was to develop a peak shaving algorithm that finds an optimal peak limit for each accounting period in terms of cost reduction. A further limitation is that the calculations are performed on an embedded system, and the computing power is therefore limited. Furthermore, no power demand profiles are available over a more extended period of time, which is why the algorithm can only rely on data collected at runtime.

Different approaches were tried, evaluated, and analyzed. Finally, the analyses led to a combined algorithm that brings together the advantages of a rigid analytical part with those of a flexible, responsive part. For the majority of the time, the power demand follows approximately a recurring pattern. While it is almost consistently low during the nighttime, it increases significantly during the morning hours and peaks midday. In the evening hours, demand falls back to the nighttime level. On holidays or weekends, there is no increase in the morning. The power curve remains approximately at night level. The algorithm's analytical part uses this repetitive pattern to calculate a peak limit at the end of each cycle, starting from the beginning of the night at 10 p.m. and lasts precisely 24 hours. The limit is determined so that the energy available to charge the energy storage device is exactly equal to the energy extracted from the battery. This allows the energy storage to start each cycle with the same charge level theoretically.

However, since the limit is only determined at the end of a cycle and is only partially optimal for the next cycle, the responsive part is used. It consists of two components, the so-called Attention Mode, and SOC monitoring. The former tries to predict and approximate the optimal peak limit based on the power demand's current course. If this approximated limit is higher than the currently valid one over a longer period of time, it is adjusted. This is to ensure that the algorithm also works for cycles that are not similar.

SOC monitoring keeps track of the battery's charge level and prevents it from running out of charge. When the charge level drops and is no longer sufficient until the next recharge time, the peak limit is increased early to relieve the battery while not increasing the peak limit disproportionately, negating the savings effect in power costs.

In simulations, the developed algorithm's application resulted in savings in power-related costs of over 35%. Unfortunately, at current power prices, which are still relatively low compared to international standards, the savings in total costs do not result in large gains. For this reason, the investment costs for an energy storage system of sufficient size could not be amortized by the use of peak shaving alone. Due to the extensive expansion of renewable energy and the subsidies in this regard, the investment costs will decrease significantly in the future, and the power-related charges for electricity will increase. This opens up more opportunities for peak shaving to achieve savings as well.

In many cases, however, an energy storage system is already available in combination with a photovoltaic plant and can also be used for peak shaving. This allows peak shaving to be used and leads to higher yields since the investment does not have to be covered by peak shaving alone.

## 6.1 Outlook

Since peak shaving currently achieves the best savings effect combined with a Photovoltaic (PV) system, it makes sense to integrate it into the algorithm. This means that the battery can be charged during good weather conditions without having to draw power from the grid. Besides, the self-consumption of the energy generated by the PV, which, according to research, is on average only about 30%, can be increased and thus achieve further cost savings.

In the course of the expansion of renewable energy sources, more and more PV systems will be brought into operation. However, this puts a strain on the utility grids, as excess energy from the PV system has to be fed into the grid. The more systems that feed into the grid, the more difficult it is for the infrastructure to distribute the energy. Sunny summer days, when solar irradiation is high, result in a large surplus of energy in these regions that is fed into the grid.

To counteract a large-scale and thus expensive expansion of the supply networks, it is evident that in the future, the feeding in of high power will be penalized in some form. A power-related charge, as is customary for the drawing of power, is conceivable.

This opens up another application for peak shaving. By applying the peak shaving algorithm, the power costs incurred by the feed-in can be reduced, and the self-consumption increases. To do this, it is necessary to use the battery, possibly through weather forecasts and based on past data, in such a way that there is enough energy for high power demands (peaks) but also enough capacity for storing excess energy from the PV system, which thus does not have to be fed into the grid.

# Appendix

# Nomenclature

**BESS** Battery Energy Storage System

**DSM** Demand Side Management

**DP** Dynamic Programming

**EEM** EOS Energy Manager

**ESS** Energy Storage System

**EV** Electric Vehicle

**LF** Load Factor

**OSP** Open Service Platform

**PV** Photovoltaic

**SOC** State of Charge

# Bibliography


[Ana+16] K. Ananda-Rao et al. “Microcontroller Based Battery Controller for Peak Shaving Integrated with Solar Photovoltaic.” In: *4th IET Clean Energy and Technology Conference (CEAT 2016).* 4th IET Clean Energy and Technology Conference (CEAT 2016). Kuala Lumpur, Malaysia: Institution of Engineering and Technology, 2016, 5 (6 .)–5 (6 .) ISBN: 978-1-78561-238-1. DOI: 10.1049/cp.2016.1262. URL: https://digital-library.theiet.org/content/conferences/10.1049/cp.2016.1262 (visited on 09/16/2020) (cit. on p. 11).

[BMH18] Neda Boyouk, Nina Munzke, and Marc Hiller. “Peak Shaving of a Grid Connected-Photovoltaic Battery System at Helmholtz Institute Ulm (HIU).” In: *2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe).* 2018 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe). Sarajevo, Bosnia and Herzegovina: IEEE, Oct. 2018, pp. 1–5. ISBN: 978-1-5386-4505-5. DOI: 10.1109/ISGTEurope.2018.8571616. URL: https://ieeexplore.ieee.org/document/8571616/ (visited on 09/16/2020) (cit. on p. 11).

[Dig20] Bundesministerium für Digitalisierung und Wirtschaftsstandort. *RIS - Systemnutzungsentgelte-Verordnung 2018 - Bundesrecht Konsolidiert, Fassung Vom 23.03.2020.* 2020. URL: https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=20010107 (visited on 03/23/2020) (cit. on p. 18).

[E-C20a] E-Control. *Das Stromnetz.* 2020. URL: https://www.e-control.at/konsumenten/das-stromnetz (visited on 09/23/2020) (cit. on p. 6).

[E-C20b] E-Control. *Netznutzungsentgelt.* 2020. URL: https://www.e-control.at/industrie/strom/strompreis/netzentgelte/netznutzungsentgelt (visited on 09/23/2020) (cit. on p. 7).

[E-C20c] E-Control. *System Charges.* 2020. URL: https://www.e-control.at/industrie/strom/strompreis/netzentgelte (visited on 09/23/2020) (cit. on p. 6).

[Fro20] Fronius. *Fronius Energy Package.* Technische Daten. Mar. 3, 2020. URL: https://fronius.com/~/downloads/Solar%20Energy/Datasheets/SE_DS_Fronius_Energy_Package_DE.pdf (visited on 03/03/2020) (cit. on p. 22).

[Har+20] Charles R. Harris et al. “Array Programming with NumPy.” In: *Nature* 585.7825 (7825 Sept. 2020), pp. 357–362. ISSN: 1476-4687. DOI: 10.1038/s41586-020-2649-2. URL: https://www.nature.com/articles/s41586-020-2649-2 (visited on 10/06/2020) (cit. on p. 21).

[Hol19] Maximilian Holland. *Powering The EV Revolution — Battery Packs Now At $156/kWh, 13% Lower Than 2018, Finds BNEF*. CleanTechnica. Dec. 4, 2019. URL: https://cleantechnica.com/2019/12/04/powering-the-ev-revolution-battery-packs-now-at-156-kwh-13-lower-than-2018-finds-bnef/ (visited on 09/18/2020) (cit. on p. 68).

[Hun07] John D. Hunter. “Matplotlib: A 2D Graphics Environment.” In: *Computing in Science Engineering* 9.3 (May 2007), pp. 90–95. ISSN: 1558-366X. DOI: 10.1109/MCSE.2007.55 (cit. on p. 21).

[iEne] inek Energie GmbH. *DC-Gekoppelte Speicher-Systeme — Inek.De*. URL: http://www.inek.de/drupal/?q=node/31 (visited on 09/24/2020) (cit. on p. 6).

[JP15] Kalpesh A. Joshi and Naran M. Pindoriya. “Day-Ahead Dispatch of Battery Energy Storage System for Peak Load Shaving and Load Leveling in Low Voltage Unbalance Distribution Networks.” In: *2015 IEEE Power Energy Society General Meeting*. 2015 IEEE Power Energy Society General Meeting. July 2015, pp. 1–5. DOI: 10.1109/PESGM.2015.7285673 (cit. on p. 3).

[KL15] Siong Lee Koh and Yun Seng Lim. “Evaluating the Economic Benefits of Peak Load Shifting for Building Owners and Grid Operator.” In: *2015 International Conference on Smart Grid and Clean Energy Technologies (ICSGCE)*. 2015 International Conference on Smart Grid and Clean Energy Technologies (ICSGCE). Oct. 2015, pp. 30–34. DOI: 10.1109/ICSGCE.2015.7454265 (cit. on p. 4).

[Kru95] P. B. Kruchten. “The 4+1 View Model of Architecture.” In: *IEEE Software* 12.6 (Nov. 1995), pp. 42–50. ISSN: 1937-4194. DOI: 10.1109/52.469759 (cit. on pp. 55, 62–64, 66).

[LC16] Alexandre Lucas and Stamatios Chondrogiannis. “Smart Grid Energy Storage Controller for Frequency Regulation and Peak Shaving, Using a Vanadium Redox Flow Battery.” In: *International Journal of Electrical Power & Energy Systems* 80 (Sept. 1, 2016), pp. 26–36. ISSN: 0142-0615. DOI: 10.1016/j.ijepes.2016.01.025. URL: http://www.sciencedirect.com/science/article/pii/S0142061516000375 (visited on 09/15/2020) (cit. on pp. 9, 11).

[LS12] Jason Leadbetter and Lukas Swan. "Battery Storage System for Residential Electricity Peak Demand Shaving." In: *Energy and Buildings*. Cool Roofs, Cool Pavements, Cool Cities, and Cool World 55 (Dec. 1, 2012), pp. 685–692. ISSN: 0378-7788. DOI: 10.1016/j.enbuild.2012.09.035. URL: http://www.sciencedirect.com/science/article/pii/S0378778812004896 (visited on 09/15/2020) (cit. on pp. 9, 10, 25, 26).

[mac20] machhina.io. *Introduction and Overview*. Oct. 27, 2020. URL: https://macchina.io/docs/00100-MacchinaIntroduction.html (visited on 10/27/2020) (cit. on pp. 52, 53).

[MM05] V.K. Mehta and Rohit Mehta. *Principles of Power System*. 4th Revised Edition. New Delhi: S. Chand Publications, 2005. URL: https://www.academia.edu/38737591/principles_of_power_system_by_v_k_mehta_and_rohit_mehta (visited on 09/09/2020) (cit. on p. 4).

[NKS08] Ali Nourai, V. I. Kogan, and Chris M. Schafer. "Load Leveling Reduces T D Line Losses." In: *IEEE Transactions on Power Delivery* 23.4 (Oct. 2008), pp. 2168–2173. ISSN: 1937-4208. DOI: 10.1109/TPWRD.2008.921128 (cit. on p. 2).

[OCB07] Alexandre Oudalov, Rachid Cherkaoui, and Antoine Beguin. "Sizing and Optimal Operation of Battery Energy Storage System for Peak Shaving Application." In: *2007 IEEE Lausanne Power Tech*. 2007 IEEE Power Tech. Lausanne, Switzerland: IEEE, July 2007, pp. 621–625. ISBN: 978-1-4244-2189-3. DOI: 10.1109/PCT.2007.4538388. URL: http://ieeexplore.ieee.org/document/4538388/ (visited on 09/16/2020) (cit. on pp. 2, 10).

[OO] Günter Obiltschnig and Maria Obiltschnig. *Macchina.Io - IoT Edge Device Software Development and Secure Remote Access Solutions*. macchina.io. URL: https://macchina.io (visited on 10/02/2020) (cit. on pp. 12, 52).

[Rah+13] A. Rahimi et al. "A Simple and Effective Approach for Peak Load Shaving Using Battery Storage Systems." In: *2013 North American Power Symposium (NAPS)*. 2013 North American Power Symposium (NAPS). Manhattan, KS, USA: IEEE, Sept. 2013, pp. 1–5. ISBN: 978-1-4799-1255-1. DOI: 10.1109/NAPS.2013.6666824. URL: http://ieeexplore.ieee.org/document/6666824/ (cit. on pp. 2, 10, 11).

[tea20] The pandas development team. *Pandas-Dev/Pandas: Pandas*. Version 1.0.1. Zenodo, Feb. 2020. URL: https://doi.org/10.5281/zenodo.3509134 (cit. on p. 21).

[Udd+18] Moslem Uddin et al. "A Review on Peak Load Shaving Strategies." In: *Renewable and Sustainable Energy Reviews* 82 (Feb. 2018), pp. 3323–3332. ISSN: 13640321. DOI: 10.1016/j.rser.2017.10.056. URL: https://linkinghub.elsevier.com/retrieve/pii/S1364032117314272 (cit. on pp. 2–4, 9).