UiA University of Agder

Master

System modeling and dispatch schedule optimization of combined PV battery system using linear optimization

Historic data, weather forecasts and day ahead spot prices are used as the conditional statements for this optimization. A simplified PV system and battery is modelled. Simulations are done in Python and HOMER. The values used are based on a case study for Avfall Sør in Lillesand, Norway.

BIRK LUDWIG SCHODER AIGNER

MAIN SUPERVISOR

Anne Gerd Imenes

CO-SUPERVISOR

Gunnstein Skomedal

University of Agder, [2021]

Faculty of [Grimstad]

Abstract

Energy storage plays a vital role in paving the way for more renewable penetration. The technology is costly, but intelligent solutions regarding dispatch strategies and system design can help reduce the total cost over the projected lifetime of a system.

For this thesis, a customizable linear programming algorithm is created within Python to optimize the battery energy scheduling based on generated PV power, electricity cost and load demand. The commercial system optimization tool HOMER is used to verify the code by running simulations based on historic data collected from NordPool and UiAs own photovoltaic system.

One benefit of the custom made code is its ability to do day-ahead optimization utilizing data from APIs. To obtain forecasted irradiation and temperature data the Solcast API was used as the only paid service for the Python script to ensure market-leading accuracy. Electricity prices were imported from the transparency platform Entsoe.

Implementing different incentives such as battery energy throughput and stored energy made it possible to alter the system behavior to optimize the system better while maintaining good working conditions for the battery. Carefully choosing the optimization parameters reduced battery cycling by 45.9% and average energy stored by 34.8% in two separate scenarios while maintaining cost savings. The case studied uses a power tariff power plan which means that the primary cost savings from including a battery come from peak shaving.

Preface

This thesis concludes my two-year Renewable Energy Master's Program at the University of Agder (UiA) in Grimstad, Norway, where I previously got my bachelor's degree in Renewable Energy. I chose to embark on the Master's as I wanted to expand my knowledge and improve my skill set to be better able to solve more complicated problems related to the subject.

The Master's thesis is a continuation of a previous study done in ENE503 on system dimensioning optimization using HOMER Pro and HOMER Grid. This sparked the interest in further understanding the subject of battery scheduling optimization and PV energy modelling.

I have also wanted to better understand how to solve complex problems using coding. In my opinion, most of the problems we are facing can be solved through intelligent technological solutions, and in that regard, coding is a large part of the solution. My background in coding is a brief introduction to data analysis and modelling techniques from the ENE418-G subject. During the start of this thesis, a lot of work was put into understanding and creating code that satisfied the requirements of the problem formulation.

Initially, I wanted to use the day ahead optimization in combination with a physical system at the university to see how this could be used together. Due to COVID-19, it was difficult to find people with the necessary knowledge to make this possible without investing much time in understanding the system from the ground up. The part of the code is still included in the thesis as proof of concept and to showcase the acquired knowledge regarding using data collected by APIs.

I want to thank my supervisors Anne Gerd Imenes and Gunnstein Skomedal for the support, guidance and sound advice. I also want to thank Gunn Spikkeland Hansen for providing load data and information for the case study.

My family and friends have been a great motivation during the Master's, especially since we all find ourselves in a challenging situation these days. I also want to thank my girlfriend for being supportive and keeping me company during this time.

Individual/group Mandatory Declaration

The individual student or group of students is responsible for the use of legal tools, guidelines for using these and rules on source usage. The statement will make the students aware of their responsibilities and the consequences of cheating. Missing statement does not release students from their responsibility.

1.	I/We hereby declare that my/our report is my/our own work and that I/We have not used any other sources or have received any other help than mentioned in the report.	☒
2.	I/we further declare that this report: - has not been used for another exam at another department/university/university college in Norway or abroad; - does not refer to the work of others without it being stated; - does not refer to own previous work without it being stated; - have all the references given in the literature list; - is not a copy, duplicate or copy of another's work or manuscript.	☒
3.	I/we am/are aware that violation of the above is regarded as cheating and may result in cancellation of exams and exclusion from universities and colleges in Norway, see Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§ 31.	☒
4.	I/we am/are aware that all submitted reports may be checked for plagiarism.	☒
5.	I/we am/are aware that the University of Agder will deal with all cases where there is suspicion of cheating according to the university's guidelines for dealing with cases of cheating.	☒
6.	I/we have incorporated the rules and guidelines in the use of sources and references on the library's web pages.	☒

Publishing Agreement

Authorization for electronic publishing of the report.

Author(s) have copyrights of the report. This means, among other things, the exclusive right to make the work available to the general public (Åndsverkloven. §2).

All theses that fulfill the criteria will be registered and published in Brage Aura and on UiA's web pages with author's approval.

Reports that are not public or are confidential will not be published.

I hereby give the University of Agder a free right to

make the task available for electronic publishing: ☒JA ☐NEI

Is the report confidential? ☐JA ☒NEI

(confidential agreement must be completed and signed by the Head of the Department)

- If yes:

Can the report be published when the confidentiality period is over? ☐JA ☐NEI

Is the task except for public disclosure? ☐JA ☒NEI

(contains confidential information. see Offl. §13/Fvl. §13)

Table of Contents

List of Figures

List of Tables

Notations

C	Electricity cost
cos θ	Angle of incident irradiation
E	Equation of time
E_{Gap}	Energy gap of material
F	Horizontal brightening
G_d	Diffuse irradiation
G_{NOCT}	Irradiation at NOCT
G_{ref}	Irradiation at STC
G_{SC}	Solar constant
G_T	Total incident irradiation
I	Generated current
I_0	Reverse saturation current
I_{mpp}	Current at maximum power point
I_{ph}	Photogenerated current
I_{sc}	Short circuit current
k	Boltzmann's constant
m	Ideality factor
N_C	Time Zone
N_{cs}	Number of cells
q	Electron charge
R_b	Beam radiation
R_s	Series resistance
R_{sh}	Shunt resistance
T	Ambient temperature
T_c	Cell temperature
t_c	Local time
t_s	Solar time
V	Terminal voltage
V_{mpp}	Voltage at maximum power point
V_{OC}	Open circuit voltage
β	Slope
γ	Azimuth angle

δ	Declination
θ	Angle of incidence
θz	Zenith angle
θz	Zenith angle
Φ	Latitude
ω	Hour angle
G_{on}	Corrected extraterrestrial irradiation
k_T	Clearness index

Abbreviations

AM Air mass

BESS Battery Electric Storage System

BMS Battery management system

CC Constant current

CV Constant voltage

DHI Diffuse irradiation

DNI Direct normal irradiation

DOD Depth of discharge

EMS Energy management system

EV Electric vehicle

GA Generic algorithm

GA Genetic algorithm

GHI Global horizontal irradiation

IEA International Energy Agency

IOT Internet of things

IV Current voltage

JSON JacaScript Object Notation

LP Linear Programming

MIP Mixed Integer Programming

MPP Maximum power point

MPPT Maximum power point tracking

MPPT Maximum power point tracking

NOCT Nominal operating temperature

PEV Plug in electric vehicle

PV Photo Voltaic

PV Photovoltaics

REG Renewable Energy Generation

SOC State of charge

SOH State of health

STC Standard test conditions

WHO World Health Organization

1. Introduction

Human activity has resulted in the release of large amounts of climate gasses mainly from the use of fossil sources in the last decades, resulting in a one degree increase in global temperatures compared to the average temperature from 1951-1980 [1]. Awareness to global warming has been increasing as its consequences have become more prominent through changes to the climate and negative impacts to human health. According to the World Health Organization (WHO) 250 000 additional deaths can be attributed to climate change [2]. This has resulted in international efforts to limit the climate change. According to the Paris agreement set into force November 4th, 2016, each of 196 participating countries agree to combat the global warming effects by providing long term solutions to decreasing emissions. The climate actions of each country vary in ambition with respect to how modernized the country is at the time. Results should be reported every 5 years and further actions should then be decided. For the Paris agreement in particular the overall goal is to limit the global warming to be a maximum of 2°C [3].

One of the main solutions to reduce the global emissions of climate gasses is the implementation of renewable energy sources. According to the International Energy Agency (IEA) the total global electricity production in 2018 resulted in 26 730 TWh. This is up 3.9% from the prior year and is steadily increasing. From this 6.9% was produced from wind and solar, 16.2% from hydro and 66.3% from combustible fuels. Energy produced with solar and wind energy increased by 24.3% and 12.4% respectively from 2017 [4]. The consumption of electricity on a global scale is also increasing steadily. In 2018 consumption increased by 4% from 2017 [4].

Electricity statistics for Norway is an exception from most of the world. In 2018 Norway produced around 150 TWh and consumed 136 TWh of electric energy [5][6]. Norway is a country with great renewable energy potential due to large amounts of rainfall and access to natural water reservoirs resulting in 90% of the electric generation coming from hydropower. In the last decade more renewable energy generation (REG) has been installed since the late 70’s and we see a shift from hydropower to wind power and private installations [5]. This is mainly because of more affordable prices which makes solutions such as solar and wind competitive solutions to conventional generation.

Renewable energy is becoming cheaper than ever, and solar power in particular is installed in much larger quantities than before, even replacing coal or oil powered plants in some places [7][8]. The main challenge when introducing renewable generation to the grid is the inconsistency of power generation. With energy sources such as solar power it is prone to have the greatest production at a time when energy demand is low. This problem is known as the duck curve and will become even more noticeable as more REG is installed [9]. If a system is producing more energy than what can be used the surplus energy will be lost without a means of storage. If a system has to reduce the energy output this is known as curtailment and should be avoided, if possible, especially when it comes to variable energy sources.

One of the solutions to this problem is to store excess energy for later use. This is rapidly becoming a large industry although deployment of such technology did see a decrease from 2018 to 2019 [10]. This could be because of lacking subsidies and the global situation regarding COVID-19. A report from MIT states that energy storage must cost $150/kWh or less to compete with conventional methods if 95% of the energy in the grid is to be supplied by a solar-wind mix [11]. As of today, a total of 159 GW of energy storage operating capacity is installed globally where 96% is pumped

hydro. When looking at the remaining 4% the main technologies are thermal, compressed air, flywheels and batteries with their shares being 38%, 11%, 15% and 32% respectively. When focusing on the batteries li-ion is the most used technology for energy storage with a market share of 83% [12]. The pricing for li-ion batteries are also decreasing and is down to 137$/kWh on average in 2020 which should be low enough for mass adoption [13].

Even though the price for li-ion storage has become somewhat reasonable it is still too expensive for most applications if the goal is to minimize cost. Another problem which must be addressed is the life cycle impact of batteries. Luckily, there might be a solution for this as 4.79 million electric vehicles (EV) entered the global market in 2019 continuing the upward trend [14]. Norway set a new record of registered EV's in 2020 and 54% of all the cars registered had electric drivetrains [15]. This means a rapidly growing market for used batteries in the coming years.

With smart energy management and repurposing of second use batteries it might be possible to reduce the storage cost and at the same time limit the environmental impact of battery electric storage systems (BESS).

As more effort is being put into creating smart energy management systems which include renewable energy generation and BESS, this is an area which is open for improvement and further study. By using open-source coding libraries and obtained theory from my years as a student in renewable energy this will therefore be the main scope of the thesis.

1.1 Problem definition

In this thesis a hybrid system with photovoltaic (PV) panels and BESS is designed, simulated, and optimized using both open-source coding and commercially available tools. Customized system design makes it possible to model different aspect with a larger degree of freedom to better suit the specific system and opens for more research potential. It also allows for day ahead and close to real time optimization when combined with APIs. The optimization is applying a linear programming (LP) approach using historical and forecasted input data obtained from measurements done at UiA, Grimstad campus. The thesis aims to assess different incentives with regards to battery energy scheduling to see how this impacts the electricity cost and battery behavior. Five different scenarios are simulated where the first four looks at energy optimization with historic data. For more practical use it is also done simulations with data for the upcoming day. This includes forecasted irradiation and temperature data as well as day ahead spot pricing for electricity.

With this in mind the main scope of the thesis is formulated:

- Establish the potential of using linear programming for battery energy scheduling optimization in a hybrid system.

There were also established subgoals which aims to further determine how a system should be designed to obtain the best results.

- Determine how a PV can system be modelled within Python to accurately predict power production.
- Determine how the results compare to commercially available tools (HOMER) to further evaluate the LP model.
- Determine how incentives such as throughput and state of charge impacts the total electricity price and battery behavior.

1.2 Limitations and assumptions

It is assumed that the system simulated in Python includes an inverter large enough for all power generated by the PV system to either be used locally or sold to the grid thereby leading to the system having no excess energy.

The historic irradiation data was given in minute intervals and the load data and electricity price data were given in hourly intervals. As a result, all the data was converted to hourly averages and as datapoints were missing or had excess data-points some form of data-manipulation had to be done to match the different datasets so that the code could run. The same manipulated data was used in both the Python program and HOMER to make the comparison accurate, but this process could impact the viability of the simulations.

As this thesis focuses on a real case study for a recycling facility in the making by Avfall Sør the load data used is collected from a similar site and assumed to be the same. This is also used as a basis for the day ahead load demand which could result in a large deviation from actual scenarios.

When calculating incident irradiation, the method used can cause deviations. Large errors have been corrected for in the code, but there might be some cases where this impacts the calculated power output.

The battery uses a simplified model which assumes that external operating conditions are optimal with regards to ambient temperatures.

If there are more assumptions and/or limitations this is mentioned in the methodology section.

2. Theory

2.1 Batteries

When implementing batteries as energy storage there are a several factors which should be considered such as: calendar ageing, cycle ageing, use in extreme temperatures or extreme charging or discharging scenarios. During its lifetime, a combination of these factors causes a reduction in capacity which makes it important to operate the battery in a way that minimizes straining factors when possible.

In this chapter a typical lithium-ion (Li-ion) battery is described together with some of the main aspects of importance when designing a battery energy scheduling algorithm.

2.1.1 Battery terminology

2.1.1.1 Battery capacity and power

One of the main aspects when choosing what battery to use is the needed capacity and power. The amount of power a battery can hold directly impacts how the system will function and what results can be expected. Capacity is usually measured in ampere-hours (Ah) and stored energy in kilowatt-hours (kWh).

To determine the capacity a common method is to first discharge the battery until it reaches its lowest voltage given by the manufacturer. The battery is then charged at a constant current (CC) for two hours until the maximum voltage is reached. It is then charged at constant voltage (CV) until the current is reduced to 5-10% of the original value. This is known as CC/CV charging. The capacity is then calculated through integration of the current over the duration of discharge. This process is repeated until the capacity stabilizes [16, p. 40]. In figure 1 typical CC/CV charging is shown.

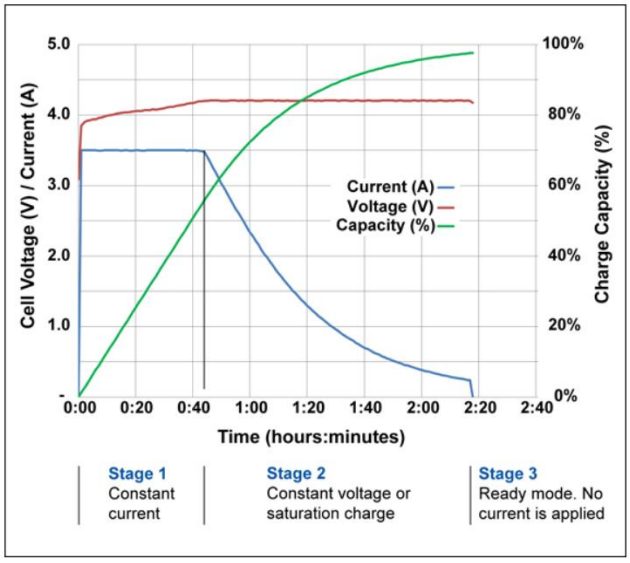


Figure 1: CCCV charging [17]

Another important factor is the charging and discharging power limits. For a battery to work well in a given system it is important that sufficient power can be delivered when needed. The amount of current delivered to or from the battery is called *C-rate*. 1C is the current at which the battery will completely empty its entire capacity in 1 hour. For a 12 Ah battery this means a current of 12A [16, p. 40]. At 0.5C the battery would empty in 2 hours with a current of 6A, and at 2C the battery would empty in 0.5 hours with a current of 24A. It is important to keep the battery within its specified voltage limits to not inflict damage to the components.

2.1.1.2 State of health (SOH)

During the lifetime of a battery or when assessing whether a second use battery can be used a crucial measurement is its state of health (SOH). This is a value which describes how much of the intended capacity is currently remaining. To obtain an accurate estimation there are many methods that can be used. Some are mainly done in labs as they need specific and complex tools. These methods are mainly used when a status report is needed prior to use or for routine checkups.

One such method is measuring the internal resistance of the battery as this is a parameter which strongly correlates with ageing and degrading of components. It is most common to measure the inner resistance as a function of voltage drop in the battery relative to its open circuit voltage (OCV). The tests are made with varying pulses of current and offer good results. The main drawback is that it needs a long time for the battery to reach equilibrium so that OCV can be obtained [18].

Another similar method is the measurement of impedance in the battery. This value is also strongly related to battery ageing. The most common way of measuring impedance is with electrochemical impedance spectroscopy (EIS). This method requires very stable conditions but offer more detailed information as to where the majority of energy loss occurs within the battery [18].

When SOH estimations are needed without specialized equipment, or when disassembly of the battery pack is not possible, modelling methods and/or estimations based on historic data can be a solution. If numbers of cycles and extreme scenarios with regards to power output/input or temperature conditions is recorded during its lifetime an estimate of remaining capacity can be made. Accurate estimates still need reference data from tests done in a lab of a new battery from new to end of life (EOL) [18].

2.1.1.3 State of charge (SOC)

For optimal system design it is important to have a precise reading on the available energy at any given time. This is known as the state of charge (SOC) and can be obtained in different ways. The simplest methods work by directly comparing the current voltage to the OCV. For lead acid batteries this relationship is approximately linear, but this is not the case for lithium-ion batteries [19].

As with SOH estimations it is possible to determine SOC with impedance measurements although this is not very suitable for real time readings.

Because of this the most common SOC estimation when using Li-Ion batteries are book keeping methods [19]. This is a way of estimating the remaining capacity based on real time current or power flow to or from the battery. By knowing the SOC at the previous timestep, $SOC(t-1)$, the momentary current, $I(t)$, and nominal capacity, Q_n, it is possible to determine the SOC at the current timestep, $SOC(t)$. This is generally known as coulomb counting.

$$SOC(t) = SOC(t-1) + \frac{I(t)}{Q_n} \qquad (2.1)$$

To account for loss factors such as temperature, historic use, or cycle life it is possible to obtain more accurate current estimations through experimental data for a given battery pack. This will improve the accuracy of SOC estimation.

Other methods such as neural networks, Kalman filters and fuzzy logic are also studied and used for these kind of battery estimations, and sometimes a combinations of several methods are applied [19].

2.1.2 Lithium-ion battery chemistry

A typical lithium-ion battery consists of a copper and aluminum electrode, an anode, a cathode, and a separator. The most used material for the anode is graphite as this offers a good balance between energy density when storing Li^+ ions as well as providing good conductivity and cycle efficiency. The cathode is usually a metal oxide which include lithium ions. One common material is Lithium Cobalt Oxide ($LiCoO_2$). To keep the two electrodes separated a material which only allows Li^+ ions to pass through while restricting electron flow. This makes it so that electrons are forced to move in an outer circuit where energy can be extracted or added through charging or discharging. The separator is made up of polymer foils to keep the structure as compact as possible and works to keep the two electrodes separate while allowing ion migration. Finally, the electrolyte consists of a material where ions can move freely. In the case of Li-ion batteries this is typically a conducting salt ($LiPF_6$) [20, p. 250]. This construction and the flow of ions during charging and discharging is shown in figure 2.

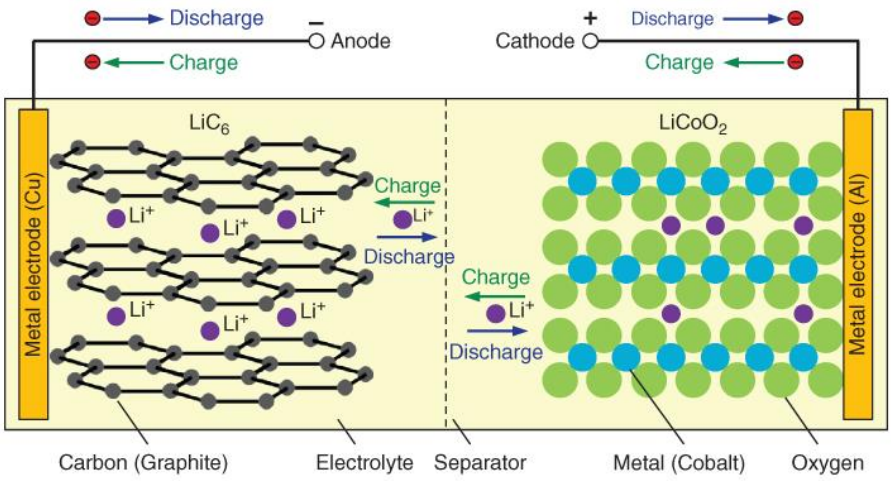


Figure 2: Construction and chemical reactions in a lithium-ion battery [20, p. 250]

When charge and discharge is left and right direction respectively the overall reaction becomes:

$$LiC_6 + 2Li_{0.5}CoO_2 \rightleftarrows C_6 + 2LiCoO_2 \qquad (2.2)$$

Another important aspect of Li-ion battery design is a property which was discovered by accident but prove to be crucial to a working design. This is known as the solid electrolyte interface (SEI) and is a layer consisting of a substance created from a reaction between the electrolyte and free Li^+ ions on the anode side during the first cycles. This layer acts as a barrier hinders the electrons from coming in contact with the electrolyte thereby limiting reduction of the active Li-ion material. By changing the way SEI is formed it is possible to improve the efficiency of the battery [21].

2.1.3 Battery life

There are many aspects that impact battery life. The main cause of capacity reduction is film growth on the SEI during cycling. While being an important reason as to why li-ion batteries can function at all continuous growth of this layer will cause a loss of active lithium and increase the inner resistance of the battery both of which reduces the capacity. High temperatures and SOC is especially impactful on SEI growth. Reduction of active material on the electrodes can also be a major contributor to reduced battery capacity. This is usually the result of high C-rates and depth of discharge (DOD) [16, p. 81].

Losses can be divided into two categories: cycle losses and calendar losses. Cycle losses occur during charging and discharging, and calendar ageing happens at times where the battery is unused. The main driving factors of cycle ageing is C-rate, DOD and in what range the battery is being used. Temperature also plays an important part. Calendar ageing is mainly impacted by temperature and SOC during the time of no use [16, p. 83].

The manufacturers usually state the estimated battery life in years and number of complete cycles at a given degree of discharge. This can be found in the battery datasheet. For lithium ion batteries this can for example be 20 years, 5000 cycles and a discharge depth of 90% [20, p. 212].

2.1.4 Equivalent circuit

When modelling batteries, it is common to use either electrochemical or electrical approaches. Using equivalent circuit with varying complexity offers a wide range of simulation and modelling options and tend to be the choice for electrical engineers. When using an equivalent circuit approach, it is common to use a combination of simple components to mimic the actual behavior of the battery. In this example a voltage source V_o represents the open circuit voltage at any point in time based on the voltage difference between the cathode and anode. A resistance, R_s, represents all resistances and ohmic impedance to the flow of current in the cell. Another resistance, R_{ct}, and a capacitor, C_{dl}, are then implemented in parallel as a branch to account for resistance during charge transfer and buildup of charges at the interfacial double layer, respectively. The values for these components can be decided through experimental tests and/or historic data [16, p. 29]. An equivalent circuit is shown in figure 3.

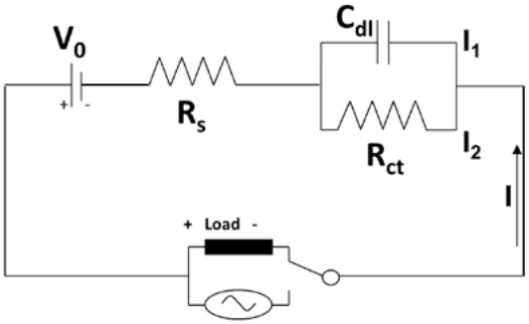


Figure 3: Equivalent circuit of battery which can be adapted for different technologies [16, p. 29].

When the correct values for the components are decided it is possible to simulate battery behavior by applying Kirchhoff's equations for currents in a circuit.

2.1.5 Battery management systems

To maximize battery life and ensure safe operation a battery management system (BMS) is designed and used to manage the battery and each of its cells. A BMS is especially important for Li-ion batteries as they are a lot less prone to damage when they are misused. The cells are also very sensitive to over- and undercharge. The main tasks of a BMS are: monitoring and estimation of battery status, protecting battery from unsafe conditions and maximize its performance [22, p. 15].

It is important that the BMS can prevent high temperatures, overcharge, overdischarge and too high charge- or discharge currents. This is done through requesting changes in the system such as varying the degree of cooling or by switching certain parts on or off directly.

The most important feature of a BMS is to control and manage the voltage of the battery and each cell in it. Whenever a cell reaches its maximum voltage, the charging must be turned off. When this happens, it does not necessarily mean that each other cell is fully charged. This leads to an unbalanced battery and another important job of the BMS, cell balancing. To maximize the capacity of the battery and to reduce the strain on specific cells the BMS should be able to either distribute the charge across the cells or use a charge/discharge pattern which repeats until all cells are balanced.

Cell balancing is also beneficial when a battery is stored as calendar ageing will be unbalanced across the cells [22, p. 24].

2.1.6 Dimensioning calculations

When deciding what battery best fit the system in question there are a couple of things to keep in mind. How long the system should be self-sufficient with battery power and the amount of power that is needed. When combining batteries with a PV system or other renewable sources it can also be beneficial to know how large the battery must be to reach predetermined goals with regards to self-consumption and renewable percentage.

To calculate the needed capacity based on a given energy demand is rather simple. If the energy demand, E_d, round trip efficiency, η, and duration, N_A, is known it is possible to calculate the necessary battery capacity, E_b.

$$E_b = \frac{E_d * N_A}{V_{oc} * \eta} \quad (2.3)$$

If it is of importance to know how long a battery can sustain a given power output, T_b, this can be done by knowing the needed discharge current, I_{dc}, the stored energy of the battery, E_b, and voltage, V_b.

$$T_b = \frac{E_b}{I_{dc} * V_b} \quad (2.4)$$

The other aspects mentioned are dependent on produced renewable energy and a question with regards to system cost becomes relevant. These calculations are therefore more complex and is better done with simulation tools.

2.2 PV systems

PV systems are dependent on many factors with regards to modelling and system design. This chapter will go over the most important aspects.

2.2.1 Module parameters

A typical PV module datasheet given by the manufacturer is shown in appendix. Here different IV curves at varying irradiation are shown as well as the most important values at STC.

I_{SC} - The maximum current when the circuit is short circuited, and the voltage is zero.

V_{OC} - The maximum voltage when the circuit is open, and the current is zero.

I_{MPP} – The current at the point of maximum power generation.

V_{MPP} – The voltage at the point of maximum power generation.

P_{max} – The maximum amount of power generated by the module, also known as the designed capacity.

Efficiency – How much of the available light energy can be converted to usable power.

Temperature coefficients – How the current, voltage and power generation is impacted by varying temperatures.

2.2.2 Fundamentals and irradiation

2.2.2.1 PV fundamentals

The main goal of any PV system is to generate as much energy as possible from the global irradiation at any given time. The amount of electric energy generated by the PV module compared to the total available irradiation energy decides the efficiency. In the last decades, many different technologies have been developed, tested, and improved. Efficiencies for commercial silicon cells have improved from 13% to 26% since 1977 and experimental technology using concentrators have reached 47% [23, p. 15]. These values have been obtained in labs at standard test conditions (STC) meaning an irradiation of

1000 W/m^2, a cell temperature of 25°C, and an air mass (AM) of 1.5. Air mass is the distance the light must travel through the atmosphere to reach the panel surface. An AM of 1 is when the sun is directly above the panel and AM0 is the irradiation outside the atmosphere known as the solar constant, G_{SC}, with a value of 1367 W/m^2 [20, p. 25]. AM depending on solar angle is shown in figure 4.

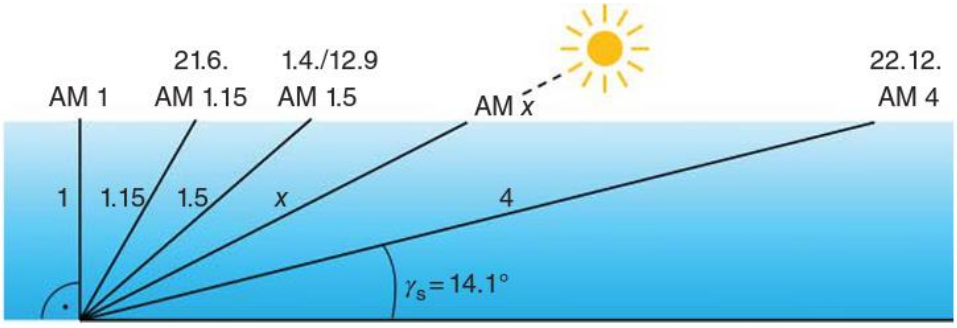


Figure 4: Air mass at different angles of the sun relative to the PV location [20, p. 25]

2.2.2.2 Incident irradiation

There are four main components discussed when incident irradiation is calculated [20, p. 34]:

Global horizontal irradiation (GHI) is the most common measurement of solar irradiation and can be decomposed to find the other components through trigonometry. This is the amount of solar irradiation that hits a surface that is horizontal to the earth.

Direct/beam/normal irradiation (DNI) is the irradiation that travels in a straight line from the sun onto a surface that is placed normal to the path of the sun.

Diffuse irradiation (DHI) is the irradiation that finds its way to the panel not in a straight line but from being scattered and reflected from particles in the atmosphere. This value is often assumed to be equally divided across the whole sky (isotropic assumption).

Reflected irradiation (Albedo) is the irradiation that is directed from the ground through reflection. This value is especially important to include when a panel has a slope. Reflectance factor of many different materials have been obtained [24].

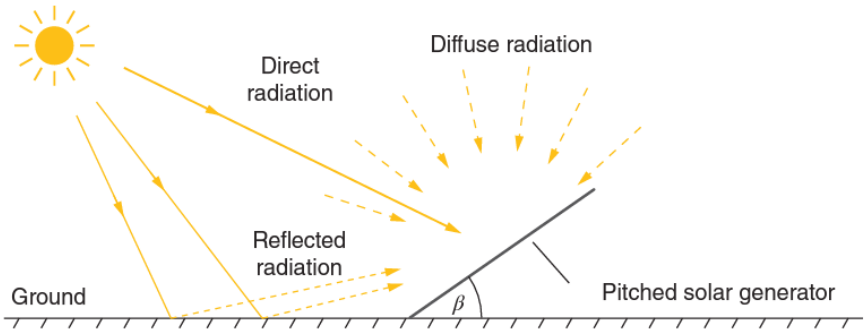


Figure 5: The components that build up incident irradiation [20, p. 35]

To calculate the expected power output at any given moment in time with a model of a PV module the primary and most important part is knowing the irradiation incident on the panel throughout the day. Simplified calculations can be done based on location and estimated annual power production but accurate representations dependent on several factors [25]:

Φ **Latitude,** is the angular location north or south of the equator (-90° to 90°).

δ **Declination,** is the angular position of the sun at solar noon with respect to the equator. This value ranges from -23.45° to 23.45° where the latter is the northern hemisphere. 23.45° is the tilt of the earth.

β **Slope,** is the angle of the surface in question relative to the ground. This can vary from 0° to 180°, but values greater than 90° means that the front of the panel is facing towards the ground.

γ **Azimuth angle,** is the orientation (north, south, east, west) of the panel. The azimuth ranges from -180° to 180° where 0° is south, east being negative and west positive.

ω **Hour angle,** is the angular displacement happening as a result of the constant rotation of the earth.

θ **Angle of incidence,** The angle between the beam radiation on the surface and the normal to that surface.

θ_z **Zenith angle,** The angle between the vertical and beam radiation.

The declination angle, δ, can be calculated as a function of the tilt of the earth (23.45°) and the day number, n, where Jan. 1st is day number 1, Jan 2nd is day number 2 etc.

$$\delta = 23.45 * \sin\left(360 * \frac{284+n}{365}\right) \quad (2.5)$$

The hour angle, ω, can be calculated as a function of the movement of the sun across the sky which is 15°/hr. This calculation is also using the solar time, t_s, as a reference. This is important as at any given location the sun reaches its highest point not at 12 local time.

$$\omega = (t_s - 12hr) * \frac{15^\circ}{hr} \quad (2.6)$$

In the previous equation the solar time, t_s, can be calculated by correcting the local time, t_c, for the time zone, Z_c, the longitude of the location, λ, and a value, E, which represents the equation of time.

$$t_s = t_c + \frac{\lambda}{\frac{15^\circ}{hr}} - Z_c + E \quad (2.7)$$

The equation of time is in place to correct for obliquity and eccentricity of earth's orbit. This is in other words to correct for the tilt of the earth relative to the plane of the earth around the sun and to account for the fact that earth's orbit is not perfectly circular. In this case the equation of time is correcting in number of hours and not minutes as given in [25, p. 11]

$$E = 3.82(0.00075 + 0.001868 * \cos B - 0.032077 * \sin B - 0.014615 * \cos 2B - 0.04089 * \sin 2B) \quad (2.8)$$

Where the variable *B* is given by:

$$B = 360° \frac{n-1}{365} \quad (2.9)$$

When all of these values are obtained it is now possible to calculate the angle of incident irradiation, *cosθ,* for any plane:

$$\cos\theta = \sin\delta \sin\varphi \cos\beta - \sin\delta \cos\varphi \sin\beta \cos\gamma + \cos\delta \cos\varphi \cos\beta \cos\omega + \cos\delta \sin\varphi \sin\beta \cos\gamma \cos\omega + \cos\delta \sin\beta \sin\gamma \sin\omega \quad (2.10)$$

When using this equation care should be taken for hour angles within sunrise and sunset as this can cause false negative or positive spikes. This again can lead to large spikes in the calculated irradiation and should be corrected for.

The zenith angle, θ_z, is calculated by inserting $\beta = 0°$ in the above equation which leads to:

$$\cos\theta z = \cos\varphi \cos\delta \cos\omega + \sin\varphi \sin\delta \quad (2.11)$$

As mentioned earlier the solar constant, G_{SC}, is the amount of solar irradiation hitting the outside of the atmosphere. This however is not true as the earths eccentric orbit impacts this value. The corrected extraterrestrial irradiation then becomes:

$$G_{on} = \mathrm{Gsc}(1 + 0.033 \cos\frac{360}{n}) \quad (2.12)$$

The horizontal value of the above, G_{on}, then becomes:

$$G_o = G_{on} * \cos\theta z \quad (2.13)$$

To obtain the extraterrestrial irradiation averaged over each timestep the above equation is then integrated over a single time step ($\omega_2 - \omega_1$).

$$\bar{G}_0 = \frac{12}{\pi} G_{on}[cos\varphi * cos\delta * (sin\omega_2 - sin\omega_1) + \frac{\pi(\omega_2 - \omega_1)}{180°} sin\varphi sin\delta] \quad (2.14)$$

To accurately calculate the total amount of incident irradiation on a plane it is important to include both diffuse, G_d, and direct/beam irradiation, G_b. In some cases, both of these values are measured but if this is not the case and only global horizontal irradiation averaged over a time period, G_{avg}, is known these values must be derived from the latter.

This is done by knowing the clearness index, k_T, and the relationship between G_d and G_{avg}. Depending on the value of k_T the ratio between G_d and G_{avg} is calculated as follows [26]:

$$k_T = \frac{G_{avg}}{\bar{G}_0} \quad (2.15)$$

$$\frac{G_d}{G_{avg}} = 1.0 - 0.09 * k_T \text{ for } k_T <= 0.22$$

$$\frac{G_d}{G_{avg}} = 0.9511 - 0.1604 * k_T + 4.388 * k_T^2 - 16.638 * k_T^3 + 12.336 * k_T^4 \text{ for } 0.22 < k_T <= 0.8$$

$$\frac{G_d}{G_{avg}} = 0.165 \text{ for } k_T > 0.8 \quad (2.16)$$

As G_{avg} equals the sum of both G_d and G_b these relationships can now be used to calculate either of the two components.

Three additional values are needed before calculating the final incident irradiation. The beam ratio, R_b, the anisotropy index, A_i, and horizon brightening, f. R_b is defined as the ratio between $cos\theta$ and $cos\theta_z$. A_i is also known as scattered irradiation and is defined as the ratio between G_b and $\bar{G}_0$. Finally, horizon brightening, f, is implemented to account for the fact that most diffuse irradiation comes from the horizon and not the sky.

$$f = \sqrt{\frac{G_b}{G_{avg}}} \quad (2.17)$$

By combining all of these variables as well as a factor for albedo or ground reflectance, ρ_g, the total incident irradiation, G_T, on a plane with slope, β, is calculated as follows (yellow=direct, green=diffuse, blue=reflected) [27]:

$$\bar{G}_T = (G_b + G_d * A_i)R_b + G_d(1 - A_i) * \left(\frac{1+cos\beta}{2}\right)\left[1 + fsin^3\left(\frac{\beta}{2}\right)\right] + G_{avg} * \rho_g\left(\frac{1-cos\beta}{2}\right) \quad (2.18)$$

2.2.3 Power generation

2.2.3.1 PV modeling and single diode model

When the incident irradiation is known a simplified estimation of power output can be calculated with the efficiency of the module and corrections with respect to the cell temperature. A more accurate prediction can be done through equivalent circuit modeling as this can include parameters that are specific to the panel used which impacts performance.

It is possible to make quite complex models, but a simplified version known as the "single diode equivalent circuit" is shown in figure 6.

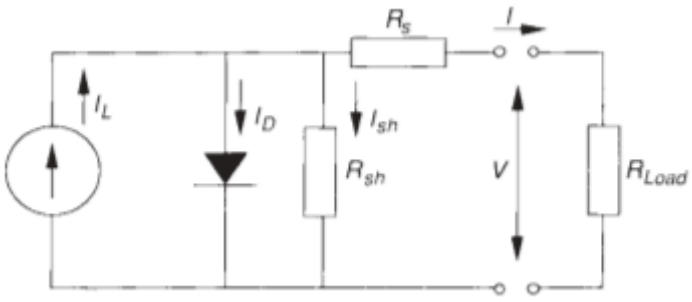


Figure 6: Single diode equivalent circuit [25, p. 750]

This model works for all Si-crystalline modules and offers sufficient accuracy when compared to more complex models such as the "two diode model". It is also worth noting that for calculations based on PV parameters given by the manufacturer the lowered accuracy of the model is negligible as the main limiting factor is the basic information given [25, p. 750].

From this equivalent circuit we can derive the total single diode equation for a module with a defined number of cells, N_{CS}, which gives:

$$I = I_{ph} - I_0\left[e^{q*\frac{V+I*R_S}{N_{cs}*m*k*T_c}} - 1\right] - \left(\frac{V+I*R_s}{R_{sh}}\right) \quad (2.19)$$

Where:

V = Terminal voltage, here V_{mpp}

I_{ph} = photogenerated current

I_0 = diode reverse saturation current

R_s = Series resistance, ohmic losses in front contacts

R_{sh} = Shunt resistance, leak currents at the edges of the solar cell

I = Generated current

q = electron charge = 1.602 E-19 coulomb

k = Boltzmann's constant = 1.381 E-23 J/K

N_{cs} = Number of cells in series

m = ideality factor (How close the cell is to ideal, usually between 1 and 2 for Si)

T_c = Cell temperature

The cell temperature impacts the efficiency of the cell and can be derived from knowing the ambient temperature, T, incident irradiation, $\bar{G}_T$, and nominal operating temperature (NOCT) of the given cell. The latter is defined as the working cell temperature when the irradiation is equal to 800 W/m^2, G_{NOCT}, the wind speed is 1 m/s and ambient temperature is 20°C.

The cell temperature at different ambient temperatures is then calculated with the following equation [20, p. 148]:

$$T_c = T + (NOCT - 20°C) * \frac{\bar{G}_T}{G_{NOCT}} \qquad (2.20)$$

Equation 2.20 assumes that the temperature of the cell is directly proportional to irradiation and does not correct for the amount of load as is done in eq. 23.3.3 in [25, p. 758]. For the estimation to be correct it is essential that the panel in question is mounted in the same manner as when it was tested to obtain NOCT [25, p. 758].

The current generated by the cell occurs on both sides of equation 2.19 creating an implicit function that needs to be solved numerically.

R_s and R_{sh} can be estimated for a specific PV module by applying some simple calculations with regards to its IV-curve [20, p. 89]. This method is shown in figure 7. R_s should be low and is usually around 0.5 Ω. R_{sh} should be high and is ideally 1000 Ω.

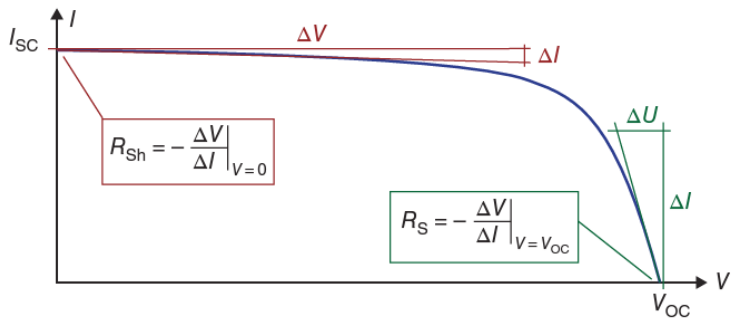


Figure 7: Calculating the series and shunt resistance from the module IV curve [20, p. 89]

This version of the diode model assumes a perfect proportional relationship between the irradiation and photogenerated current. To correct for the effects of cell temperature the following equations are used for the photogenerated current and diode reverse saturation current:

$$I_{ph} = \left(\frac{\bar{G}_T}{G_{ref}}\right) * [I_{ph\,ref} + T_{coeff} * (T_c - T_{cref})] \quad (2.21)$$

$$I_0 = I_{0\,ref} \left(\frac{T_c}{T_{cref}}\right)^3 * e^{\left(\frac{q*E_{Gap}}{m*k}\right)*\left(\frac{1}{T_{cref}} - \frac{1}{T_c}\right)} \quad (2.22)$$

Where:

G_{ref} = 1000 [W/m^2]

$I_{ph\,ref}$ = I_{sc} [A]

T_{coeff} = Temperature coefficient of short circuit current [A/°C]

T_c = Cell temperature [°K]

T_{cref} = Reference temperature = 298.15[°L]

E_{Gap} = Energy gap of material = 1.12 eV for Si-cristalline

In equations 2.19 there are five parameters (R_{sh}, I_{ph}, I_0, R_s, m) which have to be determined. They can be estimated, assumed or solved for with a base in guessed values as has been done in [25, p. 754]. For a set of parameters in any given scenario the power output of a module can then be found through P=V*I where V varies with temperature and irradiation as per the manufacturer descriptions.

2.2.3.2 IV curves

When the single diode model has been obtained it is possible to solve for either the terminal voltage or generated current at different points when varying the other. This creates what is known as the IV curve and visualizes the properties for a given module. When varying the irradiance or temperature it is also made clear how this impacts performance as can be seen in figure 8 and 9.

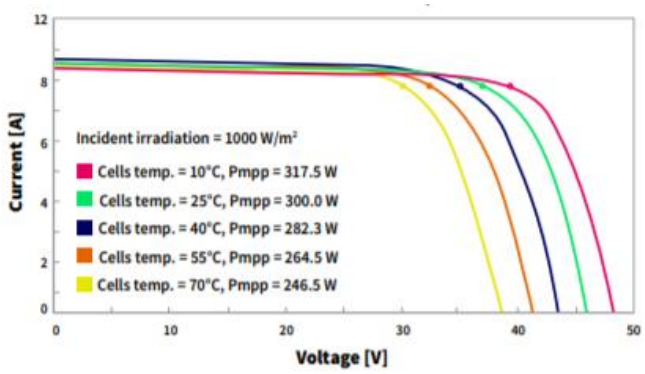


Figure 8: Temperature dependence of solar cell, circles indicate MPP [28]

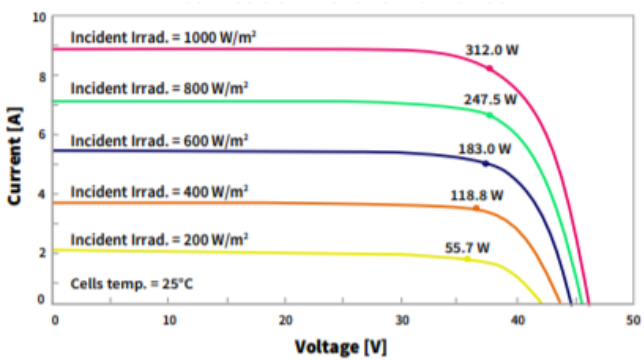


Figure 9: How a change in irradiance impacts the IV curve of a solar module, circles mark MPP [28]

The IV curve generated from the diode model can also be used to determine the maximum power point (MPP) where the product of I and V is greatest. These values are known as I_{mp} and V_{mp} and shown in figure 10.

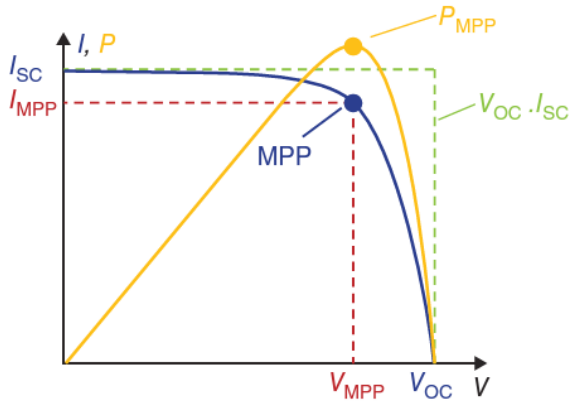


Figure 10: IV and power curve with MPP [20, p. 83]

2.2.3.3 MPPT

By implementing an algorithm and a DC/DC converter the PV system can run at optimal voltage and thereby also power. This is known as maximum power point tracking (MPPT) and commonly doubles up as a charge controller if the PV system and battery is DC coupled as discussed further in 2.2.4.5. This allows the PV system to run at optimal voltage while at the same time outputting the best current and voltage to match the current state of the battery [29]. A common algorithm is the *perturb and observe* method which determines the power output and then increases the duty factor. This increases the voltage of the system and if the power increases this process is repeated until the opposite is true as shown in figure 11. This means that the actual operating power point will vary slightly from actual MPP, but this is almost negligible for a good quality controller [20, p. 171].

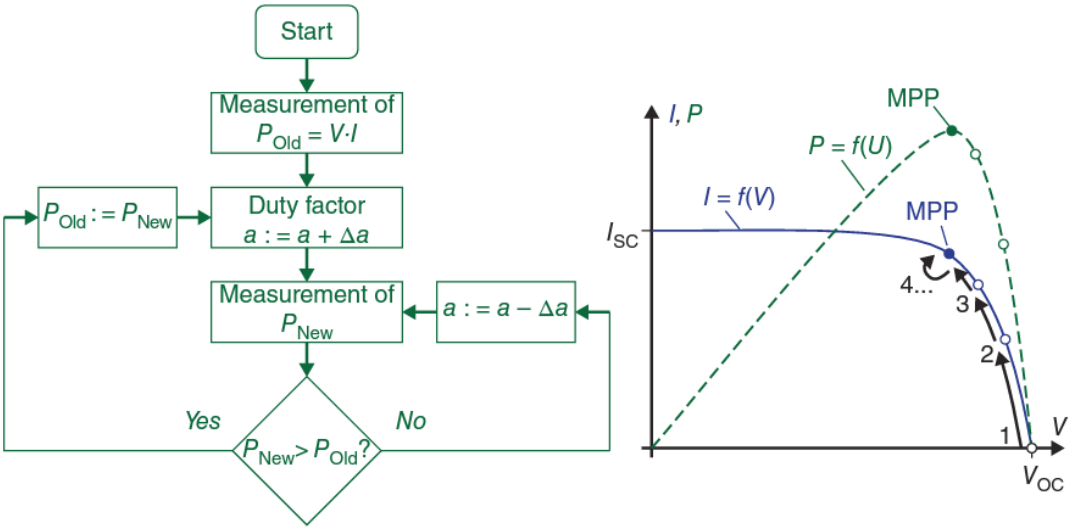


Figure 11: MPPT using peturb and observe method

2.2.4 Losses and system design

Solar panels offer good efficiencies and usually have pretty good certainty in power generation if the weather conditions are known through forecasts. There are, however, some loss factors when operating PV systems that can result in unplanned power reduction.

2.2.4.1 Shading

One of the most impactful loss factors is due to shading of the cells. If the cells are connected in parallel a single shaded cell will reduce the total current output by the value of the current passing through that particular diode. If the cells are connected in series a single shaded cell will result in it having a negative voltage from the current trying to pass through it from the previous illuminated cells, thereby reducing the current drastically. This is a recognized problem as solar modules are mainly constructed by having cells in series connection to increase the voltage while keeping the current low. To solve this bypass diodes are implemented in different ways to allow current to circumvent shaded cells and thereby reducing losses [20, p. 140]. The effect of implementing different amounts of bypass diodes is shown in figure 12.

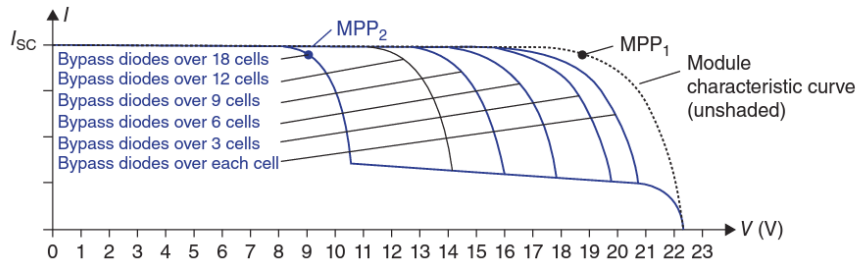


Figure 12: The effect on IV curve of having different amounts of bypass diodes installed as cells become shaded [20, p. 144]

There are many causes of shading including constructions, trees, and foliage. To address shading when doing power generation estimates or smart module design, shading profiles are created which describes how obstructions will impact direct irradiation throughout the day at different times of the year as shown in figure 13. This can be done with tools of varying degree of complexity and automation such as: manually operated sun path indicators or automatic tools such as SunEye [20, p. 267]. It is important to install the panels in a way that does not cause one panel to cast shade on another. This is avoided by calculating the distance of the shade casted during the shortest day of the year [20, p. 268].

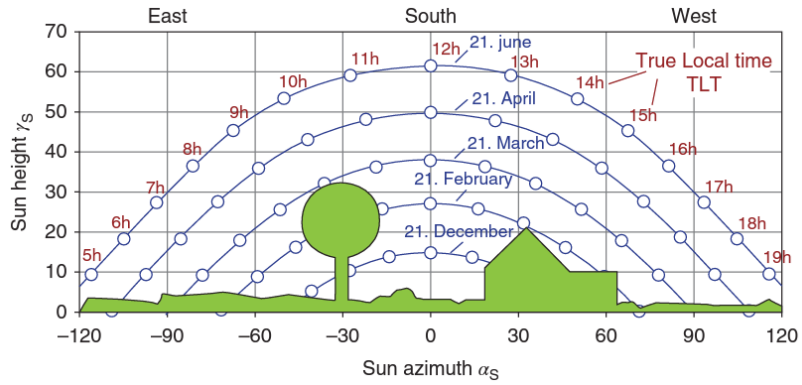


Figure 13: Sun path diagram with shading horizon [20, p. 267]

2.2.4.2 Mismatching losses

Modules with different specifications or illuminating conditions connected in strings can cause the same loss in current as described when designing modules from a number of cells. Therefore, it is common to divide multiple strings of similar properties and orientation in parallel connections.

This leads to another problem as connecting multiple strings to the same DC bus in parallel can cause one or more strings to greatly limit the total output voltage of the PV system if the strings are of different ratings or operating under different conditions. The voltage for all cells will then be equal to the lowest value, which causes a large loss of power for the panels with a higher rated V_{mpp} than what is obtained [20, p. 153]. These types of losses are known as mismatching losses and are avoided through smart system design as discussed in 2.4.4.4.

2.2.4.3 Other losses

Other impactful loss factors include soiling from things like bird dropping, dust and snow coverage. These losses can typically reach 10% or greater if the panel is not installed on a steeper angle (30°+). As mentioned earlier high temperatures can cause a loss in power but is usually accounted for in PV modeling [20, p. 265].

System components such as inverters and power lines on the DC and AC side also contribute to losses. Modern inverters should have an average efficiency greater than 95% and line losses should be less than 1% [20]. The efficiency of inverters changes depending on the working load and PV generator voltages as shown in figure 14 where P_{DC} denotes the output power and P_{DC_N} denotes the DC nominal power at the inverter input.

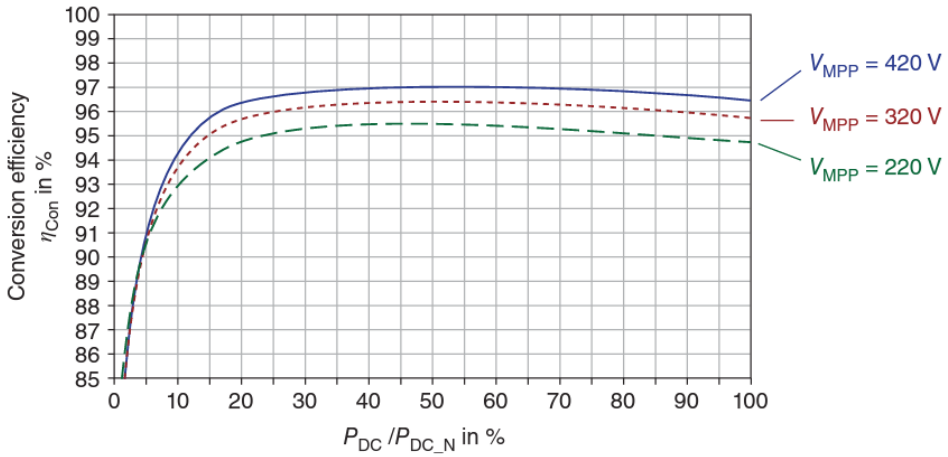


Figure 14: Conversion efficiency (%) of an inverter at different loads (%) and PV voltages (V) [20, p. 184]

As mentioned in 2.2.2.1 the cell temperature also impacts the performance of the cells where temperatures exceeding 25°C causes a loss in power generation of approx. 0.3-0.45% per degree Celsius. On the contrary, lower temperatures increase the performance [20, p. 148]. This is usually accounted for in the PV modelling and therefore excluded from the combined system losses.

A loss factor can be included when calculating the total energy generation by multiplying the generated power with a constant. A loss factor of 0.8 would imply that 80% of the generated energy is converted to usable energy or in other words a loss of 20% due to outside factors. This works well for simulations over longer periods where average losses can be estimated through advanced models but is less suitable for day ahead simulations unless the degree of soiling or shadowing is known for the upcoming time period [30].

2.2.4.4 Reducing losses with smart system design

To reduce system losses different methods with regards to system design have been developed. One way of reducing shading losses is to implement bypass diodes as discussed in 2.2.3.1. Another method involves smart implementation of inverters and can be used in combination with bypass diodes or as the primary approach. By implementing inverters for each string, or group of strings with the same structure and a high certainty of no shading, it is possible to individually control the MPP of each string or group to reduce mismatching losses. It is also possible to include inverters for each module, but this is usually deemed to complex, expensive and prone to wear and tear [20, p. 173].

A single central inverter and series inverter system is shown in figure 15.

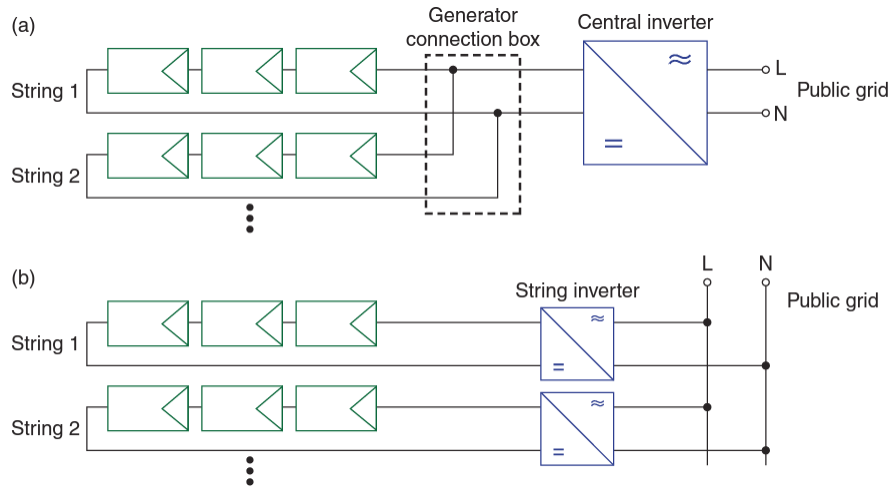


Figure 15: a) Central inverter design b) Series inverter design

String or module inverters benefit from the simplification or complete removal of direct current cabling which can further improve system efficiency [20, p. 175]. It is also less prone to complete system failure if one of the inverters stops working.

Another way of reducing mismatching losses is to include a multi-input inverter that contains a MPPT for each separate string as seen in figure 16. This is especially useful in scenarios where the strings experience different amount of shading or irradiation due to orientation. This solution makes it possible for each string to be controlled and optimized individually while maintaining the use of a shared direct voltage bus [20, p. 178].

A design which aims to maximize inverter efficiency is the Master-Slave concept. This design includes multiple inverters so that on low power production only one of the inverters, the master, is used to keep it working at a higher load factor. When the system power increases beyond the limits of the single inverter another inverter shares the load and so on. To ensure that each inverter works approximately the same amount the master can be changed each day [20, p. 178]. These concepts are shown in figure 17.

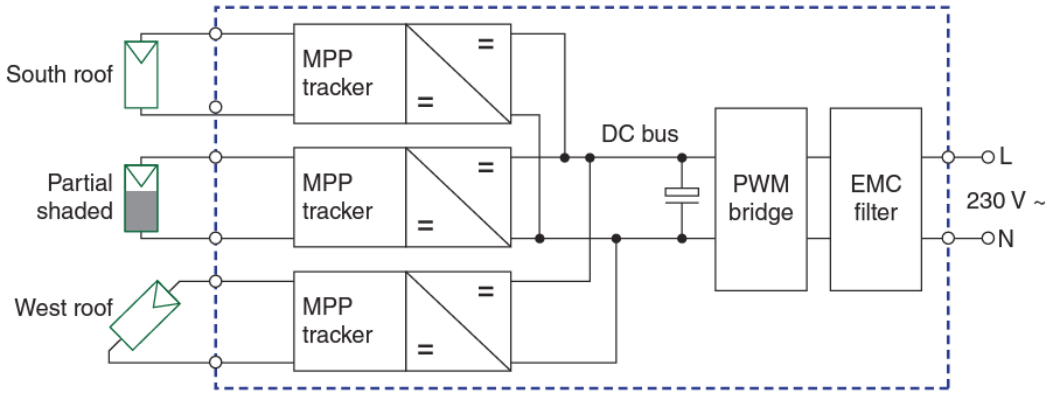


Figure 16: Multistring inverter design where the blue box contains all the components of the inverter [20, p. 178]

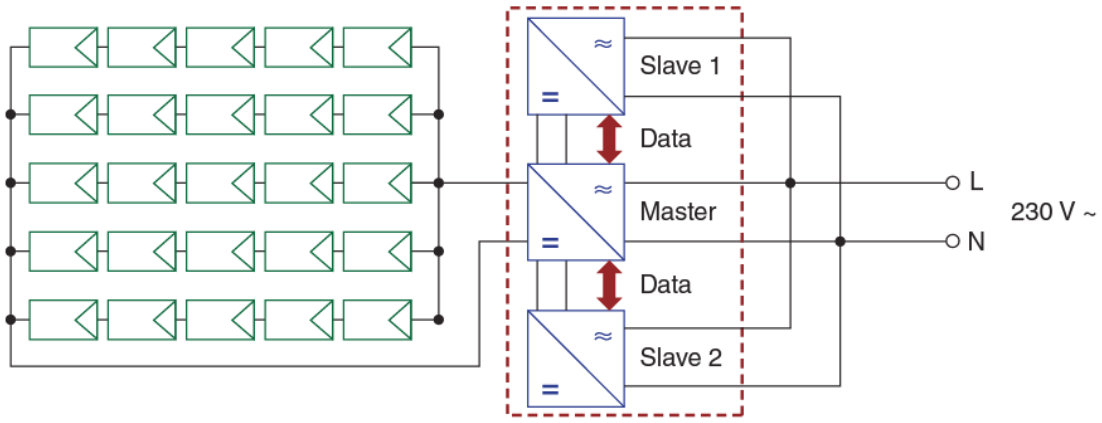


Figure 17: Master-Slave inverter design [20, p. 178]

2.2.4.5 Charge controllers

The previous chapter discussed different system designs where the panels are connected directly to the grid. If the system also includes a battery this leads to additional challenges as the battery must be charged at correct voltages. When the battery is DC coupled this means that the PV system voltage is restricted to the battery voltage causing mismatching losses as discussed in 2.2.4.2. This may greatly impact the total power production of the PV system and should be avoided. This can be achieved by connecting the battery to the AC bus with the help of a charge controller and AC/DC inverter. Otherwise, when if battery is connected to the DC-bus, the implementation of a charge controller and one or more DC/DC converters between the PV system and battery solves any issues [20, p. 197]. DC and AC coupled battery designs is shown in figure 18. By applying the same logic as used in a multistring inverter design it is possible to connect each string to its own combined MPPT charge controller if the battery is DC coupled to ensure optimal operation. There also exist solutions that combine all aspects including the inverter [31].

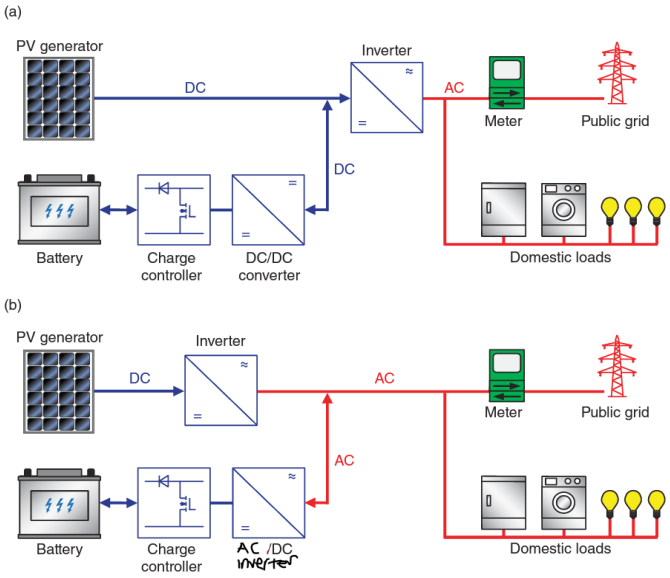


Figure 18: a) DC coupled PV battery system b) AC coupled PV battery system [20, p. 198]

2.3 Electricity pricing

Electricity price can be calculated different ways depending on the power plan used. For a typical household, the price can either be based on the spot price or be set to a fixed amount per kWh consumed. For larger industries it is common to pay for the amount of energy used as well as a fixed price for the peak power consumption each month that usually varies with the seasons. The spot price fluctuates each day and each month depending on electricity production and demand.

Norway is part of the European electricity market where power is traded at the Nord Pool Day ahead market. This means that electricity produced in either of the member countries is sold across borders for a set price decided for the upcoming day. When for example Norway experiences a large hydrological surplus, or Denmark receives much wind, it is economically beneficial to sell some of the energy to neighboring countries. As 96% of the generated power in Norway originates from hydropower, we have the ability to buy electricity when it is cheap and pump water into out reservoirs for use or sale when the prices increase. In 2015 93% of the power consumed by the members of the NordPool area was traded in the NordPool marked meaning that the spot prices for all members will vary based on the total production capacity. As the market becomes more dependent on renewable resources this creates an energy market that is more prone to large fluctuations until storage solutions become sufficient to balance the grid [32].

To calculate the total cost of electricity each month, C_T, we need to know power consumption, P_{con}, and spot price, C_s, for each hour, h, and each day, n as well as the consumer tax, C_c, state fee, C_{fee}, power tariff, C_p, maximum power, P_{max}, and fixed price, C_{fix}. The total electricity cost for a month with n days consisting of 24 hours can be calculated as follows:

$$C_T = \left(\sum_{n=1}^{n} \sum_{h=1}^{h=24} P_{con}^{h,n}\left(C_s^{h,n} + C_c + C_{fee}\right)\right) + C_{fix} + P_{max} * C_p \quad (2.23)$$

2.4 Linear optimalization

2.4.1 Model building

Linear optimization is a method which aims to find the optimal solution to a problem through maximization and minimization of an object function. An optimal solution is found when the system variables are given a set of values that creates the maximum or minimum value. In most cases these variables are required to satisfy certain conditions. In this case the optimization is said to be constrained, and if not, it is unconstrained. For real life applications it is key that the chosen variables offer a feasible solution that are within system constraints.

When creating a model for linear optimization it is useful to us the following steps [33]:

1. Start by identifying the needed actions of a system that has to happen for the goal to be reached.
2. Determine what resources are needed for each action and their respective requirements.
3. Write constraints as linear equations or linear inequalities. In some cases, the constraints will have to be converted to a linear format.

4. Identify the object function as a combination of all the previously mentioned actions and make sure that the units used are the same across all of them.

2.4.2 Mathematical presentation

The presentation of the problem is an important part of understanding what is trying to be solved. This can be done in different ways depending on the underlaying structure of the problem [33, pp. 18–21].

A problem in verbose standard form can be expressed as

$$\begin{aligned} &\textit{Minimize} && c_1x_1 + \ldots + c_nx_n \\ &\textit{Subject to} && a_{11}x_1 + \ldots + a_{1n}x_n = b_1 \\ & && a_{21}x_1 + \ldots + a_{2n}x_n = b_2 \\ & && \quad\vdots \qquad\quad \vdots \\ & && a_{m1}x_1 + \ldots + a_{mn}x_n = b_m \\ & && x_j \geq 0,\ j = 1, \ldots, n \end{aligned} \tag{2.24}$$

For problems that involve collections of sums can be expressed as

$$\begin{aligned} &\textit{Minimize} && \sum_{j=1}^{n} a_j x_j \\ &\textit{Subject to} && \sum_{j=1}^{m} a_{ij} x_j = b_i,\ \ i = 1, \ldots, m \\ & && x_j \geq 0,\quad j = 1, \ldots, n \end{aligned} \tag{2.25}$$

It is common to use terse standard form when the values are represented as matrixes

$$\begin{aligned} &\textit{Minimize} && c^T x \\ &\textit{Subject to} && Ax = b \\ & && X \geq 0 \end{aligned}$$

Where c and A are coefficient matrices, b is the right-hand side vector and x is the vector of decision variables.

2.4.3 Simplex method

There are multiple ways of solving linear optimization problems, but commonly used approaches include the Simplex and Revised simplex method [33, p. 78].

Step 1:

Convert problem into standard form. This means that 1) The problem must be a maximization problem, 2) The constraints must be given in less-than-or-equal-to inequalities, and 3) All variables must be non-negatives.

To address this an object function that aims to minimize its output can be multiplied by -1 on each side of the equation to become a maximization problem, and similarly both sides of a greater-than-or-equal-to constraint can be multiplied by -1 as well.

Step 2:

To convert each inequality constraint to equality constraints it is necessary to include what is known as slack variables, *s*, to the constraint functions. An example is shown in equation 2.26 and 2.27.

$$x_1 + x_2 \leq 3 \qquad (2.26)$$

$$x_1 + x_2 + s = 3 \qquad (2.27)$$

Step 3:

A tableau consisting of all the variables and their associated constants is created to perform row operations on the linear programming model. The constraint functions are implemented in the first rows and the object function in the last.

$$Maximize: z = 8x_1 + 10x_2 + 7x_3$$
$$s.t.: x_1 + 3x_2 + 2x_3 + s_1 = 10$$
$$x_1 + 5x_2 + x_3 + s_2 = 8$$

x1	x2	x3	s1	s2	z	b
1	3	2	1	0	0	10
1	5	1	0	1	0	8
-8	-10	-7	0	0	1	0

Figure 19: Example of the table of a minimization problem. Note that the function is converted to a minimization problem from its initial form.

Step 4:

The tableau is checked to see if an optimal solution is obtained. This is true when all values in the last row are greater-than-or-equal-to zero.

Step 5:

If an optimal solution is not found as is the case in figure 19 the next step is to find the smallest negative in the last row. In figure 19 this is -10. A pivoting variable is then found from the column that contains the smallest number; in our case this is x2. We then establish the indicator value based on the smallest value by dividing each constraint, *b*, with the value in the respective row. In our case this becomes 8/5 and 10/3. The chosen pivot value is then 5 as 8/5 is the smaller indicator.

Step 6.

For the tableau to be optimized the pivot value must become a unit value of 1. This also means that the other values in the pivot column must be set to 0 as the tableau is being optimized. The rest of the row containing the pivot value is then divided by the pivot value. To keep the tableau equivalent the values not contained in the pivot row or column must be calculated based on the new pivot values.

This value is calculated as: New value = (-1*value in old pivot column)*(value in new pivot row)+(old value). This pivot is shown in figure 20.

Old Tableau:

x1	x2	x3	s1	s2	z	b
1	3	2	1	0	0	10
1	(5)	1	0	1	0	8
-8	-10	-7	0	0	1	0

Old pivot column

New Tableau:

x1	x2	x3	s1	s2	z	b
2/5	0	7/5	1	-3/5	0	26/5
1/5	(1)	1/5	0	1/5	0	8/5 ← New pivot row
-6	0	-5	0	2	1	16

Figure 20: Pivot of tableau

Step 7.

We return to step 4 to check for an optimal solution. If none is found step 5-6 is repeated until an optimal solution is found. Then the value of each variable is decided based on it being a basic or non-basic value. A basic value means that it has a single 1 value in its column and is then set to be the same as the beta value. If a column contains a non-basic value, the value of the variable is set to zero. The final optimal solution is then calculated with the object function and the obtained values.

A flow chart showing the process when applied as an algorithm is shown in figure 21.

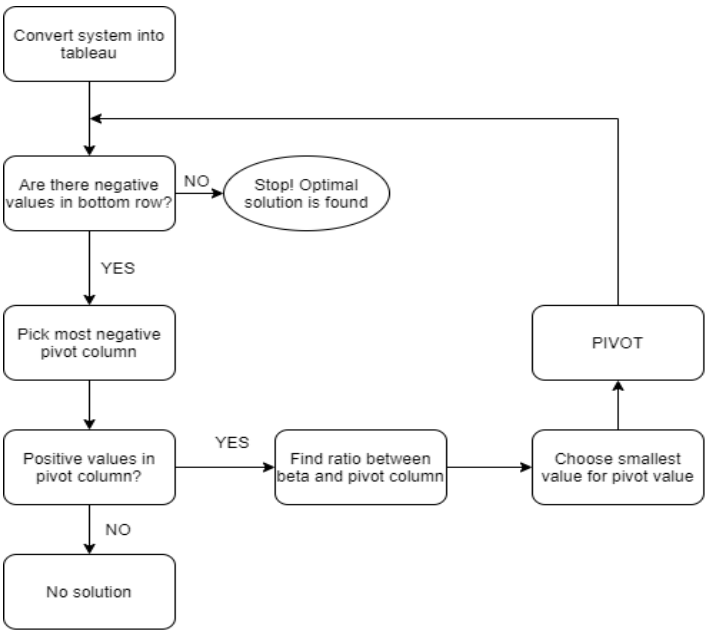


Figure 21: Simplex algorithm flow chart

2.5 API

Application Programming Interfaces, better known as APIs, allows two applications or services to talk to each other by sharing useful and important data. This allows for users to easily obtain relevant data without having to work with the source code of the programs. The API is the code which governs the access points for the server with information [34].

Figure 22: The role of an API in the system of things

The data format APIs usually works with is called JavaScript Object Notation (JSON). This is files that store data in key/value pairs and are easily read by all the major coding languages such as Java, Python, JS or PHP [34].

The main benefits of APIs are scalability, ease of use and platform independency. This makes working on the internet of things (IOT) a lot more streamlined and accessible.

Different types of APIs include:

Open APIs that offer publicly available data.

Partner APIs custom designed for enterprises used for specific partnerships and use cases.

Private APIs which are only used internally for private projects.

2.6 Weather forecast

Accurate simulations for day ahead power generation with renewable sources such as PV and wind power are highly dependent on accurate forecasting methods.

Modern approaches use satellites to gather input data at short intervals. For solar radiation forecasting satellite images are collected and processed before albedo is estimated based on statistical models using previous images for the same area for the previous month. The albedo makes it possible to separate clouds from bright spots on the ground created by snow, sand, or shiny surfaces. The original picture is then decomposed into cloud covered sky and clear sky. By performing 3D cloud modelling the thickness of the clouds, also known as “cloud opacity,” can be determined. Finally, a model made for calculating radiation based on aerosol data (dust, salt, smoke, etc.) as well as water vapor is implemented to obtain accurate estimations for GHI. Some of the recognized models are: Kasten, Bird, MAC, Iqbal-C and REST2 [35]. This data can then processed by another model to obtain DNI and DHI [36].

To obtain forecasted data many hours or days ahead a combination of models, such as machine learning, and available data is used to predict how the cloud cover will move. Longer forecasting periods will result in less accurate results. Because of this it is beneficial to update predictions often or simulate power output calculations close to the upcoming day.

3. Literature review

Different optimization techniques are being widely researched and tested to minimize the cost and maximize the utilization of renewable energy while using combined renewable energy generator (REG) and battery energy storage systems (BESS). The goal of these system is to manage the energy flow in such a way that each component is used to its full potential while the demand of the user is satisfied, and to keep the total cost at a minimum.

Table 1 contains a brief overview of existing research projects on the subject with their chosen method, goal, and state of testing for each.

Table 1: Previous studies on optimization of energy scheduling and BESS optimization.

Title	Method	Goal	State of testing	Ref
Impact of energy management system on the sizing of a grid-connected PV/Battery system	Generic algorithm (GA), MATLAB	Optimize system sizing with GA and suggest a simple operating strategy based on net power.	Simulations	[37]
Demonstration of reusing electric vehicle battery for solar energy storage and demand side management	Decision based Time of use demand side management	Creating an optimized battery management system with Kalman filter and run the system based on three different time of use scenarios based of real time measurements.	Physical system	[38]
Optimization between the PV and the Retired EV Battery for the Residential Microgrid Application	Nonlinear optimal control tool (BLOOM)	Optimizing sizing of PV system and maximizing yearly profit through nonlinear optimization of cost function.	Physical testing of components. Simulation of system	[39]
Optimizing the energy storage schedule of a battery in a PV grid-connected nZEB using linear programming	Two stage Linear optimization, MATLAB	Integrating zero energy buildings with the grid through linear optimization which manages PV generation and load demand.	Simulations	[40]
Particle swarm optimized fuzzy controller for charging–discharging and scheduling of battery energy storage system in MG applications	Fuzzy logic controller	Creating an optimization routine for a micro grid with multiple inputs.	Simulations	[41]

Optimization of the Battery Schedule for Residential Microgrid Applications	Nonlinear programming, MATLAB	Optimize the scheduling of a battery in a microgrid consisting of multiple generators.	Simulations	[42]
Battery capacity determination with respect to optimized energy dispatch schedule in grid-connected photovoltaic (PV) systems	Mixed integer programming	Peak shaving and load shifting of battery usage while considering battery degradation.	Simulations	[43]
Energy dispatch schedule optimization for demand charge reduction using a photovoltaic-battery storage system with solar forecasting	Linear optimization	Optimization of battery dispatch schedule with focus on solar forecasting.	Simulations	[44]
Design and mixed integer linear programming optimization of a solar/battery-based Conex for remote areas and various climate zones	Mixed integer linear optimization, MATLAB Two-stage	Optimized scheduling with particular focus on islanded mode and use in Iran.	Physical tests	[45]
Implementation of Optimal Two-Stage Scheduling of Energy Storage System Based on Big-Data-Driven Forecasting—An Actual Case Study in a Campus Microgrid	Linear optimization, Python	Two stage optimization of battery scheduling with daily and hourly timeframes.	Physical tests	[46]

An important aspect when creating a battery discharge algorithm is the chosen database and time interval. More accurate results will be possible if optimization is done closer to real time, but this will depend on the available data and the processing capabilities of the system. For a system such as the one used in [38] the consumption data is gathered in one minute intervals and measurements from the PV system in five second intervals. The system response is also done in five seconds intervals. This system is using real time data which differs from simulations in the sense that the time interval must be small for accurate system control.

For research which focuses strictly on optimization ahead of time a longer time interval may be sufficient, but different methods have been used to maximize the accuracy and thereby the efficiency of a hybrid battery system. In [46] a two stage optimization algorithm is being tested on a physical system. The first stage optimizes the system for cost reduction with day ahead values at one-hour

intervals, whereas the second stage aims to minimize the difference in input power from the grid between the day ahead values and the more accurate hourly values given in 5-minute intervals. This is done to describe the conditions of operation more accurately as the real time optimization becomes imminent.

The algorithm used in [46] can be explained with the simplified flow chart shown in figure 23 where $P_{L,t}^{f,d}$ and $P_{L,t}^{f,h}$ is the forecasted load data, $P_{PV,t}^{f,d}$ and $P_{PV,t}^{f,h}$ is the forecasted solar power generation data, $P_{u,t}^{s,d}$ and $P_{u,t}^{s,h}$ is the scheduled utility power and $P_{b,t}^{s,d}$ and $P_{b,t}^{s,h}$ is the scheduled battery power for day ahead and hour ahead values respectively.

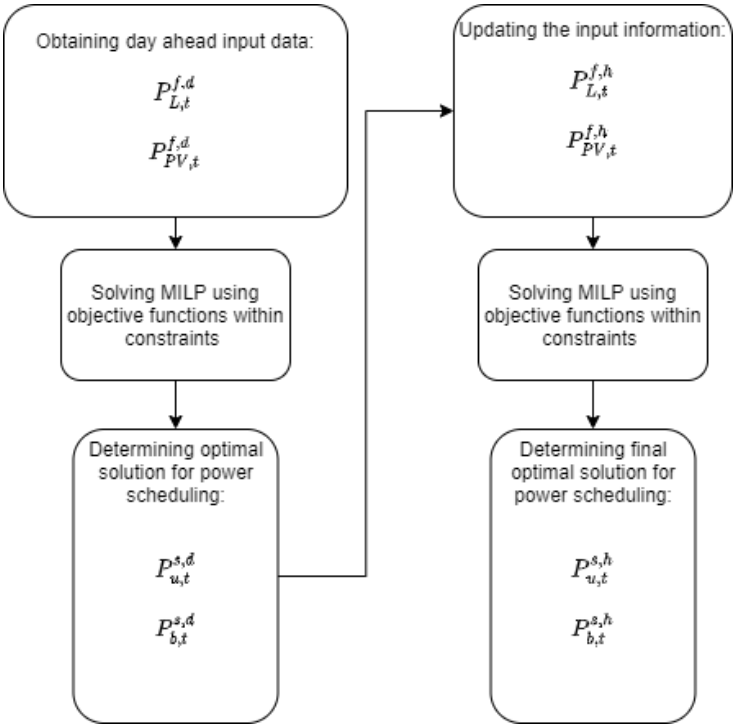


Figure 23: Flow chart for optimization done in [46]

By adapting a multi-step method, it is possible to obtain more accurate optimization results. From [46] it is made clear that this resulted in the BESS being used more frequently which compensates for errors in the day ahead scheduling caused by variations in the load demand and PV generation. This means that the battery is used to keep the utility demand at the values decided in the day ahead optimization so that the cost reduction is kept as similar as possible.

Other papers focus more on the dimensioning of the system with regards to total cost and not just the energy cost. In [37] a Genetic Algorithm (GA) is used to find the optimal sizing of a hybrid system consisting of PV panels, batteries and a plug in electric vehicle (PEV). A Generic algorithm is a way of optimizing a system based on previous combinations of values for the given variables. By comparing the results with either a target value or the previous value the algorithm loops until an optimal value is found or a set number of simulations is reached [47]. This way of finding an optimal solution is more dynamic than linear approaches and works well if the system should test different combinations of components in a system.

Although the main aim of the paper in [37] is to find optimal sizing they are also using an energy management system (EMS) to best schedule the charge and discharge based primarily on the net load resulting from the solar power generation and load demand. This way of optimizing battery scheduling is also done in [39] although the sizing of the PV system is done linearly and the optimal solution is found to be the one which gives the minimum electricity cost together with a battery of a set capacity. The study further finds that it is possible to obtain a usable optimal schedule for the battery with the given data of solar irradiation and load demand together with a simple system model.

While the goal and scope of previous studies may differ in some aspect, they usually follow similar control strategies based on much of the same variables. The main differences depend on the complexity of the component modeling. In article [37], [40] and [42] simple models are used for PV and batteries. The PV system is based on measured global irradiation which then is converted into produced energy based on cell efficiency. This method does imply some problems with regards to day-ahead scheduling as the forecasted irradiation is dependent on many different parameters, but solutions do exist. The flow of power through the battery is also simplified so that the battery only acts through limits of charging and discharging power, usually with an efficiency included to account for losses. This gives an approximate behavior of the battery and power flow but could differ from actual use case scenarios. A system like this can be described through a calculation which looks at the previous SOC, *SOC(i-1)*, the power flow, *x(i)*, to or from the battery at time, *t*, and the total available storage capacity, E_{bt_max}, to obtain the current value, *SOC(i)*, as seen in [42].

$$SOC(i) = SOC(i-1) - \frac{x(i)}{E_{bt\_max}} \qquad (3.1)$$

This calculation can be done with respect to the change in capacity seen as a percentage as done in formula (x) or it can be done by directly looking at the current charging or discharging power as seen in [37] and use this value to update the remaining capacity for each time step. This is useful to make the system more comprehensive as charging and discharging can be included as separate variables. In [39] a more complex model of the battery is constructed through an RC circuit which includes the inner ohmic resistance, R_2, the current, *I*, and an RC network, R_1 *and* C_1, used to decide the time constant behavior of the system as can be seen in figure (x). To obtain the total power during charging and discharging the open current voltage, V_{oc}, is calculated at each time step. This makes it so that the simulations can be done with parameters that more accurately match the specifications of the physical battery. K_0 and K_1 represent two battery specific parameters which can be calculated through the continuous-time frequent-domain form of the battery model [48].

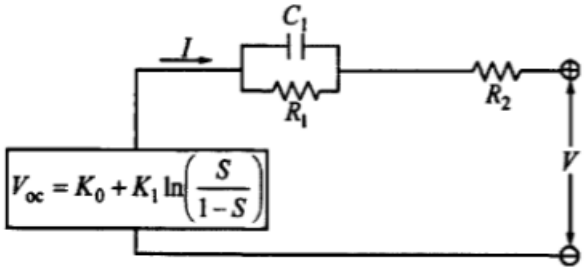


Figure 24: RC model of battery [39]

To obtain more realistic optimization results it is necessary to include variables such as temperature and minor losses. In article [41] simulations are done with fuzzy logic. This is a method where a model is fed many different parameters which can impact the outcome of the system. The parameters are not given optimal values, and through trial and error the algorithm changes the values based on the observed output where it tries to find an optimal solution based on the prerequisites of the problem [49]. This method allows for nonlinear systems which makes it possible to introduce more complexity to the system design. In the case of [41] the battery is modelled with respect to temperature which is dependent on the thermal resistance, R_{th}, Laplace transform of ambient temperature, $T_a(s)$, Laplace transform of power loss, $P_{loss}(s)$, and thermal time constant, t_c. To find the cell temperature these parameters except from the thermal resistance is modelled as follows.

$$T_{cell}(t) = L^{-1}\left(\frac{1}{1+s*t_c} * (R_{th} * P_{loss}(s) + T_a(s))\right) \qquad (3.2)$$

This way of modelling such a system makes it possible to maintain advantageous temperatures for the battery cell so that the lifetime is maximized. In the given article the temperature is kept between 0° and 35° C.

In [43] both mixed integer programming (MIP) and fuzzy logic is used to compare how the inclusion of a nonlinear battery model impacts the system design decisions. The focus of this article is to see the effects of battery degradation can impact the total cost of the system. For the linear model with an ideal battery a constant linear ageing coefficient, Z, was included to account for the ageing losses. The article concludes that it is important to factor in the capacity loss associated with the ageing of batteries when deciding on the sizing of the components as this plays a key role on the total system price over its life span.

Error with regards to solar forecasting is addressed in [44] as an important factor to account for when designing a hybrid system which includes PV generation. This could also be true for other distributed generators (DG's). The study found that over-estimation of energy generation caused the batteries to discharge below its safe limits to compensate for the lack of power to the demand load, which henceforth is called a battery failure as per the cited paper. If this happens regularly it can cause major strain on the battery compared to what is supposed to be optimal scheduling.

Until now most of the reviewed papers have focused strictly on system modelling and minimizing the system and operating costs through different methods. Another interesting and important aspect when using BESS is how the battery optimization algorithm can be tweaked to also maintain good battery health during operation. According to a copious study done at Chalmers University of electrical engineering [50] the effects of controlling the SOC and depth of discharge (DOD) can substantially increase the expected lifetime of batteries. The study looks at cases which focuses on the rate of charging (C-rate), number of cycles and SOC while at standby . To measure the capacity and impedance of the battery cells a reference performance test procedure was used. This means that the cells were tested at approximately each 200 full cycle equivalents and consisted of discharge and charge pulses at 5C and 1C every 10% SOC level.

The article concludes that large C-rates and high SOC impacts the lifetime greatly. The data found from experimental data was then used to create models of three cars with different use patterns. These models were then used to assess the real-life implications of different scenarios. When the battery is in an idle state it is possible to reduce the impact of calendar ageing by keeping the battery at low SOC.

Depending on the case studied the effect of storing the battery at 15% compared to 90% SOC increased the expected lifetime by 44-59%. Higher C-rates also reduces the lifetime of the battery, especially at higher SOC. Synthetic tests show that running at low C-rates at high SOC can be even more impactful on battery life than high C-rates at low SOC. Another finding, although not surprising from a theoretical standpoint, is that a larger battery reduces the strain when using the battery in all regions because of larger upper and lower margins.

In figure 25 we can see some of the test results at two different intervals and varying C-rate.

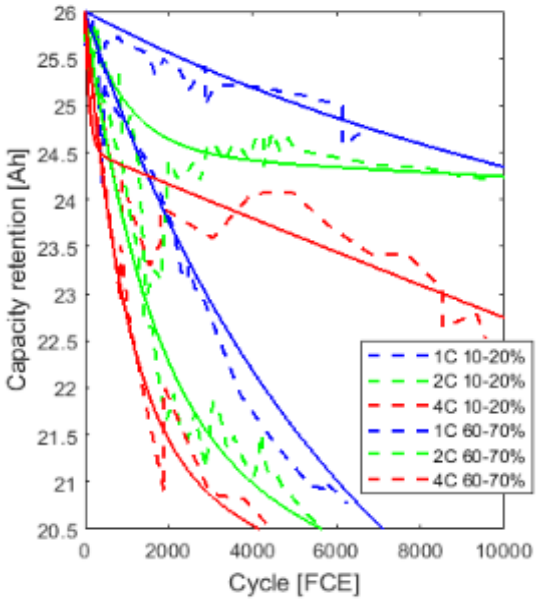


Figure 25: Capacity retention at varying intervals and C-rate [50]

In [51] a lithium-ion cell was cycled at five different intervals (5-25%, 25-45%, 45-65%, 65-85% and 75-95%). The cells were charged at 2.5A and discharged at 10A for a total of 4000 cycles. At each 500-cycle interval the remaining capacity was measured through Ah-counting and the correct amount of energy was discharged to obtain the correct lower value for the chosen interval.

The results concluded that a higher loss occurs when cycling at higher SOC. In this report the most loss happened at the 65-85% interval where a little under 8% capacity loss happened due to mainly a higher loss of active lithium and anode material. In comparison the 75-95% interval had a loss of a little over 6% but also the highest loss of cathode material. Cycling at 5-25% led to the lowest capacity loss of approximately 3%. The report concludes that intelligent load management can help reduce the cycling impact on battery life.

4. Methodology

As this thesis compares results between a custom-made algorithm in Python and the commercially available tool, HOMER, a description of both systems will be made in this chapter. Some of the results from the HOMER simulations have been acquired during a study from the semester prior to the master's thesis [52], but to verify and compare the custom-made algorithm additional simulations have been done in both programs with the same historic datasets and parameters.

4.1 Case study and system design requirements

The case study for this thesis is a planned recycling and sorting facility located outside of Lillesand, Norway with longitude and latitude coordinates: 8.1646, 58.1817. A report has already been made regarding system design by Multiconsult, and this thesis will be using some of the component parameters on that. The facility with a suggested PV layout is shown in figure 26.

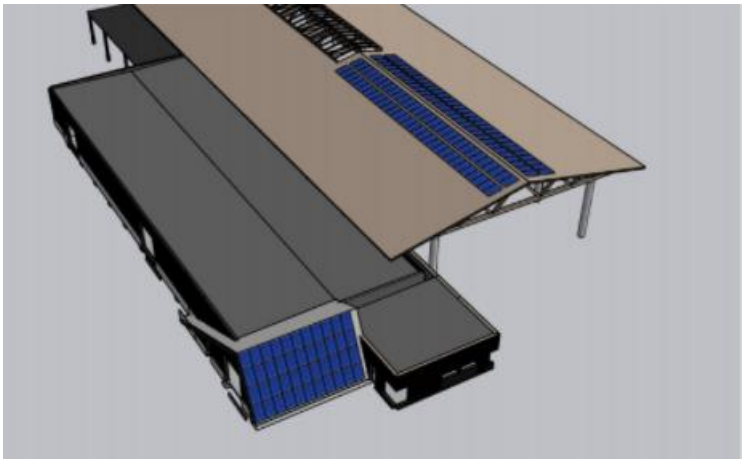


Figure 26: Suggested PV layout by Multiconsult

The PV system must have a 25-year guarantee which promises a minimum of 80% efficiency after reaching the end of this time span. The product guarantee must be 10 years. This thesis is therefore based on a 25-year simulation period when comparing to HOMER.

The proposed system includes 60 kWp installed PV panels divided between a roof with east and west facing surfaces as well as a south facing façade. The roof has an angle of 12° to each side and the façade has a steeper angle of 65°. In the Multiconsult report the PV panels on the rooftop has a capacity of 390 kWp and the Façade uses 320 kWp panels. For the purpose of this study the same 320 kWp panel is used for all surfaces. PV simulation done by Multiconsult is done in advanced tools such as PVsyst, PV*SOL and Polysun and climate data is collected from PVGIS. This should account for all system losses including but not limited to DC-losses, shading losses and soiling losses.

Battery dimension is recommended to be 50 kWh +- 10%. The output voltage must be 400V +- 10% to match the electricity supply of the system. The power output should be minimum 20 kW, the round trip efficiency should be at least 80% and the battery should be capable of minimum 2000 full cycles before reaching 70% remaining capacity. The main function of the battery is to maximize self-consumption of renewable energy and minimize the power tariff.

These values were used as the basis for the optimization done in the previous study [52] which focused on system dimensioning. For the purpose of this study the optimal system dimensions found in the

previous study is used as the base in this thesis. As the main focus of this study is to see how the parameters used for energy scheduling of the battery can impact the system performance all component dimensions are kept constant throughout the simulations.

4.2 Historic data

4.2.1 Weather data

The historic irradiation and temperature data used in both Python and HOMER simulations was collected at the roof of the J5 building at UiA in 2020 and is given in minute intervals in its raw format. The data was then converted into hourly averages for use in the simulation. The heigh and location difference from this location and the building in the case study may cause the results to differ from what it would be at the case site. But for the purpose of assessing how the system will work depending on different seasons and conditions it is deemed sufficient. It is also worth noting that HOMER only uses global horizontal irradiation data (GHI) which is then converted into direct and diffuse irradiation using formula 2.15 and 2.16 given in 2.2.2.2. The same separation model and GHI data is therefore also used in Python to make the comparison valid. Because of this, only the historic GHI is needed from the weather data. Some of the datapoints were missing and this was corrected for by copying data from the day or days prior and pasting it in the correct time frame. This change makes the data a little less valid as real-life examples but had to be done to match the data points for simulation while still having non-zero values. Irradiation and temperature data for January and June is shown in figure 27 and 28 to highlight two months with extreme highs and lows.

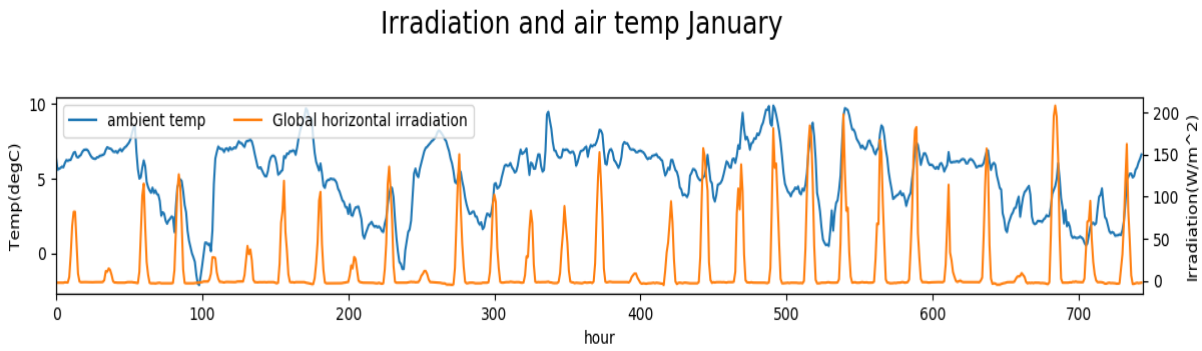


Figure 27: Historical irradiation and temperature data for January 2020

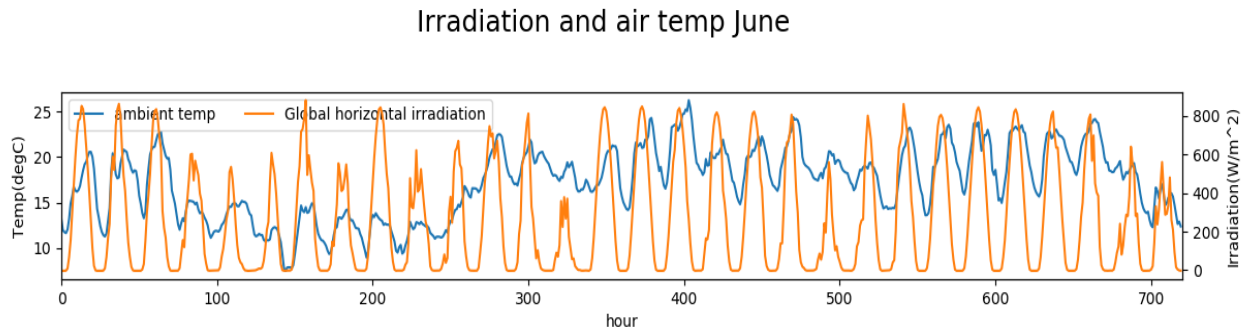


Figure 28: Historical irradiation and temperature data for June 2020

4.2.2 Electricity price data

The historic price data has been obtained from NordPool historical market data and is the average spot pricing from 2013-2019 given on an hourly basis [53]. The final dataset includes the electricity fee and energy fond fee which comes to an additional 0.2724 NOK/kWh as the facility in this thesis exceeds a yearly power consumption of 100 000 kWh [54]. For the energy sold to the grid an additional 0.04 NOK/kWh had been added to the spot price as this is the standard for customers at Agder Energi. The total purchase and sell price for electricity in January and June is shown in figure 29 and 30.

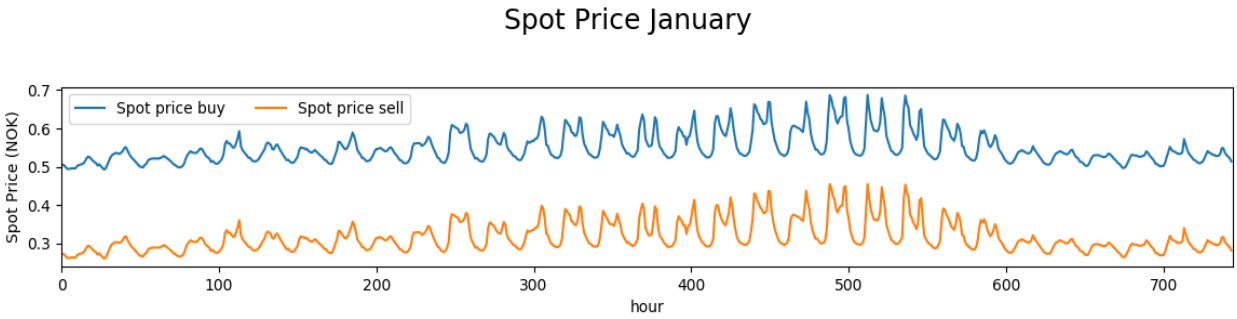


Figure 29: Spot pricing of electricity in January 2020

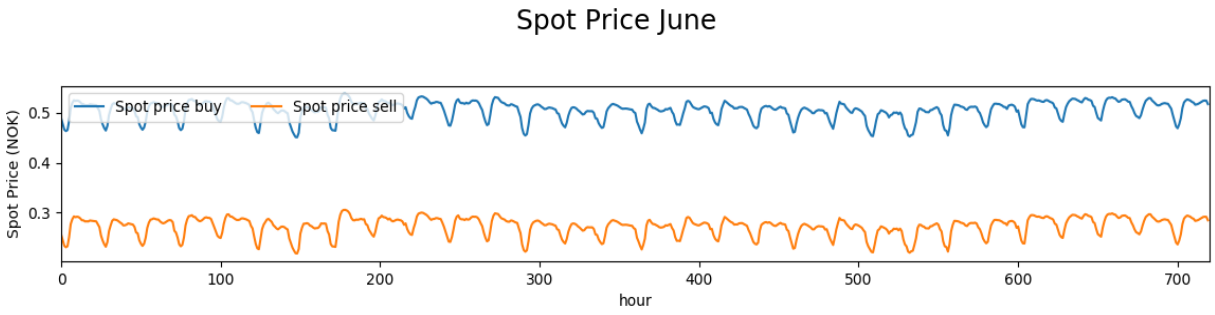


Figure 30: Spot pricing of electricity in June 2020

As Avfall Sør will be on a power tariff plan an additional cost is added for each kW at peak consumption for each month.

4.2.3 Load data

The historic load data is the measured electricity consumption in 2019 at a comparable location as that of the case study. Some changes will be made such as the removal of a heated area and the replacement of conventional construction machinery with 15 electrical machines. There will also be roofing above the driving area as well as a mountainside causing shading at the location which will increase the need for lighting. The electric machinery will reach a peak consumption of 5.5 kW at startup and around 1.9 kW during operation for 5 minutes 5 times a day on average. For the purpose of this study these corrections have not been made to the power demand, but it could be interesting to see how the battery would react to the sudden peaks caused by the machinery. The power demand for January and June is shown in figure 31 and 32.

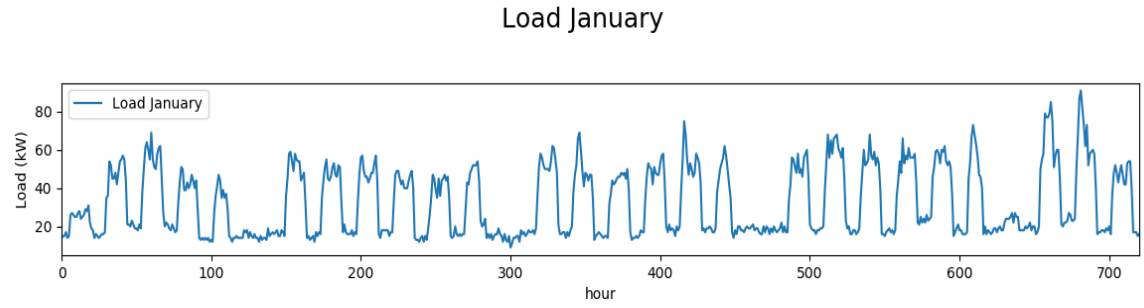


Figure 31: Load demand January

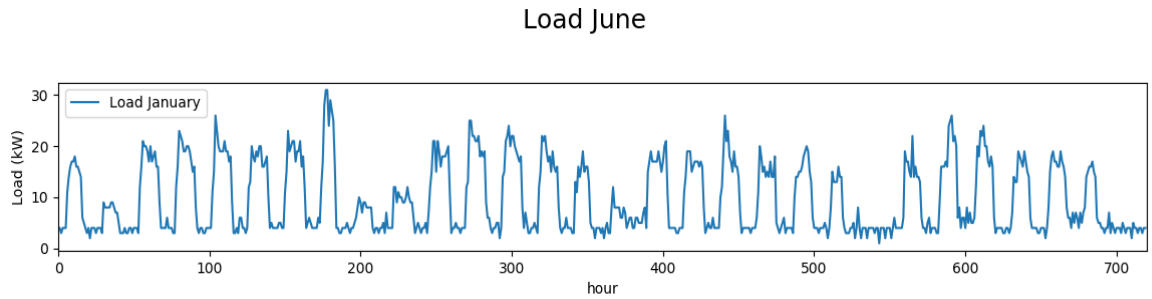


Figure 32: Load demand June

4.3 Day ahead data

For the day ahead optimization the algorithm uses data gathered from two different API's. The API used for day ahead spot prices is created from open-source data gathered from Entsoe, which is a transparency platform open for public use [55]. For weather forecast data an API created by Solcast is used [56].

4.3.1 Weather data

The temperature and irradiation data are collected from Solcast servers through their API. Solcast [56] is a Global forecasting service which offers accurate forecasts with a variety of options and a large historic database. This is mainly a paid service but offers a free option for research purposes [57]. This includes 10 calls to the API per day and 650$ in historical data credit. For the purpose of this study only

the API is needed and 10 calls a day is more than sufficient as the day ahead data is required once per simulation. Because of this, and their well-designed web page this is the chosen tool for day ahead weather data for this study.

Solcast uses a fleet of five satellites to obtain global weather information as described in chapter 2.6. For Solcast specifically the imaginary data is given in 1-2km resolution and is collected in 5–15-minute intervals depending on what satellite is being used at the given location. Solcast uses a combination of models including a modified version of the REST2 clear sky model, and an in-house separation model created by Dr. Nick Engerer to estimate the solar radiation and obtain the diffuse and direct irradiation [36]. For this thesis, the day ahead direct and diffuse irradiation is taken directly from the Solcast API. Solcast also provides irradiation predictions in 10, 50 and 90 percentiles so that the user can choose whether to use the most probable irradiation or be very safe and use the lowest estimated values depending on what is best suited the use case. In this thesis the 50-percentile median has been used. Data is given in either JSON, CSV or XML format and for this thesis JSON is chosen as the preferred format. Solcast also includes a tool which can inform the user how accurate forecasts in a given area is based on historical data. For a location similar to the one used in this study the error comes to between 4.82% and 7.69% for day-ahead forecasting.

4.3.2 Electricity price data

Spot price data for the day ahead simulation is collected from an API based on data from the Entsoe transparency platform as this is open for public use [55]. This is done as NordPool have a strict policy regarding use of their API [58]. This data is the same as what would have been obtained from NordPool but is not updated until 13:00 each day. This means that day ahead simulations must be done after this point when using this system. The API is open for public use and made by a user in a Norwegian forum based on information given by Entsoe on their webpage. As the day ahead spot prices does not change only one call per day is needed. For the buy prices additional fees of 0.2724 NOK/kWh are added at each time step. For the sell price 0.04 NOK/kWh is added.

4.3.3 Load data

For the day ahead load data a part of the historic dataset correlating to the simulated date is used. This does not accurately represent an actual load demand and the day ahead simulation can therefore only be seen as a theoretical optimization and not a realistic one. When the load pattern of the construction in question is better known a possible load demand can be obtained and fed to the algorithm. As the load demand cannot be perfectly predicted optimization done closer to real time would be beneficial to correct the system response accordingly. This is however not done in this thesis.

4.4 System design

In this chapter the chosen system design used in each program is described. As the main focus of this thesis is to assess how different battery scheduling methods can minimize electricity cost and improve battery life, the exact dimensions and system cost for the inverters in the Python program have not been

calculated and is assumed to be perfectly dimensioned to the PV and battery system. As described in the theory there are many ways of installing PV and battery systems. Choosing either AC or DC coupling is not relevant to simulation results but is important to better understand the physical limitations of the problem and how this impacts real life performance and decision making.

4.4.1 Old design

In the previous study [52] both the PV system and battery is connected to the DC bus and each of the three PV strings is connected via its own MPPT charge controller. This means that the central inverter must undertake charging, discharging, and PV power inversion. This leads to less components but causes more strain on the inverter. The system design is shown in figure 33.

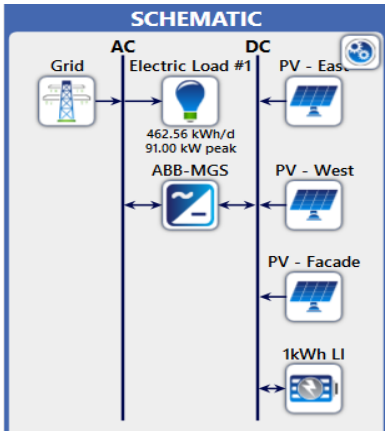


Figure 33: System design used in the previous project

4.4.2 New design

For this thesis further investigation was done with regards to system design which led to the conclusion that for a grid connected system of this size the most logical and optimal solution is to adapt an AC coupled battery and PV system which can be seen in figure 34. This allows for each of the strings to have its own inverter and MPPT controller while not having to adjust with respect to the battery voltage, only the other strings. This is seen as the best solution as the market for large scale MPPT charge controllers is limited when the battery exceeds a certain size and is better suited smaller off-grid solutions. When using AC coupling an inverter and battery controller can be properly dimensioned to fit the battery making the system design more optimal. The main drawback is the inclusion of more components which can result in increased initial cost and more losses. With this design MPPT controllers are also not necessarily needed for all the strings. In the case of this study, it might only be beneficial on the façade which is more prone to shading. For this thesis all of the strings are MPPT controlled.

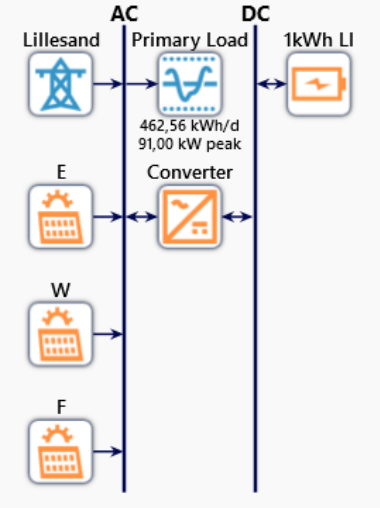


Figure 34: AC coupled system design

4.5 HOMER Pro and HOMER Grid

Hybrid Optimization of Multiple Energy Resources (HOMER) is a company which focuses on economic optimization of microgrid designs. Their software combines the engineering and economic aspect of system design in a single model which simplifies system planning [59]. During one of my prior projects [52] the same case study was analyzed and optimized using both HOMER Pro and HOMER Grid.

HOMER Pro focuses mainly on comparing different solutions with regards to system design and dimensioning so that the total cost is minimized over a given time prospect. This program includes a large number of variables and constraints so that many systems can be simulated and compared simultaneously to obtain the best possible solutions with the least amount of risk [60]. For the purpose of the previous study HOMER Pro offered an option to implement a renewable fraction constraint which was deemed an important criterion for system design from a renewable energy standpoint. For the purpose of the previous study only the base program was used, but for more specific tasks such as battery degradation modelling or multiple generators designs several modules are available for purchase.

Homer Grid is another program that offers much of the same customization but has a focus on behind-the-meter system design. This includes peak shaving and optimization of energy mix. The program also includes modern solutions such as EV charging [61]. For the purpose of this thesis and the prior study the main feature of interest of this program is the inclusion of an advanced storage model and dispatch strategy.

The dispatch strategy used in HOMER Grid is said to have “perfect foresight”. This includes knowing the PV generation, utility rate schedule and electricity demand for each time step in the future. The dispatch algorithm looks 48 hours ahead to decide what the best battery actions are. HOMER Grid favors offsetting grid purchases rather than selling energy stored in the battery. This means that the battery will tend to wait until days with capacity shortage and high prices if deemed reasonable. For peak shaving the discharge algorithm tries to find the lowest possible grid power demand limit. This is done by ensuring that the available power from any generators or batteries will be able to meet the load demand at each time step for the given month [62].

4.5.1 System dimensioning using HOMER Pro

The following chapter describes the parameters and constraints used in HOMER Pro for optimization done in the previous project [52].

Among four different scenarios two were of the most interest and considered the most relevant to the study. Each scenario had the same simulation variables shown in table 2 with the only difference being a renewable fraction constraint of respectively 25 and 50 percent.

The main parameters in HOMER are the pricing for each component, load demand schedule, solar data, temperature, and electricity prices. These were kept the same during all of the simulation scenarios.

Constraints and parameters:

- Total runtime of simulation is set to 25 years with 8760 timesteps for each year.
- Electricity pricing when drawn from the grid is based on spot prices, energy fees, the demand tariff and consumption price from Agder Energi, and a fixed monthly price to the electricity provider.
- Feed in prices uses average spot pricing from 2013 to 2019 with an added 0.04 NOK/kWh. Sales to the grid is also restricted to only happen when the price is greater or equal to 0.28 NOK/kWh when simulating in HOMER pro.
- The pricing for the main system inverter is set to 1666 NOK/kW as this is close to the current market value [63]. The lifetime is set to 15 years. Maintenance is set to 10 NOK/kW/year. Efficiency is set to 95%.
- The cost for PV is set to be 5212 NOK/kW installed capacity including installation based on Avfall Sør price estimate.
- MPPT converters for each of the three strings is set to 1000 NOK/kW [64]. Maintenance is set to 10 NOK/kW/year.
- The lifetime of the PV system is set to 25 years.
- NOCT for the PV system is set to 47°. Power loss per degree change is set to -0.5%/°C. Efficiency at STC is set to 13 %.
- The loss factor is set to 20% due to soiling etc. This converts to a derating factor of 80% in HOMER.
- The reflectance factor is set to 20%.
- East string has an azimuth of -81, West string 99 and the façade is set to 9.
- The slope of the roof mounted system is set to 12°. And the façade is fixed at 65°.
- MPPT dimensioning is set to be 1.2 times the maximum power output from the PV string as proposed by HOMER and an efficiency of 95% [65].

- Battery size is set to be strings of 30, 1kWh, 6V, 500A cells with 24kWh available capacity based on the upper and lower limits. This results in a base 180V battery pack with a maximum output power of approx. 90kW. This is the same as a first gen battery from a Nissan Leaf [66].
- The upper charge and lower charge limit are set to 100% and 20% respectively.
- Battery price is set to be 2160 NOK/kWh with replacement cost of 2000 NOK/kWh. Maintenance is set to 10 NOK/kWh/year. This reflects what Avfall Sør aims to pay for second use batteries and may be lower or higher.
- Solar data is downloaded from a NASA database within HOMER with average values from 1983 to 2005 to predict the data for the given location.
- Temperature data is using downloaded NASA data within HOMER from 1984 to 2013 to create a prediction of the temperatures at the given location.
- The optimization algorithm is using a Load Following strategy with stock settings

Table 2: Simulation parameters used in the two dimensioning scenarios

Component	**Amount**	**Capacity**	**Tilt**	**Optimizer**
PV – East	---------	10-80 kWp	12°	YES
PV – West	---------	10-80 kWp	12°	YES
PV - Facade	---------	14.1 kWp	65°	NO
Battery	1 – infinite	24 kWh - infinite	-------	YES
Converter	1	0 – infinite	-------	YES

From these simulations the scenario with a 50% renewable fraction constraint resulted in a large amount of excess energy which could not be converted to AC by the inverter with its optimal dimension. The optimal dimensions of a system with a 25% renewable fraction constraint resulted in a more balanced system with a smaller energy loss percentage and is therefore used for further simulations in both HOMER grid and Python for this thesis.

4.5.2 HOMER grid setup for algorithm validation

Because of limitations with the Python algorithm some changes have been made with regards to constraints in HOMER grid so that the program can be used to verify the validity of the LP algorithm results. The main limitation is that losses and dimensioning of the inverters is not included in the Python script. This includes the MPPT string inverters and the battery inverter. For this reason, these components are assumed to be ideal and have a capacity of 1000 kW as an infinite size is not possible. The price of these components uses the dimensions from 4.5.1 to make the final system cost more realistic. From the dimensioning optimization we find that the optimal sizing of the PV system and battery size given the constraints and pricing in 4.5.1 to be: 27.9 kW roof east, 17.7 kW roof west, 14.1 kW façade south and a 48 kWh battery. Another difference from the HOMER Pro simulation is the PV module data used as this is set as the parameters for the 320W module given in appendix.

The main difference between the HOMER and Python simulation is how battery scheduling is done and how the PV array power output is calculated. HOMER uses the following formula [27]:

$$P_{PV} = Y_{PV} f_{PV} \left(\frac{\bar{G}_T}{G_{STC}}\right) [1 + \alpha_p (T_c - T_{c,STC})] \qquad (4.1)$$

Where:

P_{PV} is the generated power, Y_{PV} is the rated capacity of the PV array under STC, f_{PV} is the derating factor to account for losses, $\bar{G}_T$ is the incident irradiation calculated as described in 2.2.2.2, G_{STC} is 1000 W/m^2, α_p is the power loss coefficient (%/°C), T_C is the cell temperature at the current time step and $T_{c,STC}$ is 25°C. HOMER does use a more complex calculation to obtain the cell temperature than what is used in Python [27]. Because of this the MPPT in HOMER does not actually impact the power generation results unless a loss is included. It will only impact the cost of the system which is not in focus for this thesis.

4.6 Python script

Python is a powerful coding language for data manipulation and simulation as many different open-source libraries are available for almost any given problem. For the problem in this thesis Pyomo [67] was used to create the optimization algorithm, SciPy.optimize [68] was used for root finding to solve implicit equations and Request [69] was used to import data from the two APIs. Pyomo is an open source package that can solve many different optimization problems including linear problems as is done in this thesis. The Python script utilizes basic data analysis methods and list comprehension to analyze and split datasets into data that can be used in the LP algorithm. The code is given in appendix.

4.6.1 PV power generation

To calculate the power the incident irradiation for each time step had to be calculated. This was done by following the procedure described in 2.2.2.2. As mentioned in that chapter some combinations of angles and solar positioning can cause large deviations. As the Python algorithm assumes an infinite large inverter all energy generated will either be used or sold to the grid. If the calculated irradiation becomes very large this will in turn cause the generated current and therefore also power to be very large. Extreme

peaks even for short amounts of time will impact the final electricity cost greatly. Figure 35 shows the calculated incident irradiation for the west facing panels in May.

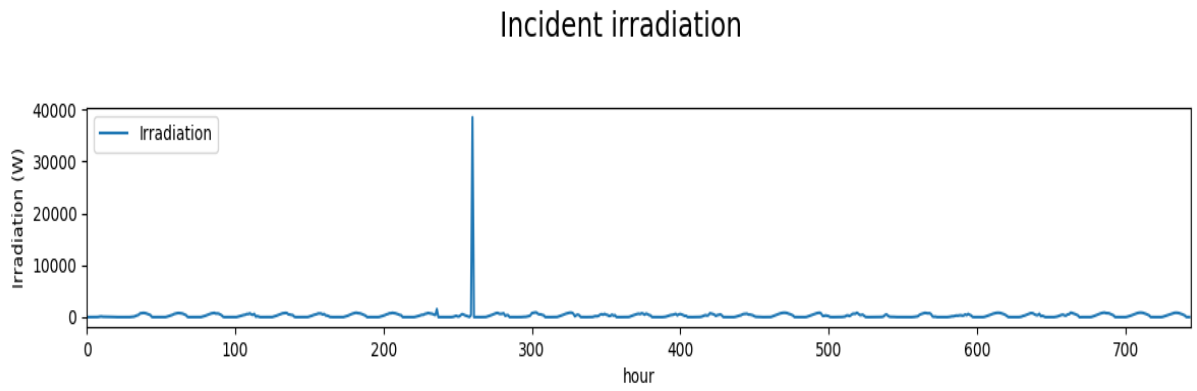


Figure 35: Deviation in incident irradiation in May

This one value alone would make the total results for the entire month unusable for any actual comparison. To address this issue a piece of code was implemented to replace all values that are negative or above a certain threshold with zero. This could also have been fixed using data manipulation before calculations, but due to time limitations this was not done.

This thesis chose to use the parameters from a poly-christalline silicon module shown in appendix when calculating the generated current using the single diode model as described in 2.2.3.1.

To simulate the behavior of MPPT the value for V_{mpp} is corrected for with the temperature coefficient given by the manufacturer for each time step. V_{mpp} also have a non-linear dependency on irradiation to a less degree than temperature and is for the purpose of this thesis not accounted for. As the V_{mpp} is calculated with known values for each time step the MPPT is assumed to be ideal. The change in generated current with respect to temperature is accounted for in the single diode model.

As mentioned in 2.2.3.1 some of the parameters used in the diode model must be calculated or assumed. R_{sh} and R_s were calculated from the IV curve at STC of the chosen module as described in figure 7. R_s is found from the open-circuit point and R_{sh} is found at the short-circuit point. This is not the most accurate calculation but results in 0.58Ω and 198Ω respectively which in reasonable values for further calculation.

The total power production for each string is then calculated based on the generated current, I_{module}, the V_{mpp}, and the number of modules corresponding to the installed capacity, $E_{string,tot}$, and rated capacity of each module E_{module}.

$$E_{string} = I_{module} * V_{mpp} * \frac{E_{string,tot}}{E_{module}} \qquad (4.2)$$

Calculated generated current for a day in June for each of the three string is shown in figure 36, 37 and 38. The GHI and incident irradiation is divided by 100 to better fit the plot and is only given for comparison.

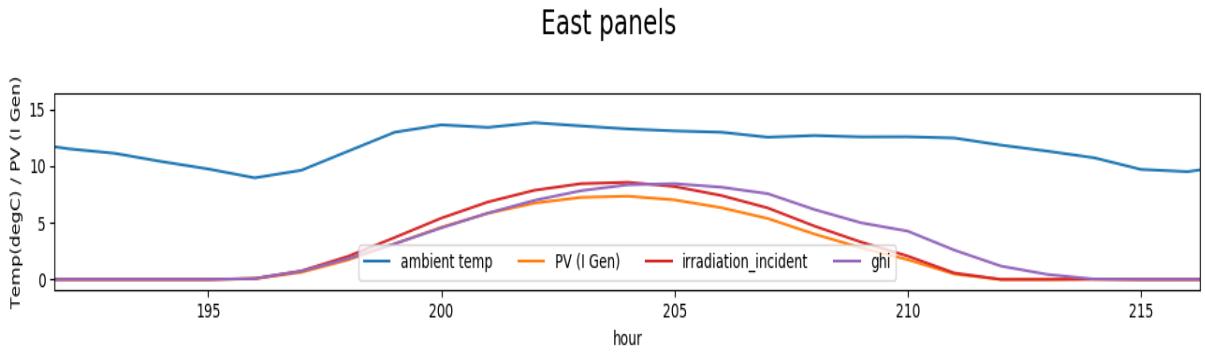


Figure 36: Calculated generated current on east string

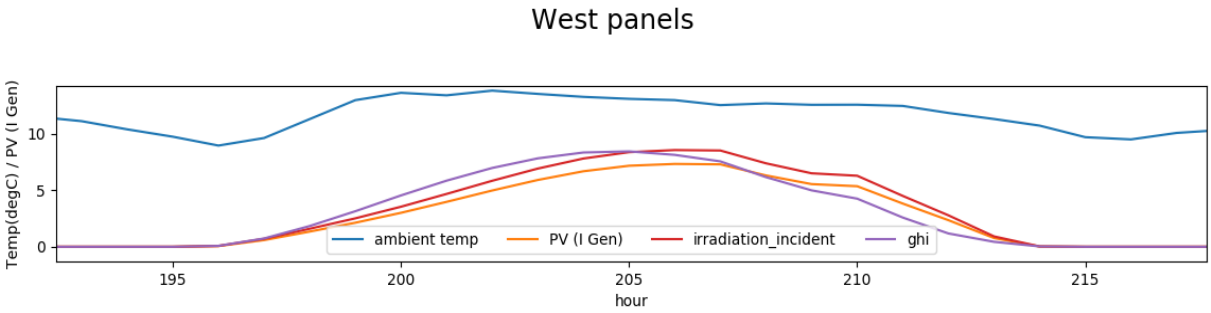


Figure 37: Calculated generated current on west string

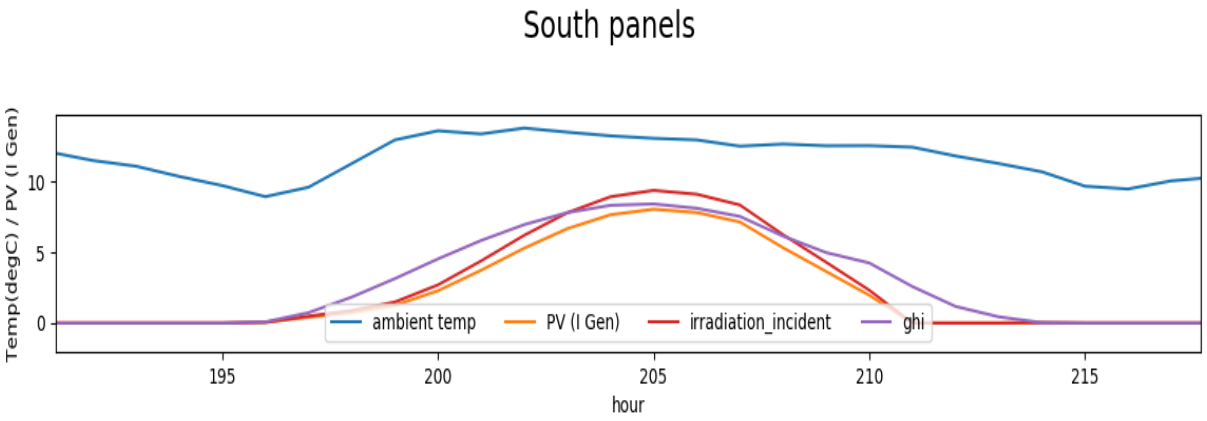


Figure 38: Calculated generated current on south string

4.6.2 Linear optimization algorithm

The linear model used is a modified version of a similar optimization algorithm made by Edward Barbour with his permission which is included in appendix.

The goal of the modified algorithm is to minimize electricity cost while also having the ability to work in a way that can increase the expected battery life based on the theory in this thesis. Because of the nature of linear programming the dispatch strategy knows all the upcoming values from PV generation to load demand meaning that it has "perfect foresight" in much the same way as HOMER grid. The main difference is that the Python algorithm can look ahead indefinitely whereas HOMER grid optimizes with a 48 hour outlook window as described earlier.

The algorithm uses the net demand as a starting point for the optimization, which means that the PV power production is subtracted from the power demand. For time steps with a higher demand than PV generation, *posLoad*, is created. When the opposite is true this value is added to another list, *negLoad*. This makes it so that the algorithm "knows" when there is an excess of renewable energy and responds accordingly.

The object function, F_{obj}, which is to be minimized is as follows (base function = green, SOC incentive = blue, Throughput incentive = yellow):

$$F_{obj} = \sum_0^t (C_{buy}(t) * posNetLoad(t) + C_{sell}(t) * negNetLoad(t)) + w_1 * E_n(t) + w_2 * (EInGrid(t) + EInPV(t) - negEOutLocal(t) - posEOutLocal(t))) + P_{max} * C_p \quad (4.3)$$

Where: C_{buy} is the utility power price for each time step associated with the positive net load, *posNetLoad*. C_{sell} is the sell price for each time step associated with the negative net load, *negNetLoad*. C_p is the power tariff associated with the maximum net demand, P_{max}. w_1 is a weight value associated with the energy content of the battery, E_n, and w_2 is the weighting value associated with total throughput of the battery.

To solve the system of equations the open-source solver GLPK [70] is used. This solver primarily uses the primal or dual simplex model for linear problems, which are versions of the Simplex method described in 2.4.3.

4.6.2.1 Variables

The algorithm takes the following variables where negative values are marked with *neg* and positive values are marked with *pos*:

E_n - Stored energy at each time step

posDeltaSOC – Positive change in stored energy

negDeltaSOC – Negative change in stored energy

EInGrid – Energy into battery from the grid

EInPV – Energy into battery from PV system

negEOutLocal – Energy discharged for local use

negEOutExport – Energy discharged for export

posNetLoad – The total load that is evaluated with regards to the buy price

negNetLoad – The total load that is evaluated with regards to sell price

P_{max} – The maximum net demand for the simulated period

4.6.2.2 Constraints and rules

A set of constraints and rules is introduced to keep the system working within its physical limitations while obeying defined strategies.

To cause less strain on the battery it is kept from discharging completely with the following constraint:

$$0.2 * 48\ kWh \leq E_n \leq 48\ kWh \quad (4.4)$$

The energy charged into the battery is limited to 1C or 48 kW for both grid and PV charging:

$$0 \leq EInGrid \leq 48\ kW \quad (4.5)$$

$$0 \leq EInPV \leq 48\ kW \quad (4.6)$$

The energy stored in the battery is calculated each time step based on the sum of energy charged and discharged from the battery. Due to the nature of this problem, it is never beneficial to charge and discharge at the same time. If this were the case a Boolean variable would have needed to be introduced. The constraint is as follows:

$$E_n(t) = E_n(t-1) + posDeltaSOC(t) + negDeltaSoc(t) \quad (4.7)$$

To make the algorithm find the appropriate grid demand limit a variable, *Pmax*, is introduced which must be greater than or equal to the net load in each time step where PV power generation is less than the demand, *posNetLoad*:

$$Pmax \geq posNetLoad(t) \quad (4.8)$$

Two rules are created to ensure that the total energy in and out of the battery is equal to the sum of all charging variables and discharging variables, respectively. This is also where the battery round trip efficiency, η_{bat}, is included:

$$EInGrid(t) + EInPV(t) == posDeltaSOC/\eta_{bat} \quad (4.9)$$

$$negEOutLocal(t) + negEOutExport(t) == negDeltaSOC(t) * \eta_{bat} \quad (4.10)$$

To ensure that the energy charged into the battery from the PV system does not exceed the available energy, the following constraints is included:

$$EInPV(t) \leq -negLoad(t) \quad (4.11)$$

Applying the same logic, the energy discharged from the battery to be used locally cannot exceed the load demand from the facility:

$$negEOutLocal(t) \geq -posLoad(t) \quad (4.12)$$

Finally, two rules are defined to create a positive and negative net load which is dependent on buy and sell price. These are then used in the object function.

$$posNetLoad(t) == posLoad(t) + EInGrid(t) + negEoutLocal(t) \quad (4.13)$$

$$negNetLoad(t) == negLoad(t) + EInPV(t) + negOutExport(t) \quad (4.14)$$

A flow chart describing the Python code is shown in figure 36.

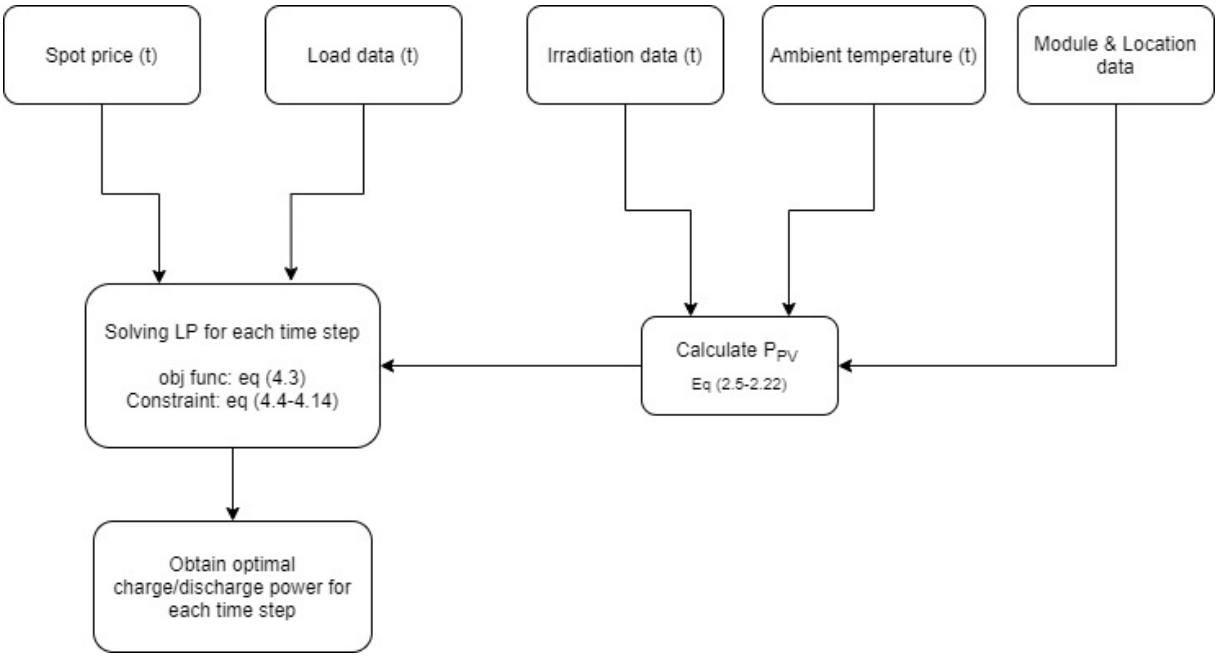


Figure 39: Flowchart describing how the code is built up.

4.6.2.3 System Performance

It is important to know how the system performs in the different scenarios. Self-consumption, P_{self}, is the amount of produced energy that is used locally compared to the total production, $E_{PV,tot}$, and is an important measure to see if the system is well balanced. Cost reduction is the main aim of the LP algorithm and therefore the total cost, C_{tot} is calculated in all simulation scenarios. Number of cycles, N_{cycles}, is also of importance with regards to battery life.

$$P_{self} = 1 - \left(\frac{\sum_1^t -negLoad(t) - EInPV(t)}{E_{PV,tot}}\right) \quad (4.15)$$

Equation 4.15 is not completely accurate in all cases as it assumes that all of the stored PV generated energy is used locally. In the case of this study, it is always more beneficial to use PV generated locally than to sell it, and therefore it is regarded as for this thesis.

$$C_{tot} = P_{max} * C_p + \sum_1^t C_{buy}(t) * posNetLoad + C_{sell} * negNetLoad \quad (4.16)$$

$$N_{cycles} = \sum_1^t EInGrid(t) + EInPV(t) - negEOutExport(t) - negEOutLocal \quad (4.17)$$

4.7 Simulation scenarios

In this chapter different simulation scenarios are discussed to properly understand how changing the parameters of the simulation change the result. In the scenarios where the weighting factor, w_1, is changed, this is done through trial and error to see in what region the best results are obtained. The values with the best results are then used for further simulation. The first scenario uses HOMER and Python to validate the code, and the remaining four uses LP in Python.

4.7.1 Scenario 1: Validating the Python algorithm

To validate the custom made model simulations with the same parameters are made in both HOMER Grid and Python as described in 4.5.2. HOMER Grid simulates over a 25 year period whereas simulations for each month must be made in Python to get peak shaving for each month due to limitations of the code. The system cost will be the same in both simulations, so the main results of interest is power generation, electricity cost and SOC to see if the custom made algorithm using open source tools can compare with an acknowledged commercial tool.

4.7.2 Scenario 2: Peak shaving

Peak shaving aims to minimize the peak power consumption from the grid each month to reduce the power tariff. This is done by utilizing the battery at each point in time where the decided limit is

exceeded. The algorithm aims to find a value which can be obtained with the help of the battery or PV generated power whenever it is needed.

For this scenario simulations are done for a year with and without peak shaving activated. w_1 and w_2 is set to zero. The electricity cost without a battery is also included for comparison. This is done to assess how much the inclusion of a battery and peak shaving impacts the total electricity price. All the other parameters are the same as in the previous scenario.

4.7.3 Scenario 3: Battery life extension by avoiding high SOC

Based on the literature study and theory regarding battery life one of the main factors for reduced battery life is cycling and storage at high state of charge. To address the latter a variable is included to make the algorithm minimize the average SOC for the simulated time period.

For this scenario January and June is simulated with different values for w_1 starting with 1, 0.5, 0.05, 0.005 and 0.0005 to see how this impacts number of cycles, SOC and electricity cost. The entire year is then simulated with the best weight values and compared to the base case. Peak shaving is deactivated and w_2 is set to zero.

4.7.4 Scenario 4: Battery life extension by avoiding high cycling

Another way of increasing the expected battery life is to reduce the amount of cycling. This can be done by including the sum of charging and discharging variables to the object function for the total throughput to be minimized. Peak shaving is deactivated, w_1 is set to zero and w_2 starts as 1, 0.5, 0.05, 0.005 and 0.0005 for the different simulations before the entire year is ran at the best obtained weight value.

4.7.5 Scenario 5: Day ahead scheduling

Simulations are made with data collected from the two APIs. This is done to verify that LP can be used for shorter timeframes and assess how much the cost can be reduced on a daily basis. It is also necessary to see if data collection from the APIs is sufficient for day ahead optimization if used in combination with a physical system. Peak shaving is turned off and the initial charge of the battery is set to 50%.

5. Results and discussion

5.1 Validation and comparison

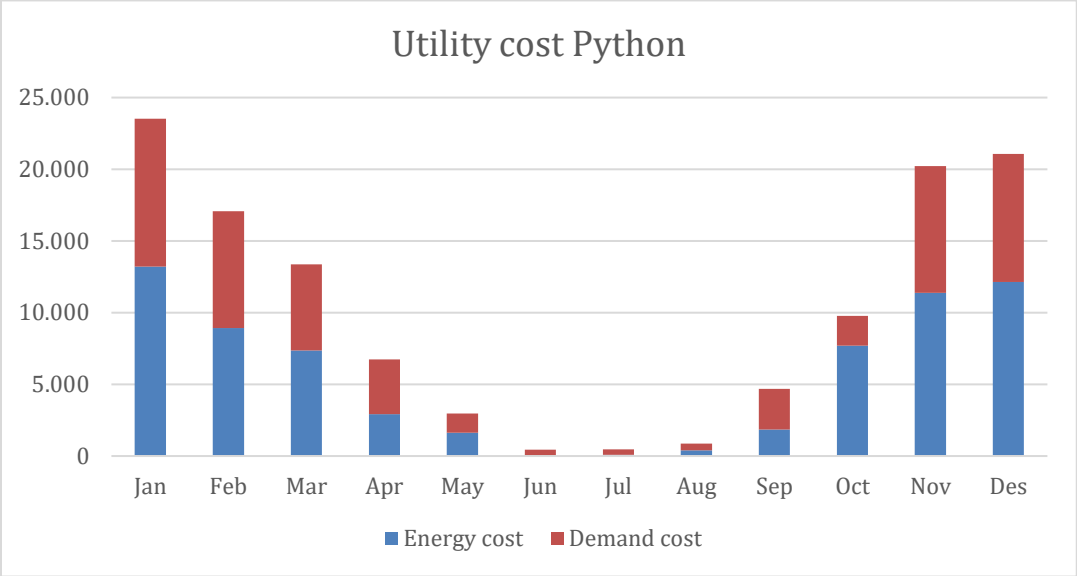


Figure 40: Utility costs from Python simulations

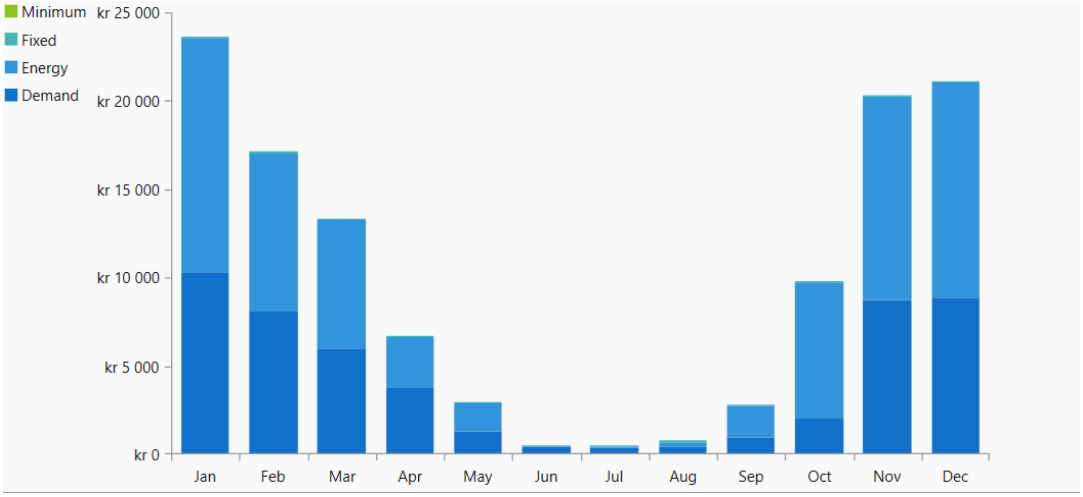


Figure 41: Utility costs from HOMER Grid

From figure 40 and 41 It is clear that the system used in Python and HOMER both result in almost the same utility cost. The total annual electricity cost in Python becomes 121 000 NOK and for the same time period this results in 119 506 NOK for HOMER. This means that the custom linear programming algorithm obtains very similar results with regards to cost savings.

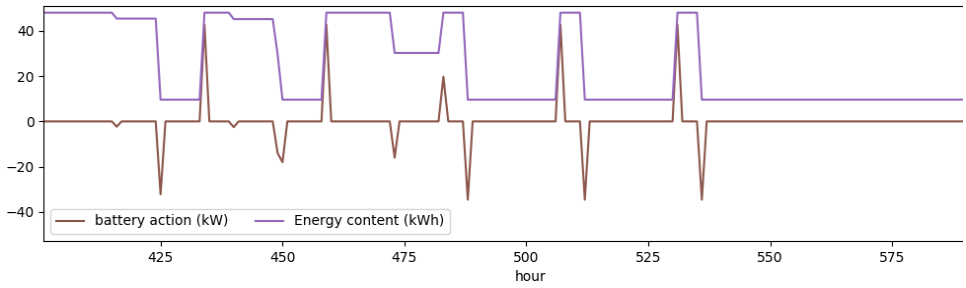


Figure 42: Battery action and energy content in January from Python

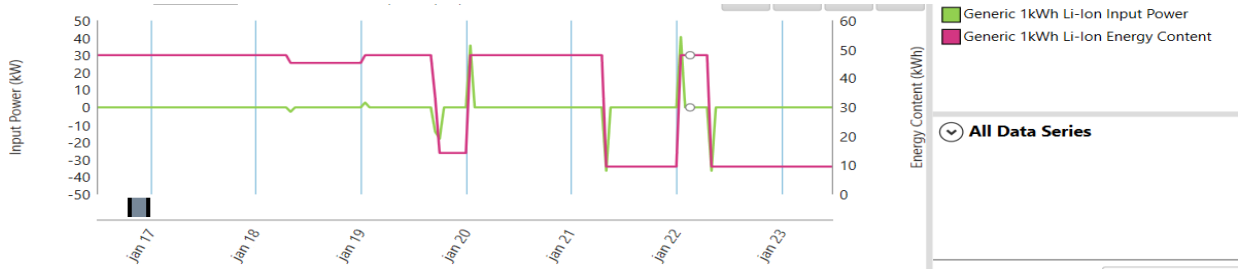


Figure 43: Battery action and energy content in January from HOMER Grid

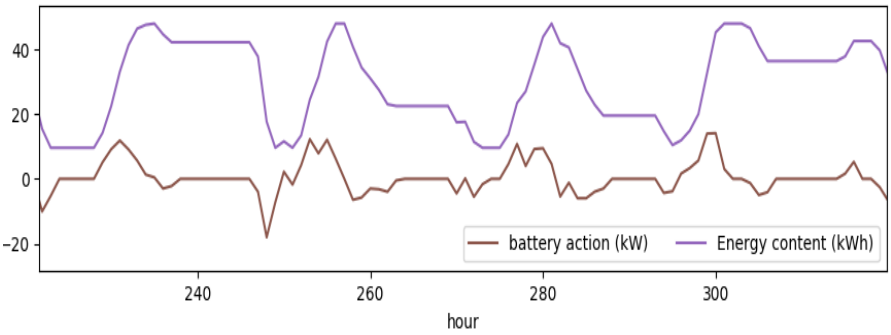


Figure 44: Battery action and energy content in June from Python

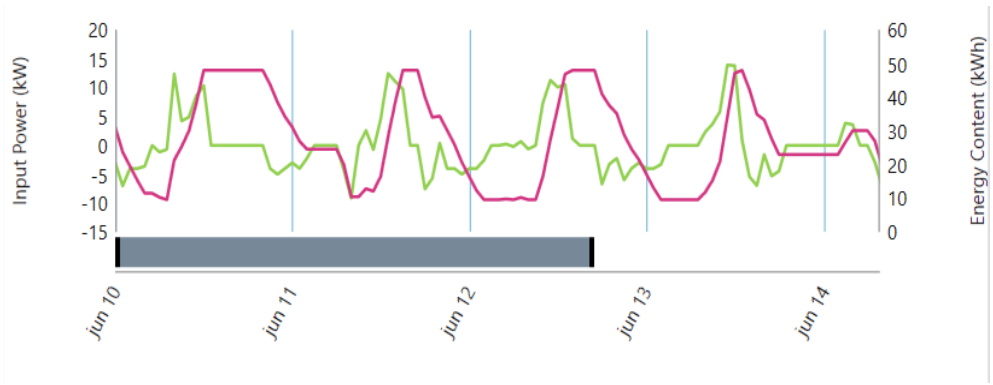


Figure 45: Battery action and energy content in June from HOMER Grid

When looking at the battery behavior during same time periods in both Python and HOMER both follow a similar pattern in both January and June although the LP model tends to use the battery more aggressively while HOMER chooses to keep the battery charged at higher SOC for longer periods of time in January. This correlates with the dispatch strategy of HOMER which favors that energy is stored for longer periods as described in 4.5. Another factor is that HOMER looks ahead only 48 hours while the Python algorithm has no limitation in this regard. HOMER reduces the peak grid demand by 18.53 kW in January whereas Python reduces the peak by 14.35 kW.

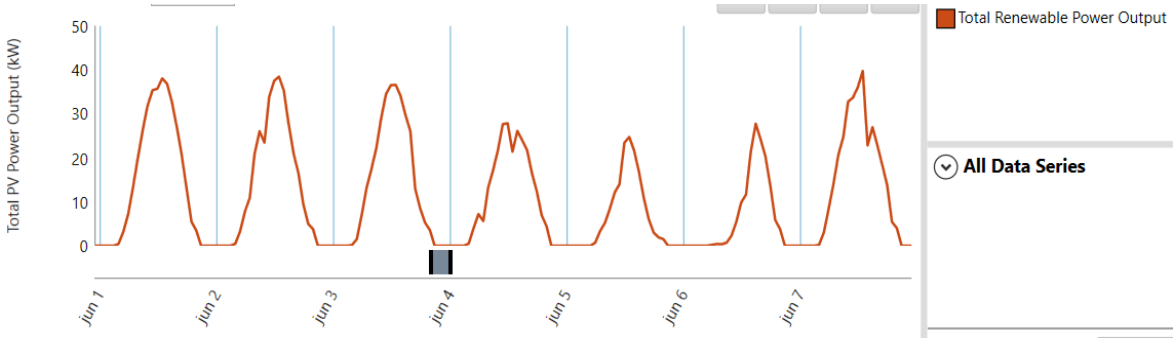


Figure 46: PV power output June HOMER

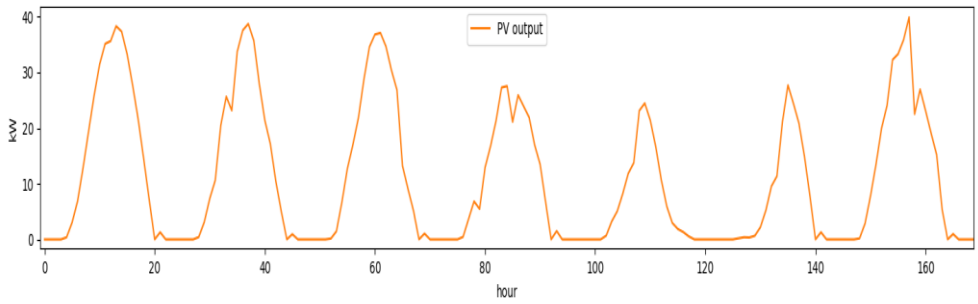


Figure 47: Total PV power output June in Python

Lastly the total power output from both programs is compared for a similar time period. The results are very similar proving that the diode model used in Python offers sufficient accuracy for further simulation. The result from Python has some small deviations during sunset hours, but this could be because of the separation model used as mentioned in 2.2.2.2. The annual renewable generation becomes 53 371 kWh in Python and 53 719 kWh in HOMER.

As we can see the higher electricity price in HOMER might be a result from lower PV production, and less aggressive peak shaving.

5.2 Peak shaving

Table 3: Simulation with and without batteries and peak shaving

Peak shaving	Electricity cost January (NOK)	Electricity cost June (NOK)	Electricity cost Annual (NOK)
Off	25 546	876	137 168
On	23 516	386	121 000
Difference	-7.9 %	-11.87 %	-11.8%
No battery			138 500

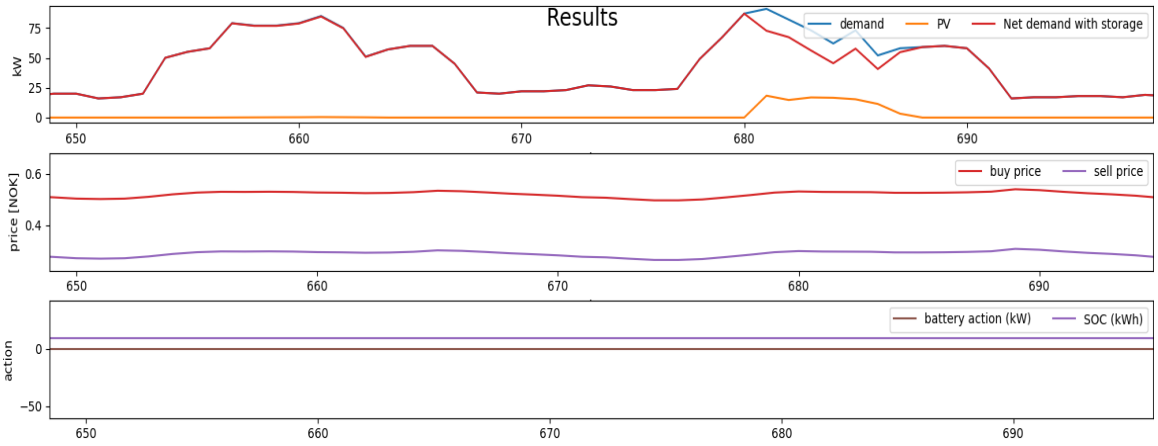


Figure 48: Without peak shaving in January

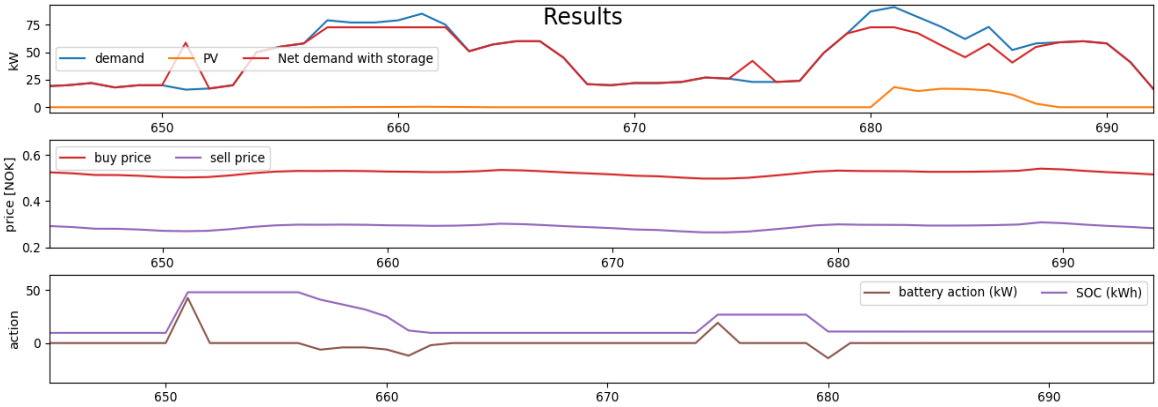


Figure 49: With peak shaving in January

When peak shaving is activated, this results in an annual 16.6% decrease in electricity price. From figure 47 and 48 it is clear that the algorithm chooses to charge the battery to be able to reduce the maximum peak occurring in January when peak shaving is turned on. The peak is reduced by 14.35 kW and for this to be possible all peaks exceeding this limit must be reduced to the same value as can be seen in figure 48 by the decrease in net demand.

5.3 SOC as incentive

Table 4: Results from simulating January and June with different weight values for SOC incentive. The best option is marked in green.

W_1	Electricity cost January (NOK)	Average SOC January (%)	Cycles January	Self-consumption January (%)	Electricity cost June (NOK)	Average SOC June (%)	Cycles June	Self-consumption June (%)
0	25 546	70	11.8	100	876	55	44.5	73.5
1	25 555	20	3.7	100	1243	20	2.8	60
0.5	25 555	20	3.7	100	1187	20.5	3.8	60
0.05	25 554	20	3.7	100	1007	23	14.5	62.4
0.005	25 550	22.3	8.5	100	883	44.7	45.4	73.5
0.0005	25 548	28	10.9	100	876.9	44.7	45.4	73.5

Table 5: Annual simulations with the best weighting value from the previous simulations and no weighting for comparison.

W_1	Electricity cost Annual (NOK)	Average SOC Annual (%)	Cycles Annual	Average self-consumption annual (%)
0	137 168	55.68	294	87
0.0005	137 251	36.31	260	89.2
Difference	~ 0%	-34.8%	-11.6%	+2.5%

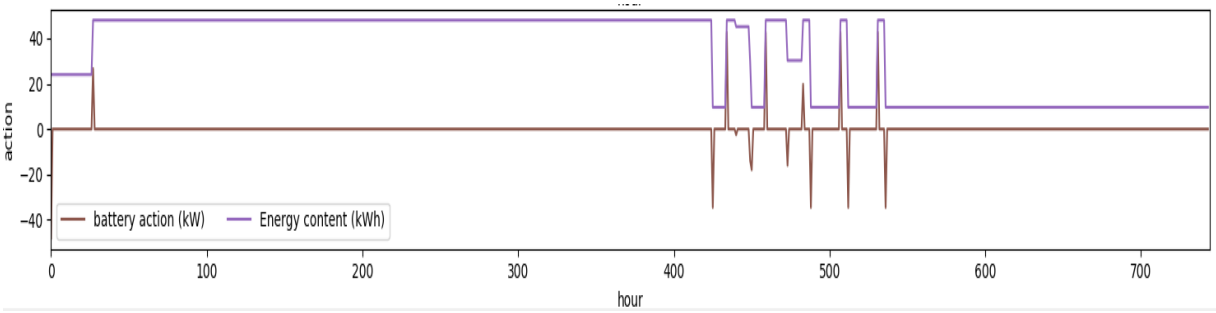


Figure 50: Battery action and SOC with no weighting on SOC in January

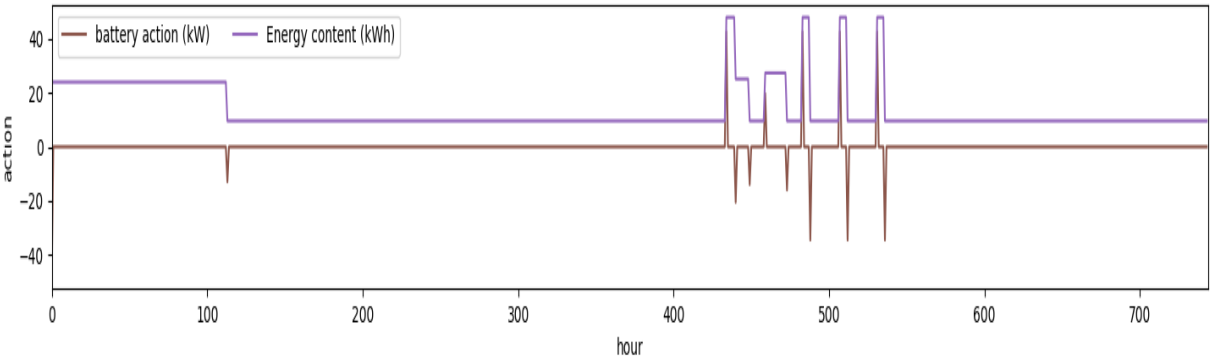


Figure 51: Battery action and SOC with 0.0005 as the weight value on SOC in January

By introducing the sum of stored energy to the object function the algorithm tries to minimize the amount of energy stored at any period in time unless it is economically beneficial to do otherwise. As we can see from table 4, January has its average SOC reduced by 60%, with only a small increase in electricity cost. For June, the average SOC is reduced by 18.1 %.

This incentive proves most beneficial for periods with long time periods where the battery is unused, but it does help during periods with more cycling as well. The annual average SOC is reduced by 34.8% when using the most optimal weighting factor for annual simulation. The self-consumption improves due to more available capacity for when excess PV energy is available.

5.4 Throughput as incentive

Table 6: Results from simulating January and June with different weight values for cycling incentive. The best option is marked in green.

W_2	Electricity cost January (NOK)	Average SOC January (%)	Cycles January	Self-consumption January (%)	Electricity cost June (NOK)	Average SOC June (%)	Cycles June	Self-consumption June (%)
0	23 516	70	11.8	100	385	55	44.5	73.5
1	23 537	87.8	2.1	100	656	42.8	2.85	60.8

0.5	23 530	87.8	2.4	100	526	47.2	6.65	61.7
0.075	23 521	87.8	3.1	100	458	63.1	21.9	66.3
0.05	23 521	87.8	3.1	100	386	55.3	44.5	73.5
0.005	23 516	71.3	10.16	100	385.8	55.3	44.48	73.5
0.0005	23 516	71.3	10.3	100	385.8	55.3	44.48	73.5

Table 7: Annual simulations with the best weighting value from the previous simulations and no weighting for comparison.

W_2	**Electricity cost Annual (NOK)**	**Average SOC Annual (%)**	**Cycles Annual**	**Average self-consumption annual (%)**
0	137 168	55.68	294	87
0.075	137 304	58.38	159	85
Difference	~ 0%	+4.8%	-45.9%	-2.3%

Introducing throughput as an incentive causes the total cycles in January and June to be reduced by 73.7% and 50.7% respectively. Annual simulation with 0.075 as weight factor results in a cycle reduction of 45.9%. Self-consumption is impacted during the months with large amounts of PV power output as the battery use is minimized, causing more of the generated energy not utilized by the load to be exported directly rather than being stored for future use. The average annual self-consumption is reduced by 2.3% and the electricity cost increases by 3.3%.

As discussed in the case study the battery should be capable of minimum 2000 cycles before reaching its end of life. Without including the throughput incentive this means that the battery lasts a minimum of 6.8 years. By implementing the incentive and sacrificing some self-consumption the minimum battery life should reach 12.58 years. To counteract this positive change the average SOC does increase.

5.5 Day ahead simulation

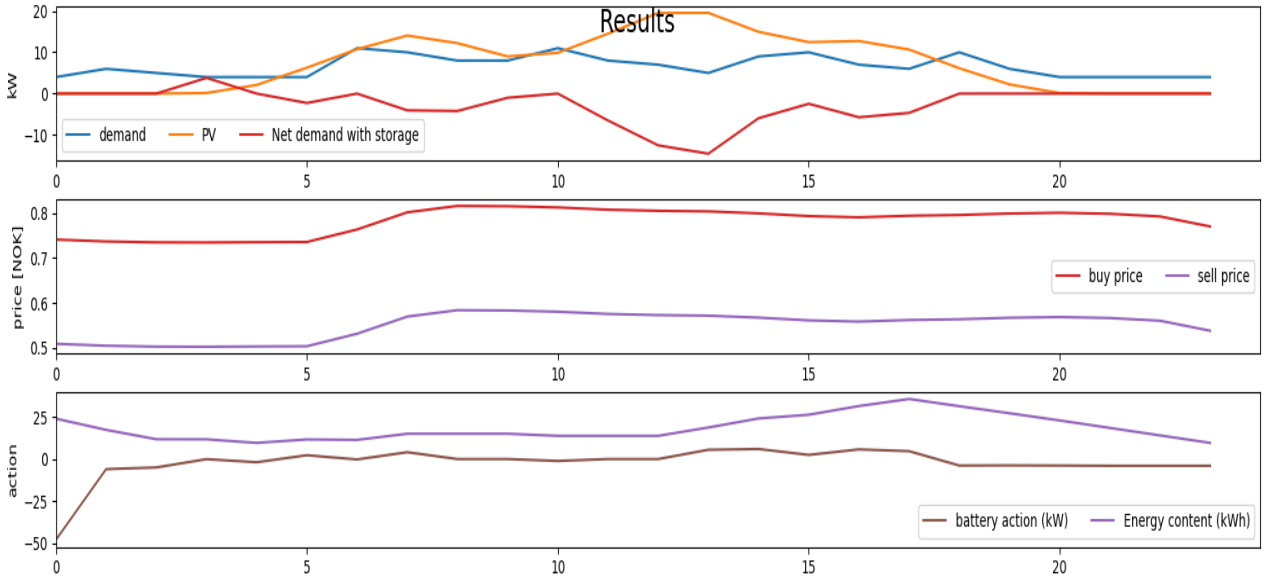


Figure 52: Day ahead simulation for 27.05.2021

The day ahead simulation uses forecasted weather and spot price data for 27.05.21 and load data from the historic dataset with a corresponding load signature to the month simulated. As we can see from figure 49, the LP program discharges the battery from its initial energy content to meet the load demand in the morning. As the PV production is greater than the load demand most of the day it is beneficial to use the battery and charge it with surplus PV generated energy. The stored energy is then used to meet the load demand after sunset.

Table 8: Key values from day ahead simulation

Electricity cost (NOK)	**Average SOC (%)**	**Cycles**	**Self-consumption (%)**
-1.32	0.39	2.42	81.2

It is clear that the code made for day ahead optimization works well, and further improvements and implementation with a physical setup could give good results. A limitation of the day ahead optimization is the fact that electricity prices are not available more than 24 hours ahead. By looking 48 hours ahead for example, the algorithm would not necessarily use the battery to meet load demand whenever possible as is the case here.

5.6 Comparison and general discussion

Table 9: Comparison of scenarios with peak shaving activated

Scenario	Electricity cost Annual (NOK)	Average SOC Annual (%)	Cycles Annual	Average self-consumption annual (%)
2: Peak shaving	121 000	55.68	294	87
3: SOC reduction	121 083	30.61	260	89.2
4: Cycle reduction	121 136	58.38	159	85

When comparing the results from the different scenarios it is clear that the system response can be changed depending on the desired outcome by including additional incentives without a large impact on electricity cost.

Peak shaving with no other incentives as in scenario 2 proves to be the best option with regards to electricity cost savings. The average SOC is also not that high, but as seen in 5.3 it can be large for some months with low cycling. The total price reduction each year is 17 500 compared to having no battery as seen in table 3. If the cost of the battery is approximately 100 000 as per 4.5.1 this would lead to a payback time of 5.7 years which is within the expected minimum battery life of 6.8 years as per 5.4. As peak shaving is an important factor when discussing electricity cost reduction when on a power tariff this is kept on during the other scenarios.

Including an incentive for reducing the energy stored in the battery in scenario 3 leads to best self-consumption and a lower average SOC which is important for increased battery storage life. It does however also result in an increased electricity price as periods with low energy cost could mean unused saving potential as the model values limiting energy storage over charging from the grid. This scenario leads to a payback time of 5.74 years with an expected minimum battery life of 7.7 years.

Finally, the inclusion of an incentive for battery throughput in scenario 4 leads to a 46% reduction in cycles while keeping the electricity cost low. This scenario leads to a payback time of 5.76 years with an estimated battery life of 12.58 years. This makes scenario 4 the most profitable over time as the battery lasts longer before having to be replaced. The main drawback being a reduction in self-consumption as generated energy is sold to the grid rather than being stored in the battery unless the cost savings are big enough.

It is clear that periods with different amounts of PV power production benefit from different incentives and weighting. In hindsight it would have been interesting and beneficial to simulate the system with a combination of incentives and weights for each month to see if further improvements could be made with regards to balancing battery prevention and cost savings. Day ahead optimization could also benefit from simulations with different incentives as the limited foresight of 24 hours causes the battery to be utilized whenever load demand can be met instead of planning ahead.

6. Conclusion

The main aim of this thesis is to evaluate how well linear programming (LP) can function as a tool for battery energy scheduling optimization in a hybrid system. When comparing the custom made LP algorithm with the commercial tool HOMER it is clear that the LP algorithm correctly optimizes the system to minimize electricity cost. Peak shaving works well and reduces the annual electricity cost by 11.8%.

LP also makes it possible to include specific constraints, rules and incentives to the objective function which allows for a great amount of customization with regards to discharge strategy. It was found that a small weight on stored energy content of the battery greatly reduces the average SOC, especially during month with low amounts of PV generation.

When including battery energy throughput as an incentive this was found to greatly reduce the number of cycles without a large increase in electricity cost. As battery life is rather closely related to the number of cycles the results found in this thesis means that such an incentive can substantially improve battery life.

It is clear from day ahead simulations that LP optimization works well with forecasted data collected from APIs. The main limitation is reduced foresight of only 24 hours as this could reduce the overall system performance over time.

When optimizing a hybrid system care should be taken when load and power generation data is modelled as this greatly impacts the optimal scheduling. In the case of this study the implementation of advanced incident irradiation calculations and the single diode model allows the Python code to calculate the total generated PV output at each time step before running the optimization algorithm. The results are similar to what is calculated by commercial tools and is therefore considered to be sufficient.

When designing a LP optimization algorithm, it is important to understand how different incentives impact the total outcome of any given system. Care should be taken when choosing weighting factors as this greatly impacts the system response. When simulating with different weights it became clear that too high values caused the battery to cost more money than it saved excluding peak shaving.

From the simulations made in this thesis it is clear that proper system modelling and LP design can result in cost savings and improved battery operation.

7. Recommendations and future work

It would be beneficial for a system described in the thesis to implement more realistic loads for day ahead scheduling. This could also make it possible to do “real time” peak shaving if the load exceeds a value deemed to be higher than the monthly estimated average.

Day ahead optimizing for a longer period would make it possible to assess how different incentives could impact the cost savings and battery prevention. It would also be interesting to implement a two-step process including both day ahead and hour ahead optimization.

Creating a model based on an existing system would make it possible to verify if the PV model combined with forecasted irradiation data offer accurate results, and if the LP algorithm works in combination with a physical system.

More accurate battery modelling could be beneficial to create optimization results which better suit the given parameters of a physical system.

It would be very interesting to compare the linear approach in this thesis to other commonly used methods such as neural networks or fuzzy logic controllers. This would also allow for non-linear simulation which could include accurate battery life models to further understand how different discharge strategies impact battery life.

List of References

[1] "Global Warming vs. Climate Change | Resources – Climate Change: Vital Signs of the Planet." [Online]. Available: https://climate.nasa.gov/resources/global-warming-vs-climate-change/. [Accessed: 04-Feb-2021].
[2] "Climate change and health." [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/climate-change-and-health. [Accessed: 05-Feb-2021].
[3] "The Paris Agreement | UNFCCC." [Online]. Available: https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement. [Accessed: 05-Feb-2021].
[4] "Electricity Information 2019 – Analysis - IEA." [Online]. Available: https://www.iea.org/reports/electricity-information-overview. [Accessed: 08-Feb-2021].
[5] "Kraftproduksjon - NVE." [Online]. Available: https://www.nve.no/energiforsyning/kraftproduksjon/. [Accessed: 09-Feb-2021].
[6] "Samlet energibruk - NVE." [Online]. Available: https://www.nve.no/energibruk-effektivisering-og-teknologier/energibruk/samlet-energibruk/?ref=mainmenu. [Accessed: 09-Feb-2021].
[7] "Renewable Power Generation Costs in 2018," */publications/2019/May/Renewable-power-generation-costs-in-2018*.
[8] "Renewable Power Generation Costs in 2019," */publications/2020/Jun/Renewable-Power-Costs-in-2019*.
[9] "More of a good thing – is surplus renewable electricity an opportunity for early decarbonisation? – Analysis - IEA." [Online]. Available: https://www.iea.org/commentaries/more-of-a-good-thing-is-surplus-renewable-electricity-an-opportunity-for-early-decarbonisation. [Accessed: 10-Feb-2021].
[10] "Energy Storage – Analysis - IEA." [Online]. Available: https://www.iea.org/reports/energy-storage. [Accessed: 11-Feb-2021].
[11] "How Inexpensive Must Energy Storage Be for Utilities to Switch to 100 Percent Renewables? - IEEE Spectrum." [Online]. Available: https://spectrum.ieee.org/energywise/energy/renewables/what-energy-storage-would-have-to-cost-for-a-renewable-grid. [Accessed: 11-Feb-2021].
[12] "Electricity Storage Technology Review Prepared for Electricity Storage Technology Review," 2020.
[13] "Battery Pack Prices Cited Below $100/kWh for the First Time in 2020, While Market Average Sits at $137/kWh | BloombergNEF." [Online]. Available: https://about.bnef.com/blog/battery-pack-prices-cited-below-100-kwh-for-the-first-time-in-2020-while-market-average-sits-at-137-kwh/. [Accessed: 11-Feb-2021].
[14] "Global EV Outlook 2020 – Analysis - IEA." [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2020. [Accessed: 11-Feb-2021].
[15] "Elbilsalget vokser til nye rekordnivåer." [Online]. Available: https://motor.no/audi-bilsalget-elbil/elbilsalget-vokser-til-nye-rekordnivaer/189219. [Accessed: 11-Feb-2021].
[16] "Design and Analysis of Large Lithium-Ion Battery Systems." [Online]. Available: http://web.b.ebscohost.com/ehost/ebookviewer/ebook/ZTAwMHh3d19fMTE1NTE5OF9fQU41?sid=0cdbd7ae-497c-4d77-a971-d49d482f36ff@sessionmgr101&vid=0&format=EB&rid=1. [Accessed: 07-May-2021].
[17] "How to Charge Li-ion with a Parasitic Load – Battery University." [Online]. Available: https://batteryuniversity.com/learn/article/how_to_charge_li_ion_with_a_parasitic_load. [Accessed: 08-May-2021].
[18] N. Noura, L. Boulon, and S. Jemeï, "A review of battery state of health estimation methods: Hybrid electric vehicle challenges," *World Electr. Veh. J.*, vol. 11, no. 4, pp. 1–20, 2020, doi: 10.3390/wevj11040066.
[19] W.-Y. Chang, "The State of Charge Estimating Methods for Battery: A Review," *ISRN Appl.*

Math., vol. 2013, pp. 1–7, Jul. 2013, doi: 10.1155/2013/953792.
[20] Mertens konrad, *Photovoltaics 2nd edition*. 219AD.
[21] A. Wang, S. Kadam, H. Li, S. Shi, and Y. Qi, "Review on modeling of the anode solid electrolyte interphase (SEI) for lithium-ion batteries," *npj Computational Materials*, vol. 4, no. 1. Nature Publishing Group, p. 15, 01-Dec-2018, doi: 10.1038/s41524-018-0064-0.
[22] "Battery Management Systems for Large Lithium-ion Battery Packs." [Online]. Available: http://web.a.ebscohost.com/ehost/ebookviewer/ebook/ZTAwMHh3d19fMzQ1Njg5X19BTg2?sid=8223bdb8-e558-4791-92a7-8341900ee822%40sessionmgr4007&vid=0&format=EB&rid=1. [Accessed: 09-May-2021].
[23] "Solar cell efficiencies graph." [Online]. Available: https://www.nrel.gov/pv/assets/pdfs/best-research-cell-efficiencies.20200104.pdf. [Accessed: 10-May-2021].
[24] "Albedo Measurements." [Online]. Available: http://www.climatedata.info/forcing/albedo/. [Accessed: 21-May-2021].
[25] A. J. Duffie and A. W. Beckman, *Solar engineering of thermal processes*, 4th ed. 2013.
[26] D. G. Erbs, S. A. Klein, and J. A. Duffie, "Estimation of the diffuse radiation fraction for hourly, daily and monthly-average global radiation," *Sol. Energy*, vol. 28, no. 4, pp. 293–302, Jan. 1982, doi: 10.1016/0038-092X(82)90302-4.
[27] "How HOMER Calculates the PV Array Power Output." [Online]. Available: https://www.homerenergy.com/products/pro/docs/latest/how_homer_calculates_the_pv_array_power_output.html. [Accessed: 13-May-2021].
[28] M. Power, "Technical Drawings SOLAR CELLS POLY-CRYSTALLINE 156 × 156 MM 72 PCS. (6×12)-4 BUS BARS."
[29] "What is Maximum Power Point Tracking (MPPT) | Northern Arizona Wind & Sun." [Online]. Available: https://www.solar-electric.com/learning-center/mppt-solar-charge-controllers.html/. [Accessed: 12-Dec-2020].
[30] "PV Derating Factor." [Online]. Available: https://www.homerenergy.com/products/pro/docs/latest/pv_derating_factor.html. [Accessed: 16-May-2021].
[31] "EasySolar 24/1600/40-16 MPPT 100/50 - Kr 16995." [Online]. Available: https://www.sparelys.no/easysolar-24-1600-40-16-mppt-100-50?gclid=Cj0KCQjw16KFBhCgARIsALB0g8J9uUeFvOQ4go-Lea4xN8OaX8vrFImf2lSjoCrFFj0rUNPE79T-vv0aAoAYEALw_wcB. [Accessed: 22-May-2021].
[32] "Norway and the European power market - NVE." [Online]. Available: https://www.nve.no/norwegian-energy-regulatory-authority/wholesale-market/norway-and-the-european-power-market/. [Accessed: 12-May-2021].
[33] R. W. Cottle and M. N. Thapa, *Linear and Nonlinear Optimization*. 2017.
[34] "What is an API? | (API) Application Program Interface Definition." [Online]. Available: https://apifriends.com/api-management/what-is-an-api/. [Accessed: 13-May-2021].
[35] N. A. Engerer and F. P. Mills, "Validating nine clear sky radiation models in Australia," *Sol. Energy*, vol. 120, pp. 9–24, Oct. 2015, doi: 10.1016/j.solener.2015.06.044.
[36] "How we estimate and use satellite, aerosol and weather sources to make our data." [Online]. Available: https://solcast.com/solar-radiation-data/inputs-and-algorithms/. [Accessed: 13-May-2021].
[37] J. Abushnaf and A. Rassau, "Impact of energy management system on the sizing of a grid-connected PV/Battery system," *Electr. J.*, vol. 31, no. 2, pp. 58–66, Mar. 2018, doi: 10.1016/j.tej.2018.02.009.
[38] S. Tong, T. Fung, M. P. Klein, D. A. Weisbach, and J. W. Park, "Demonstration of reusing electric vehicle battery for solar energy storage and demand side management," *J. Energy Storage*, vol. 11, pp. 200–210, Jun. 2017, doi: 10.1016/j.est.2017.03.003.
[39] S. Li, H. He, Y. Chen, M. Huang, and C. Hu, "Optimization between the PV and the Retired EV Battery for the Residential Microgrid Application," in *Energy Procedia*, 2015, vol. 75, pp. 1138–1146, doi: 10.1016/j.egypro.2015.07.537.

[40] G. S. Georgiou, P. Christodoulides, and S. A. Kalogirou, "Optimizing the energy storage schedule of a battery in a PV grid-connected nZEB using linear programming," *Energy*, vol. 208, p. 118177, Oct. 2020, doi: 10.1016/j.energy.2020.118177.
[41] M. Faisal, M. A. Hannan, P. J. Ker, M. S. A. Rahman, R. A. Begum, and T. M. I. Mahlia, "Particle swarm optimised fuzzy controller for charging–discharging and scheduling of battery energy storage system in MG applications," *Energy Reports*, vol. 6, pp. 215–228, Dec. 2020, doi: 10.1016/j.egyr.2020.12.007.
[42] G. I. Marinova and V. G. Guliashki, "Optimization of the Battery Schedule for Residential Microgrid Applications," *IFAC-PapersOnLine*, vol. 49, no. 29, pp. 226–231, Jan. 2016, doi: 10.1016/j.ifacol.2016.11.055.
[43] M. Gitizadeh and H. Fakharzadegan, "Battery capacity determination with respect to optimized energy dispatch schedule in grid-connected photovoltaic (PV) systems," *Energy*, vol. 65, pp. 665–674, Feb. 2014, doi: 10.1016/j.energy.2013.12.018.
[44] R. Hanna, J. Kleissl, A. Nottrott, and M. Ferry, "Energy dispatch schedule optimization for demand charge reduction using a photovoltaic-battery storage system with solar forecasting," *Sol. Energy*, vol. 103, pp. 269–287, May 2014, doi: 10.1016/j.solener.2014.02.020.
[45] M. Taslimi, P. Ahmadi, M. Ashjaee, and M. A. Rosen, "Design and mixed integer linear programming optimization of a solar/battery based Conex for remote areas and various climate zones," *Sustain. Energy Technol. Assessments*, vol. 45, p. 101104, Jun. 2021, doi: 10.1016/j.seta.2021.101104.
[46] B. C. Jeong, D. H. Shin, J. B. Im, J. Y. Park, and Y. J. Kim, "Implementation of optimal two-stage scheduling of energy storage system based on big-data-driven forecasting - An actual case study in a campus microgrid," *Energies*, vol. 12, no. 6, 2019, doi: 10.3390/en12061124.
[47] "Introduction to Genetic Algorithms — Including Example Code | by Vijini Mallawaarachchi | Towards Data Science." [Online]. Available: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3. [Accessed: 30-Mar-2021].
[48] X. S. Hu, F. C. Sun, and Y. Zou, "Online model identification of lithium-ion battery for electric vehicles," *J. Cent. South Univ. Technol. (English Ed.*, vol. 18, no. 5, pp. 1525–1531, Oct. 2011, doi: 10.1007/s11771-011-0869-1.
[49] "Fuzzy Logic Definition." [Online]. Available: https://www.investopedia.com/terms/f/fuzzy-logic.asp. [Accessed: 30-Mar-2021].
[50] E. Wikner and T. Thiringer, "Extending battery lifetime by avoiding high SOC," *Appl. Sci.*, vol. 8, no. 10, Oct. 2018, doi: 10.3390/app8101825.
[51] S. Gantenbein, M. Schönleber, M. Weiss, and E. Ivers-Tiffée, "Capacity fade in lithium-ion batteries and cyclic aging over various state-of-charge ranges," *Sustain.*, vol. 11, no. 23, pp. 1–15, 2019, doi: 10.3390/su11236697.
[52] B. Aigner, "Analyzing the implementation of second use batteries as storage solution for solar power at local waste station using HOMER pro and HOMER grid," 2020.
[53] "Historical Market Data | Nord Pool." [Online]. Available: https://www.nordpoolgroup.com/historical-market-data/. [Accessed: 17-May-2021].
[54] "Priser | Agder Energi." [Online]. Available: https://www.aenett.no/kundeforhold/kundebetingelser/kundebetingelser-privatkunde/tariffer/. [Accessed: 07-Dec-2020].
[55] "ENTSO-E Transparency Platform." [Online]. Available: https://transparency.entsoe.eu/. [Accessed: 18-May-2021].
[56] Solcast, "Solar irradiadiance data," 2021. [Online]. Available: https://solcast.com/. [Accessed: 18-May-2021].
[57] "Free solar irradiation historical and forecasting data for researchers." [Online]. Available: https://solcast.com/solar-data-api/free-solar-radiation-historical-and-forecasting-data-for-researchers/. [Accessed: 18-May-2021].
[58] "API | Nord Pool." [Online]. Available: https://www.nordpoolgroup.com/trading/api/. [Accessed: 21-May-2021].
[59] "Optimize your renewable hybrid power system," 2020.

[60] "Optimizing microgrid design," 2020.
[61] "Leverage powerful optimization, proven results," 2020.
[62] "HOMER Grid Dispatch Strategy." [Online]. Available: https://www.homerenergy.com/products/grid/docs/1.1/homer_grid_dispatch_strategy.html. [Accessed: 26-May-2021].
[63] "Victron QUATRRO 48V 10000VA 140A - kombi inverter og 140A lader – Strømkilden.no - Lag din egen strøm." [Online]. Available: https://stromkilden.no/products/quatrro-48v-10000va-140a?variant=31606235267149&currency=NOK&utm_medium=product_sync&utm_source=google&utm_content=sag_organic&utm_campaign=sag_organic&utm_campaign=gs-2019-10-21&utm_source=google&utm_medium=cpc&gclid=CjwKCAiAq8f-BRBtEiwAGr3DgXkB0KRAiRKhmnA0ceYjdHoIuIs4nlh_a-E8IWvK_LVCcOrQL-3GZBoC_KMQAvD_BwE. [Accessed: 10-Dec-2020].
[64] "Morningstar TS-MPPT-60-600V-48-DB - Kr 21800." [Online]. Available: https://www.sparelys.no/morningstar-ts-mppt-60-600v-48-db?gclid=CjwKCAjw-qeFBhAsEiwA2G7Nl39xSx1Br8MjcTyhvE0hK2cS9Vw-RfOWQJMwoHHXGEB4oujhwYYjrhoCz4oQAvD_BwE. [Accessed: 23-May-2021].
[65] "Photovoltaic Panels (PV)." [Online]. Available: https://www.homerenergy.com/products/pro/docs/latest/photovoltaic_panels_pv.html. [Accessed: 12-Dec-2020].
[66] "69. INSIDE THE BATTERY OF A NISSAN LEAF - Qnovo." [Online]. Available: https://qnovo.com/inside-the-battery-of-a-nissan-leaf/. [Accessed: 23-May-2021].
[67] "About — Pyomo." [Online]. Available: http://www.pyomo.org/about. [Accessed: 19-May-2021].
[68] "Optimization and root finding (scipy.optimize) — SciPy v1.6.3 Reference Guide." [Online]. Available: https://docs.scipy.org/doc/scipy/reference/optimize.html. [Accessed: 20-May-2021].
[69] "requests · PyPI." [Online]. Available: https://pypi.org/project/requests/. [Accessed: 26-May-2021].
[70] "GLPK for MPL." [Online]. Available: http://www.maximalsoftware.com/solvers/glpk.html. [Accessed: 20-May-2021].

Appendix

CODE

```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 22 12:59:25 2021

@author: Birk Aigner
"""

# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a code which is designed for optimal battery energy scheduling and
PV power production modelling.

Author: Birk Aigner
Year: 2021

"""
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import time
import math
from scipy.optimize import fsolve
import requests
import json
from datetime import datetime, timedelta

sin = math.sin
cos = math.cos

#Period to simulate
period = "jan" #Can be set to each of the 12 months (jan,feb,mar,apr,may
...), DayAhead or year


#Importing datasets
weather_year = pd.read_excel("2020_weather_UiA.xlsx")
load = pd.read_excel("Lastprofil.xlsx", "Effektuttak Mjåvann")
spot_price = pd.read_excel("Spot_2013-2019.xlsx")

#Parameters battery and electricity cost
Cap = 48 # Capacity in kWh
Pb_max = 48  #Maximum power output in kW
Pb_min = -48 #Minimum power output in Kw
SoC_init = 0.5*Cap
eta_bat = 0.9
Power_cost_summer = 47.25 #Cost per KW of max power drawn from grid
Power_cost_winter = 141.78

if period =='may' or period =='jun' or period =='jul' or period =='aug' or
period =='oct':
    Power_cost = Power_cost_summer
```

```
else:
    Power_cost = Power_cost_winter


#Parameters solar panel

N_panels_1 = 87 #east
N_panels_2 = 55 #west
N_panels_3 = 44 #south
N_cells = 72
I_sc = 9.515
V_oc = 44.84
I_mpp = 8.51
V_mpp = 37.62
Iph_ref = I_mpp #Amperes per 1kW of irradiation at mpp
Irr_ref = 1000 #W/m^2
I0_ref = 1*10**-10
n = 1 #Ideality factor
Temp_coeff_I = 0.0005 #% per deg K
Temp_coeff_Isc = Temp_coeff_I*I_sc #Amps per deg K
TC_vmpp = 0.0034 #%
T_ref = (25+273.15) #cell temperature at STC condition
q = 1.602*10**-19 #Electron charge
k = 1.381*10**-23 #Boltzmanns constant
E_gap = 1.12  #Bandgap for silicon in eV
V_pv_cell = V_mpp/N_cells #vmpp for each cell
eta_PVSyst = 0.8
A_PV_cell = (167.5*100.1)/N_cells # cm^2 per cell
NOCT = 48 #Nominal operating cell temperature in Celcius
alpha = np.deg2rad(58.1817) #Latitude of location in rad
lamda = (8.1646) #Longitude of location
Nc = 2 #Time zone Norway with respect to GMT
G_sc = 1.367 #Solar constant (kW/m^2)
d_f = 0.8 #Solar panel losses on site


################ Rs and Rsh calculations from IV_curve
#############################

Rsh = -((34.2367-0)/(8.65285-8.82556))
Rs = -((42.9-46.0705)/(5.42314-0))

######################### Importing day ahead API data
############################

presentday = datetime.now()
NextDay_Date = presentday + timedelta(1)
NextDay_Date_Formatted = NextDay_Date.strftime ('%Y-%m-%d') # format the
date to yyyymmdd

def jprint(obj):
    # create a formatted string of the Python JSON object
    text = json.dumps(obj, sort_keys=True, indent=4)
    print(text)

parameters = {
        "zone":"NO2",
        "date":str(NextDay_Date_Formatted)
}
```

```
spot_price_day_ahead = requests.get("https://norway-power.ffail.win/",
params=parameters)
irradiation = requests.get("https://api.solcast.com.au/weather_sites/5b3d-
a0d6-5a59-
612a/forecasts?period=PT30M&hours=24&format=json&api_key=VDN6E3e9vP79zQsb3f
K9jdJdZdPizHeI")

#jprint(spot_price_day_ahead.json())
Data_prices = spot_price_day_ahead.json()

#print(irradiation.status_code)
#jprint(irradiation.json())
Data_irradiation = irradiation.json()


price_day_ahead = []

#Code to convert json to readable format
def nested_dict_values_iterator(dict_obj):
    # Iterate over all values of given dictionary
    for value in dict_obj.values():
        # Check if value is of dict type
        if isinstance(value, dict):
            # If value is dict then iterate over all its values
            for v in  nested_dict_values_iterator(value):
                yield v
        else:
            # If value is not dict type then yield the value
            yield value

for value in nested_dict_values_iterator(Data_prices):
    price_day_ahead.append(value)

irr_ghi = [i['ghi'] for i in Data_irradiation['forecasts']] #Global
horizontal irradiation
irr_ebh = [i['ebh'] for i in Data_irradiation['forecasts']] #Direct (beam)
normal irradiation
irr_dhi = [i['dhi'] for i in Data_irradiation['forecasts']] #Diffuse
horizontal irradiation
temp = [i['air_temp'] for i in Data_irradiation['forecasts']] #Ambient
temperature

new_day = str(NextDay_Date_Formatted + 'T' + '00:00:00.0000000Z')
irr_index = next((index for (index, d) in
enumerate(Data_irradiation['forecasts']) if d["period_end"] == new_day),
None) #Getting the index of the first value for the next day to automize
script


irr_ghi = irr_ghi[irr_index:irr_index+48] #collecting the next 48 values,
which is given in 30 mins intervals
irr_ebh = irr_ebh[irr_index:irr_index+48]
irr_dhi = irr_dhi[irr_index:irr_index+48]

irr_ghi = [irr_ghi[x:x+2] for x in range(0, len(irr_ghi), 2)] #Dividing
values into sublists with two values for each hour
irr_ebh = [irr_ebh[x:x+2] for x in range(0, len(irr_ebh), 2)]
irr_dhi = [irr_dhi[x:x+2] for x in range(0, len(irr_dhi), 2)]

def avg_irr(irr):
```

```
    return (sum(irr)/float(len(irr)))

irr_ghi = [avg_irr(x) for x in irr_ghi]
irr_ebh = [avg_irr(x) for x in irr_ebh]
irr_dhi = [avg_irr(x) for x in irr_dhi]

#Implementing solar panel angle and lattitude of the sun for current day
ahead optimization#
civil_time_DayAhead = [x for x in range(1,25)]
days_DayAhead = ((datetime.now().timetuple().tm_yday)+1) #Day ahead as
number of total days in a year (365)

def irradiation_PV_day_ahead(civil_time,irr_ebh,irr_dhi, beta, gamma,
albedo): #beta = slope // gamma = azimuth // albedo = reflection in %

    delta = np.deg2rad(23.45 * sin(np.deg2rad(360 * ((284 + days_DayAhead)
/ (365))))) #Solar declination
    B = np.deg2rad(360 * ((days_DayAhead - 1) / 365))
    E = 3.82 * (0.000075 + 0.001868 * cos(B) - 0.032077 * sin(B) - 0.014615
* cos(2*B) - 0.04089 * sin(2*B)) #Equation of time
    solar_time = civil_time + (lamda / 15) - Nc + E
    solar_time_plus1 = civil_time + 1 + (lamda / 15) - Nc + E
    omega = np.deg2rad((solar_time - 12) * 15) #hour angle in rad
    omega_plus1 = np.deg2rad((solar_time_plus1 - 12) * 15)
    cos_theta = sin(delta) * sin(alpha) * cos(beta) - sin(delta) *
cos(alpha) * sin(beta) * cos(gamma) + cos(delta) * cos(alpha) * cos(beta) *
cos(omega) + cos(delta) * sin(alpha) * sin(beta) * cos(gamma) * cos(omega)
+ cos(delta) * sin(beta) * sin(gamma) * sin(omega) #Angle of incidence
    cos_theta_z = cos(alpha) * cos(delta) * cos(omega) + sin(alpha) *
sin(delta) #Zenith angle
    G_on = G_sc * 1000 * (1 + 0.033 * cos((360 * days_DayAhead) / (365)))
#Extraterrestial irradiation in Watts/m^2
    G_o_avg = (12 / math.pi) * G_on * (cos(alpha) * cos(delta) *
(sin(omega_plus1) - sin(omega)) + ((math.pi * (np.rad2deg(omega_plus1)-
np.rad2deg(omega)))/180) * sin(alpha) * sin(delta))
    irr_avg = irr_ebh+irr_dhi #Global solar radiation
    R_b = cos_theta/cos_theta_z #beam radiation ratio
    A_i = irr_ebh/G_o_avg #anisotropy index

    if irr_avg > 0:
        f = math.sqrt(irr_ebh/irr_avg)
    else:
        f = 0

    return (irr_ebh + irr_dhi * A_i) * R_b + irr_dhi * (1-A_i) * ((1 +
cos(beta)) / (2)) * (1 + f * sin(beta / 2) ** 3) + irr_avg * albedo * ((1 -
cos(beta)) / (2))

avg_irr_incident_1 =
[irradiation_PV_day_ahead(x,y,z,np.deg2rad(12),np.deg2rad(-
81),np.deg2rad(0.2)) for x,y,z in zip(civil_time_DayAhead, irr_ebh,
irr_dhi)]
avg_irr_incident_2 =
[irradiation_PV_day_ahead(x,y,z,np.deg2rad(12),np.deg2rad(99),np.deg2rad(0.
2)) for x,y,z in zip(civil_time_DayAhead, irr_ebh, irr_dhi)]
avg_irr_incident_3 =
[irradiation_PV_day_ahead(x,y,z,np.deg2rad(65),np.deg2rad(9),np.deg2rad(0.2
)) for x,y,z in zip(civil_time_DayAhead, irr_ebh, irr_dhi)]

irr_DayAhead = irr_ghi
```

```
###############
temp_DayAhead = temp[irr_index:irr_index+48] #Forecasted temperature for
the next 24 hours in 30 minute intervals
temp_DayAhead = [temp_DayAhead[x:x+2] for x in range(0, len(temp_DayAhead),
2)]

################
price_day_ahead = price_day_ahead[::3]
spot_DayAhead_buy = [x+0.2724 for x in price_day_ahead]
spot_DayAhead_sell = [x+0.04 for x in price_day_ahead]
############################################### API
#####################################

#Simplifying load data and converting values to list
load = load.drop(columns = ["Temp" , "Day"])
load = load.values.tolist()

#Removing sublist format
load_flat = []
for i in load:
  for j in i:
    load_flat.append(j)

load_year = load_flat
load_jan = load_year[:744]
load_feb = load_year[744:1416]
load_mar = load_year[1416:2160]
load_apr = load_year[2160:2880]
load_may = load_year[2880:3624]
load_jun = load_year[3624:4344]
load_jul = load_year[4344:5088]
load_aug = load_year[5088:5832]
load_sep = load_year[5832:6552]
load_oct = load_year[6552:7296]
load_nov = load_year[7296:8016]
load_dec = load_year[8016:8769]
load_DayAhead = load_year[2880:2904]

active_load = vars()["load_" + period]
active_load = np.array(active_load) #Converting from list to array for
easier manilpulation

#Extracting hourly average spot prices for january and june
spot_price_buy = spot_price[["Avg buy"]]

spot_price_buy = spot_price_buy[:len(spot_price_buy)-27] #Not needed if the
data contains the same amount of values
spot_price_buy = np.array(spot_price_buy)
spot_price_buy = spot_price_buy.reshape(8760,)
spot_price_sell = spot_price[["Avg sell"]]
spot_price_sell = spot_price_sell[:len(spot_price_sell)-27]
spot_price_sell = np.array(spot_price_sell)
spot_price_sell = spot_price_sell.reshape(8760,)
spot_year_buy = spot_price_buy
spot_year_sell = spot_price_sell
spot_jan_buy = spot_price_buy[:744]
spot_jan_sell = spot_price_sell[:744]
spot_feb_buy = spot_price_buy[744:1416]
spot_feb_sell = spot_price_sell[744:1416]
spot_mar_buy = spot_price_buy[1416:2160]
```

```
spot_mar_sell = spot_price_sell[1416:2160]
spot_apr_buy = spot_price_buy[2160:2880]
spot_apr_sell = spot_price_sell[2160:2880]
spot_may_buy = spot_price_buy[2880:3624]
spot_may_sell = spot_price_sell[2880:3624]
spot_jun_buy = spot_price_buy[3624:4344]
spot_jun_sell = spot_price_sell[3624:4344]
spot_jul_buy = spot_price_buy[4344:5088]
spot_jul_sell = spot_price_sell[4344:5088]
spot_aug_buy = spot_price_buy[5088:5832]
spot_aug_sell = spot_price_sell[5088:5832]
spot_sep_buy = spot_price_buy[5832:6552]
spot_sep_sell = spot_price_sell[5832:6552]
spot_oct_buy = spot_price_buy[6552:7296]
spot_oct_sell = spot_price_sell[6552:7296]
spot_nov_buy = spot_price_buy[7296:8016]
spot_nov_sell = spot_price_sell[7296:8016]
spot_dec_buy = spot_price_buy[8016:8769]
spot_dec_sell = spot_price_sell[8016:8769]

active_spot_buy = vars()["spot_" + period + "_buy"]
active_spot_sell = vars()["spot_" + period + "_sell"]

#Creating list with average hourly weather data for january and june
irr_year = weather_year["GHI irradiance (W/m2)"].tolist() #Data in minutes
(Global horizontal irradiation)
temp_year = weather_year["Air temperature (degC)"].tolist() #Data in
minutes

irr_year = [irr_year[x:x+60] for x in range(0, len(irr_year), 60)] #Data in
60 minutes per hour sublists
irr_year = irr_year[:len(irr_year)-24] #Removing the last 24 values to keep
the list in the same format as the other data

irr_jan = irr_year[:744]
irr_feb = irr_year[744:1416]
irr_mar = irr_year[1416:2160]
irr_apr = irr_year[2160:2880]
irr_may = irr_year[2880:3624]
irr_jun = irr_year[3624:4344]
irr_jul = irr_year[4344:5088]
irr_aug = irr_year[5088:5832]
irr_sep = irr_year[5832:6552]
irr_oct = irr_year[6552:7296]
irr_nov = irr_year[7296:8016]
irr_dec = irr_year[8016:8769]

active_irr = vars()["irr_" + period]

avg_irr_active = []
for x in active_irr:
    if len(active_irr) > 24:
        avg_irr_active.append(sum(x)/float(len(x)))
    else:
        avg_irr_active.append(x)

avg_irr_fix = []

for x in avg_irr_active:
    if x >= 0:
```

```
        avg_irr_fix.append(x)
    else:
        avg_irr_fix.append(0)

ghi_plot = [x/100 for x in avg_irr_fix]

#Assigning each day of the year to its yearly number
days_year = np.repeat(list(range(1,366)),24)
days_jan = np.repeat(list(range(1,32)),24)
days_feb = np.repeat(list(range(32,60)),24)
days_mar = np.repeat(list(range(60,91)),24)
days_apr = np.repeat(list(range(91,121)),24)
days_may = np.repeat(list(range(121,152)),24)
days_jun = np.repeat(list(range(152,182)),24)
days_jul = np.repeat(list(range(182,213)),24)
days_aug = np.repeat(list(range(213,244)),24)
days_sep = np.repeat(list(range(244,274)),24)
days_oct = np.repeat(list(range(274,305)),24)
days_nov = np.repeat(list(range(305,335)),24)
days_dec = np.repeat(list(range(335,366)),24)

day_of_year_active = vars()["days_" + period]

#Assigning a value of 1 to 24 for each hour in the month
civil_time_year = np.tile(list(range(1,25)),int(len(irr_year)/24))
civil_time_jan = np.tile(list(range(1,25)),int(len(irr_jan)/24))
civil_time_feb = np.tile(list(range(1,25)),int(len(irr_feb)/24))
civil_time_mar = np.tile(list(range(1,25)),int(len(irr_mar)/24))
civil_time_apr = np.tile(list(range(1,25)),int(len(irr_apr)/24))
civil_time_may = np.tile(list(range(1,25)),int(len(irr_may)/24))
civil_time_jun = np.tile(list(range(1,25)),int(len(irr_jun)/24))
civil_time_jul = np.tile(list(range(1,25)),int(len(irr_jul)/24))
civil_time_aug = np.tile(list(range(1,25)),int(len(irr_aug)/24))
civil_time_sep = np.tile(list(range(1,25)),int(len(irr_sep)/24))
civil_time_oct = np.tile(list(range(1,25)),int(len(irr_oct)/24))
civil_time_nov = np.tile(list(range(1,25)),int(len(irr_nov)/24))
civil_time_dec = np.tile(list(range(1,25)),int(len(irr_dec)/24))

civil_time_active = vars()["civil_time_" + period]

#Correcting for azimuth and slope of panels

def irradiation_PV_month(civil_time, avg_irr, day_of_year_active, beta,
gamma, albedo): #Beta = slope // gamma = azimuth // albedo = ground
reflectance in %

    delta = np.deg2rad(23.45 * sin(np.deg2rad(360 * ((284 +
day_of_year_active) / (365))))) #Solar declination
    B = np.deg2rad(360 * ((day_of_year_active - 1) / 365))
    E = 3.82 * (0.000075 + 0.001868 * cos(B) - 0.032077 * sin(B) - 0.014615
* cos(2*B) - 0.04089 * sin(2*B)) #Equation of time
    solar_time = civil_time + (lamda / 15) - Nc + E
    solar_time_plus1 = civil_time + 1 + (lamda / 15) - Nc + E
    omega = np.deg2rad((solar_time - 12) * 15) #hour angle in rad
    omega_plus1 = np.deg2rad((solar_time_plus1 - 12) * 15)
    cos_theta = sin(delta) * sin(alpha) * cos(beta) - sin(delta) *
cos(alpha) * sin(beta) * cos(gamma) + cos(delta) * cos(alpha) * cos(beta) *
cos(omega) + cos(delta) * sin(alpha) * sin(beta) * cos(gamma) * cos(omega)
+ cos(delta) * sin(beta) * sin(gamma) * sin(omega) #Angle of incidence
```

```
    cos_theta_z = cos(alpha) * cos(delta) * cos(omega) + sin(alpha) *
sin(delta) #Zenith angle
    G_on = G_sc * 1000 * (1 + 0.033 * cos((360 * day_of_year_active) /
(365))) #Extraterrestial irradiation in Watts/m^2
    G_o_avg = (12 / math.pi) * G_on * (cos(alpha) * cos(delta) *
(sin(omega_plus1) - sin(omega)) + ((math.pi * (np.rad2deg(omega_plus1)-
np.rad2deg(omega)))/180) * sin(alpha) * sin(delta))
    R_b = cos_theta / cos_theta_z #beam radiation ratio
    k_t = avg_irr / G_o_avg #Clearness index


    if k_t <= 0.22:
        G_d = (1-0.09 * k_t) * avg_irr #Diffuse irradiation
    elif k_t > 0.22 and k_t <= 0.8:
        G_d = (0.9511 - 0.1604 * k_t + 4.388 * k_t ** 2 - 16.638 * k_t ** 3
+ 12.336 * k_t ** 4) * avg_irr
    elif k_t > 0.8:
        G_d = 0.165


    G_b = avg_irr - G_d #direct (beam) irradiation

    if G_b > 0 and avg_irr > 0:
        f = math.sqrt(G_b/avg_irr)
    else:
        f = 0

    A_i = G_b/G_o_avg #anisotropy index

    return (G_b + G_d * A_i) * R_b + G_d * (1-A_i) * ((1 + cos(beta)) /
(2)) * (1 + f * (sin(beta / 2) ** 3)) + avg_irr * albedo * ((1 - cos(beta))
/ (2))

if len(avg_irr_fix) > 24:
    avg_irr_incident_1 =
[irradiation_PV_month(x,y,z,np.deg2rad(12),np.deg2rad(-81),0.2) for x,y,z
in zip(civil_time_active, avg_irr_fix, day_of_year_active)]
    avg_irr_incident_2 =
[irradiation_PV_month(x,y,z,np.deg2rad(12),np.deg2rad(99),0.2) for x,y,z in
zip(civil_time_active, avg_irr_fix, day_of_year_active)]
    avg_irr_incident_3 =
[irradiation_PV_month(x,y,z,np.deg2rad(65),np.deg2rad(9),0.2) for x,y,z in
zip(civil_time_active, avg_irr_fix, day_of_year_active)]

def remove_outliers(irr_incident):
    avg_irr = []

    for x in irr_incident:
        if x > 2000:
            avg_irr.append(0)
        elif x < 0:
            avg_irr.append(0)
        else:
            avg_irr.append(x)

    return avg_irr

avg_irr_1 = remove_outliers(avg_irr_incident_1)
avg_irr_2 = remove_outliers(avg_irr_incident_2)
avg_irr_3 = remove_outliers(avg_irr_incident_3)
```

```
#####################

temp_year = [temp_year[x:x+60] for x in range(0, len(temp_year), 60)]
temp_year = temp_year[:len(temp_year)-24]

temp_jan = temp_year[:744]
temp_feb = temp_year[744:1416]
temp_mar = temp_year[1416:2160]
temp_apr = temp_year[2160:2880]
temp_may = temp_year[2880:3624]
temp_jun = temp_year[3624:4344]
temp_jul = temp_year[4344:5088]
temp_aug = temp_year[5088:5832]
temp_sep = temp_year[5832:6552]
temp_oct = temp_year[6552:7296]
temp_nov = temp_year[7296:8016]
temp_dec = temp_year[8016:8769]

active_temp = vars()["temp_" + period]

avg_temp_active = []  #Ambient temperature
for y in active_temp:
    avg_temp_active.append(sum(y)/float(len(y)))


def cell_temp(avg_irr,avg_temp_active):
    avg_temp_active_cell = []
    for x, y in zip(avg_irr, avg_temp_active):
        avg_temp_active_cell.append(y+(NOCT-20)*(x/800))
    return avg_temp_active_cell

avg_temp_active_cell_1 = cell_temp(avg_irr_1,avg_temp_active)
avg_temp_active_cell_2 = cell_temp(avg_irr_2,avg_temp_active)
avg_temp_active_cell_3 = cell_temp(avg_irr_3,avg_temp_active)

Vmpp_cell_1 = [(V_mpp*(1-TC_vmpp*(i-25))) for i in avg_temp_active_cell_1]
Vmpp_cell_2 = [(V_mpp*(1-TC_vmpp*(i-25))) for i in avg_temp_active_cell_2]
Vmpp_cell_3 = [(V_mpp*(1-TC_vmpp*(i-25))) for i in avg_temp_active_cell_3]

#Calculating daily hourly average PV generation
def Generated_current(I,x,y,z):
    I_ph = (x/Irr_ref)*(Iph_ref+Temp_coeff_Isc*((y+273.15)-(T_ref)))
    I_sat = I0_ref*((y+273.15)/T_ref)**3*math.exp(((q*E_gap/1*k)*(1/T_ref-
1/(y+273.15))))
    return I_ph-
I_sat*(math.exp((q*(z/N_cells+I*Rs))/(N_cells*n*k*(y+273.15)))-1)-
(z/N_cells+I*Rs)/Rsh-I

I_generated_1 = [fsolve(Generated_current, I_mpp, args=(x,y,z)) for x, y, z
in zip(avg_irr_1, avg_temp_active_cell_1, Vmpp_cell_1)]
I_generated_2 = [fsolve(Generated_current, I_mpp, args=(x,y,z)) for x, y, z
in zip(avg_irr_2, avg_temp_active_cell_2, Vmpp_cell_2)]
I_generated_3 = [fsolve(Generated_current, I_mpp, args=(x,y,z)) for x, y, z
in zip(avg_irr_3, avg_temp_active_cell_3, Vmpp_cell_3)]

I_generated_1 = np.array(I_generated_1).squeeze()
I_generated_2 = np.array(I_generated_2).squeeze()
I_generated_3 = np.array(I_generated_3).squeeze()
```

```
Ps_active = []

for x,y,z,a,s,d in zip(I_generated_1,I_generated_2,I_generated_3,
Vmpp_cell_1, Vmpp_cell_2, Vmpp_cell_3):
    if x > 0:

Ps_active.append((x*a*N_panels_1/1000+y*s*N_panels_2/1000+z*d*N_panels_3/10
00)*d_f)
    else:
        Ps_active.append(0)

Ps_active_array = np.array(Ps_active)
Ps_active_tot = sum(Ps_active)

#Calculating total daily power generation for day ahead optimization
Ps_active_daily = [sum(Ps_active[i:i+24]) for i in range(0, len(Ps_active),
24)]

#Calculating total daily power demand for day ahead optimization
Pd_active_daily = [sum(active_load[i:i+24]) for i in range(0,
len(active_load), 24)]

Day_ahead_active = []

for i, j in zip(Ps_active_daily, Pd_active_daily):
    if i > j:
        Day_ahead_active.append([1]*24)
    else:
        Day_ahead_active.append([0]*24)

Day_ahead_active_flat = []
for i in Day_ahead_active:
  for j in i:
    Day_ahead_active_flat.append(j)

#Calculating net load for the chosen period
net_load_active = active_load - Ps_active


#-------------------------------------------------------

#Splitting the net load into positive or negative values so that the
algorithm knows whether to sell or buy power from the grid
posLoad = np.copy(active_load - Ps_active)
negLoad = np.copy(active_load - Ps_active)

for j,e in enumerate(net_load_active): #j indicate the index and e indicate
the actual value
    if e>=0:  #If the net load is greater than 0 it means that power will
be bought from the grid. Therefore the negative load is set to 0 and is
unused
        negLoad[j]=0
    else:
        posLoad[j]=0
#We then add each value in a dictionary for either positive or negative
load
posLoadDict = dict(enumerate(posLoad))
negLoadDict = dict(enumerate(negLoad))

#Maximum monthly load demand used to compare scenarios
```

```
if len(active_load) > 24:
    P_max_demand = max(posLoad)
else:
    P_max_demand = 0  #This is not actually the case, but is here to give
the correct total cost in line 643 depending on simulation period

#Creating optimization rutine using Pyomo
from pyomo.core import Var
import pyomo.environ as pyomo

#Establishing the model
m = pyomo.ConcreteModel()

#The total simulation time will be equal to the length of the chosen
timeframe.
m.Time = pyomo.RangeSet(0, len(net_load_active)-1)

#Variables (OutputVars 0 to 9)
m.SOC = pyomo.Var(m.Time, bounds=((0.2*Cap),(Cap)), initialize=0) #This
variable will have different upper bounds depending on the day ahead values
decided by the SOC_bounds constraint
m.posDeltaSOC = pyomo.Var(m.Time, initialize=0) #Positive change in SoC -
This and the next variable is used to calculate total SoC at any time,t
m.negDeltaSOC = pyomo.Var(m.Time, initialize=0) #Negative change in SoC
m.EInGrid = pyomo.Var(m.Time, bounds=(0,Pb_max), initialize=0)  #How much
power from the grid is fed into the battery at time t
m.EInPV = pyomo.Var(m.Time, bounds=(0,Pb_max), initialize=0) #How much of
the generated from PV is fed into the battery at time t
m.negEOutLocal = pyomo.Var(m.Time, bounds=(Pb_min,0), initialize=0)
#Discharge from battery to local demand, 0.2 is in place to account for
inverter efficiency
m.negEOutExport = pyomo.Var(m.Time, bounds=(Pb_min,0), initialize=0) #Sold
electricity from battery, 0.2 is in place to account for inverter
efficiency
m.posNetLoad = pyomo.Var(m.Time, initialize=posLoadDict)
m.negNetLoad = pyomo.Var(m.Time, initialize=negLoadDict)
m.P_max = pyomo.Var(domain=pyomo.NonNegativeReals)

# Parameters not established earlier
m.posLoad = pyomo.Param(m.Time, initialize=posLoadDict) #Total power drawn
from grid unless battery is discharging
m.negLoad = pyomo.Param(m.Time, initialize=negLoadDict) #Total power sold

#Object function which aims to reduce price and average SOC as well as
maximize the self consumption if desired
def Obj_fn(m):
    return (sum((active_spot_buy[i]*m.posNetLoad[i]) +
(active_spot_sell[i]*m.negNetLoad[i]) + (0*m.SOC[i]) +
(0*(m.EInGrid[i]+m.EInPV[i]-m.negEOutLocal[i]-m.negEOutExport[i])) for i in
m.Time) + m.P_max*Power_cost)
m.total_cost = pyomo.Objective(rule=Obj_fn,sense=pyomo.minimize)



# constraints

def SOC_rule(m,t):
    if t==0:
        return (m.SOC[t] == SoC_init)
    else:
```

```
        return (m.SOC[t] == m.SOC[t-1]+m.posDeltaSOC[t]+m.negDeltaSOC[t])

m.Batt_SOC = pyomo.Constraint(m.Time,rule=SOC_rule)

#---------PEAK SHAVING----------

def P_max_rule(m,i):
    if len(active_load) > 24: #This variable is not included for day ahead
optimization
        return m.P_max >= m.posNetLoad[i]
    else:
        return m.P_max == 0
#This makes it so that the algorithm finds the lowest possible P_max when
using the battery if the simulated time is an entire month
m.peak_cons = pyomo.Constraint(m.Time, rule=P_max_rule)


#Implementing charging efficiency
def pos_E_in_rule(m,i):
    return (m.EInGrid[i]+m.EInPV[i]) == m.posDeltaSOC[i]/eta_bat
m.posEIn = pyomo.Constraint(m.Time, rule=pos_E_in_rule)
# ensure discharging eff multiplied
def neg_E_out_rule(m,i):
    return (m.negEOutLocal[i]+m.negEOutExport[i]) ==
m.negDeltaSOC[i]*eta_bat
m.negEOut = pyomo.Constraint(m.Time, rule=neg_E_out_rule)


# ensure charging rate obeyed
def E_charging_rate_rule(m,i):
    return (m.EInGrid[i]+m.EInPV[i])<=Pb_max
m.chargingLimit = pyomo.Constraint(m.Time, rule=E_charging_rate_rule)
# ensure DIScharging rate obeyed
def E_discharging_rate_rule(m,i):
    return (m.negEOutLocal[i]+m.negEOutExport[i])>=Pb_min
m.dischargingLimit = pyomo.Constraint(m.Time, rule=E_discharging_rate_rule)


# ensure that EInPV cannot exceed local PV
def E_solar_charging_rule(m,i):
    return m.EInPV[i]<=-m.negLoad[i]
m.solarChargingLimit = pyomo.Constraint(m.Time, rule=E_solar_charging_rule)

# ensure that negEOutLocal cannot exceed local demand
def E_local_discharge_rule(m,i):
    return m.negEOutLocal[i]>=-m.posLoad[i]
m.localDischargingLimit = pyomo.Constraint(m.Time,
rule=E_local_discharge_rule)

# calculate the net positive demand
def E_pos_net_rule(m,i): #Cost calculated with buy price
    return m.posNetLoad[i] == m.posLoad[i]+m.EInGrid[i]+m.negEOutLocal[i]
m.E_posNet = pyomo.Constraint(m.Time,rule=E_pos_net_rule)

# calculate exported energy
def E_neg_net_rule(m,i): #Cost calculated with sell price
    return m.negNetLoad[i] == m.negLoad[i]+m.EInPV[i]+m.negEOutExport[i]
m.E_negNet = pyomo.Constraint(m.Time,rule=E_neg_net_rule)

#Choosing solver
```

```
opt = pyomo.SolverFactory("glpk")

t = time.time()
results = opt.solve(m)
elapsed = time.time() - t
print ('Time elapsed:', elapsed)


#reading in the value for each of the variables
outputVars = np.zeros((10,len(active_spot_sell)))


j = 0
for v in m.component_objects(Var, active=True):
    varobject = getattr(m, str(v))
    for index in varobject:
        outputVars[j,index] = varobject[index].value
    j+=1
    if j>=10:
        break

#get information we want to read
P_max = outputVars[9]
P_max = max(P_max)
cost_without_batt = P_max_demand*Power_cost +
np.sum([(active_spot_buy[i]*posLoad[i] + active_spot_sell[i]*negLoad[i])
for i in range(len(active_spot_buy))])
cost_with_batt = P_max*Power_cost +
np.sum([(active_spot_buy[i]*outputVars[7,i] +
active_spot_sell[i]*outputVars[8,i]) for i in range(len(active_spot_buy))])
Utility_cost = np.sum([(active_spot_buy[i]*outputVars[7,i] +
active_spot_sell[i]*outputVars[8,i]) for i in range(len(active_spot_buy))])
Power_cost = P_max*Power_cost
PV_charge = np.sum([(outputVars[4,i]) for i in
range(len(active_spot_buy))])
Grid_charge = np.sum([(outputVars[3,i]) for i in
range(len(active_spot_buy))])
Excess_energy = np.sum([-negLoad[i] - (outputVars[4,i]) for i in
range(len(active_spot_buy))])
Avg_SOC = np.sum([(outputVars[0,i]) for i in range(len(active_spot_buy))])
Batt_export = np.sum([(outputVars[6,i]) for i in
range(len(active_spot_buy))])
Batt_local =  np.sum([(outputVars[5,i]) for i in
range(len(active_spot_buy))])


print ('Cost without battery:', cost_without_batt)
print ('Cost with battery:', cost_with_batt)
print ('Difference: %.4f'%((cost_with_batt - cost_without_batt) /
np.abs(cost_without_batt)))
print ("P_max:", P_max)
print ("Self_consumption_bat", 1-(Excess_energy/Ps_active_tot)) #How much
of the generated energy is used locally
print ("Self_consumption", 1-(sum(-negLoad)/Ps_active_tot))
print ("Average SOC", (Avg_SOC/(len(active_load)))/48) #Average SOC per
hour
print ("PV charge:", PV_charge)
print ("Grid charge:", Grid_charge)
print ("Battery export", -Batt_export)
print ("Battery local", -Batt_local)
```

```
print ("Throughput", PV_charge+Grid_charge-Batt_export-Batt_local)
print ("Sold energy", -Excess_energy)
print ("Renewable fraction", (Ps_active_tot)/sum(active_load))
print ("Peak reduction", P_max_demand - P_max)
print ("Cycles", (PV_charge+Grid_charge-Batt_export-Batt_local)/Cap)
print ("Utility cost", Utility_cost)
print ("Power cost", Power_cost)

newNetLoad = outputVars[7]+outputVars[8]
netDemand = posLoad + negLoad + outputVars[5] + outputVars[3]

# plotting results
colors = sns.color_palette()
hrs = np.arange(0,len(active_load))
fig = plt.figure(figsize=(18,9))
fig.suptitle('Results', fontsize=18)
ax1 = fig.add_subplot(3,1,1)
l1, = ax1.plot(hrs,active_load,color=colors[0])
l2, = ax1.plot(hrs,Ps_active,color=colors[1])
l3, = ax1.plot(hrs,netDemand,color=colors[3])
ax1.set_xlabel('hour'), ax1.set_ylabel('kW')
ax1.legend([l1,l2,l3],['demand','PV','Net demand with storage'],ncol=3)
ax1.set_xlim([0,len(active_load)]);
ax2 = fig.add_subplot(3,1,2)
l1, = ax2.plot(hrs,active_spot_buy,color=colors[3])
l2, = ax2.plot(hrs,active_spot_sell,color=colors[4])
ax2.set_xlabel('hour'), ax2.set_ylabel('price [NOK]')
ax2.legend([l1,l2],['buy price','sell price'],ncol=2)
ax2.set_xlim([0,len(active_load)]);
ax3 = fig.add_subplot(3,1,3)
l1, = ax3.plot(hrs,np.sum(outputVars[3:7,:], axis=0),color=colors[5])
l2, = ax3.plot(hrs,outputVars[0],color=colors[4])
ax3.set_xlabel('hour'), ax3.set_ylabel('action')
ax3.legend([l1,l2],['battery action (kW)','Energy content (kWh)'],ncol=2)
ax3.set_xlim([0,len(active_load)]);
fig.tight_layout()


avg_irr_1_plot = [x/100 for x in avg_irr_1]
avg_irr_2_plot = [x/100 for x in avg_irr_2]
avg_irr_3_plot = [x/100 for x in avg_irr_3]

colors = sns.color_palette()
hrs = np.arange(0,len(active_load))
fig = plt.figure(figsize=(14,7))
fig.suptitle('East panels', fontsize=18)
ax1 = fig.add_subplot(3,1,1)
l1, = ax1.plot(hrs,avg_temp_active,color=colors[0])
l2, = ax1.plot(hrs,I_generated_1,color=colors[1])
l3, = ax1.plot(hrs,avg_irr_1_plot,color=colors[3])
l4, = ax1.plot(hrs,ghi_plot,color=colors[4])
ax1.set_xlabel('hour'), ax1.set_ylabel('Temp(degC) / PV (I Gen)')
ax1.legend([l1,l2,l3,l4],['ambient temp','PV (I
Gen)','irradiation_incident','ghi'],ncol=4)
ax1.set_xlim([0,len(active_load)]);

colors = sns.color_palette()
hrs = np.arange(0,len(active_load))
fig = plt.figure(figsize=(14,7))
fig.suptitle('West panels', fontsize=18)
```

```
ax1 = fig.add_subplot(3,1,1)
l1, = ax1.plot(hrs,avg_temp_active,color=colors[0])
l2, = ax1.plot(hrs,I_generated_2,color=colors[1])
l3, = ax1.plot(hrs,avg_irr_2_plot,color=colors[3])
l4, = ax1.plot(hrs,ghi_plot,color=colors[4])
ax1.set_xlabel('hour'), ax1.set_ylabel('Temp(degC) / PV (I Gen)')
ax1.legend([l1,l2,l3,l4],['ambient temp','PV (I
Gen)','irradiation_incident','ghi'],ncol=4)
ax1.set_xlim([0,len(active_load)]);

colors = sns.color_palette()
hrs = np.arange(0,len(active_load))
fig = plt.figure(figsize=(14,7))
fig.suptitle('South panels', fontsize=18)
ax1 = fig.add_subplot(3,1,1)
l1, = ax1.plot(hrs,avg_temp_active,color=colors[0])
l2, = ax1.plot(hrs,I_generated_3,color=colors[1])
l3, = ax1.plot(hrs,avg_irr_3_plot,color=colors[3])
l4, = ax1.plot(hrs,ghi_plot,color=colors[4])
ax1.set_xlabel('hour'), ax1.set_ylabel('Temp(degC) / PV (I Gen)')
ax1.legend([l1,l2,l3,l4],['ambient temp','PV (I
Gen)','irradiation_incident','ghi'],ncol=4)
ax1.set_xlim([0,len(active_load)]);

############# OTHER PLOTS ###########
##Time period changed depending on wanted plot
#
#colors = sns.color_palette()
#hrs = np.arange(0,len(irr_jun))
#fig = plt.figure(figsize=(14,7))
#fig.suptitle('Irradiation and air temp June', fontsize=18)
#ax1 = fig.add_subplot(3,1,1)
#ax2 = ax1.twinx()
#l1, = ax1.plot(hrs,[sum(x)/float(len(x)) for x in
temp_jun],color=colors[0])
#l2, = ax2.plot(hrs,[sum(x)/float(len(x)) for x in
irr_jun],color=colors[1])
#ax1.set_xlabel('hour'), ax1.set_ylabel('Temp(degC)'),
ax2.set_ylabel('Irradiation(W/m^2)')
#ax1.legend([l1,l2],['ambient temp','Global horizontal
irradiation'],ncol=2,loc="upper left")
#ax1.set_xlim([0,len(irr_jun)]);
#
#colors = sns.color_palette()
#hrs = np.arange(0,len(spot_jun_buy))
#fig = plt.figure(figsize=(14,7))
#fig.suptitle('Spot Price June', fontsize=18)
#ax1 = fig.add_subplot(3,1,1)
#l1, = ax1.plot(hrs,spot_jun_buy,color=colors[0])
#l2, = ax1.plot(hrs,spot_jun_sell,color=colors[1])
#ax1.set_xlabel('hour'), ax1.set_ylabel('Spot Price (NOK)')
#ax1.legend([l1,l2],['Spot price buy','Spot price sell'],ncol=2,loc="upper
left")
#ax1.set_xlim([0,len(spot_jun_buy)]);
#
#colors = sns.color_palette()
#hrs = np.arange(0,len(load_jun))
#fig = plt.figure(figsize=(14,7))
#fig.suptitle('Load June', fontsize=18)
#ax1 = fig.add_subplot(3,1,1)
```

```
#l1, = ax1.plot(hrs,load_jun,color=colors[0])
#ax1.set_xlabel('hour'), ax1.set_ylabel('Load (kW)')
#ax1.legend([l1,l2],['Load January'],ncol=2,loc="upper left")
#ax1.set_xlim([0,len(load_jun)]);
#
#colors = sns.color_palette()
#hrs = np.arange(0,len(avg_irr_incident_2))
#fig = plt.figure(figsize=(14,7))
#fig.suptitle('Incident irradiation', fontsize=18)
#ax1 = fig.add_subplot(3,1,1)
#l1, = ax1.plot(hrs,avg_irr_incident_2,color=colors[0])
#ax1.set_xlabel('hour'), ax1.set_ylabel('Irradiation (W)')
#ax1.legend([l1,l2],['Irradiation'],ncol=2,loc="upper left")
#ax1.set_xlim([0,len(avg_irr_incident_2)]);
```

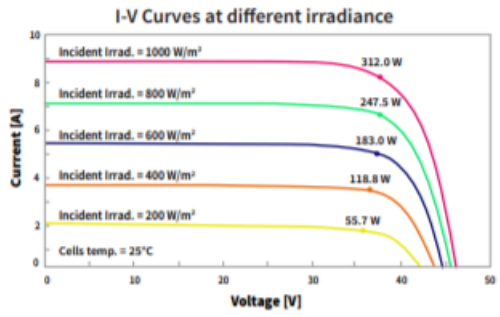
PV parameters

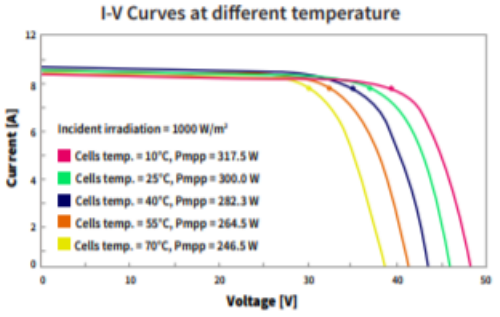
Electrical Characteristics

SOLAR CELLS	POLY-CRYSTALLINE 156 × 156 MM 72 PCS. (6×12) – 4 BUS BARS				
Maximum Power (Pmax)	300 Wp	305 Wp	310 Wp	315 Wp	320 Wp
Voltage at Pmax (Vmp)	37.23 V	37.24 V	37.32 V	37.46 V	37.62 V
Current at Pmax (Imp)	8.06 A	8.19 A	8.31 A	8.41 A	8.51 A
Open-Circuit Voltage (Voc)	44.71 V	44.72 V	44.76 V	44.82 V	44.84 V
Short-Circuit Current (Isc)	8.947 A	9.094 A	9.234 A	9.371 A	9.515 A
Maximum System Voltage (V DC)	1000 V (iec), 600 V (UL)				
Cell Efficiency	17.46 %	17.75 %	18.05 %	18.34 %	18.63 %
Module Efficiency	15.46 %	15.72 %	15.98 %	16.23 %	16.49 %
Number of By-pass Diodes	6				
Maximum Series Fuse	15 A				
Temperature Coefficient of Pmax	- 0.45 % / °C				
Temperature Coefficient of Voc	- 0.34 % / °C				
Temperature Coefficient of Isc	- 0.05 % / °C				
Nominal Operating Cell Temperature	47 ± 2 °C				

Test Parameters

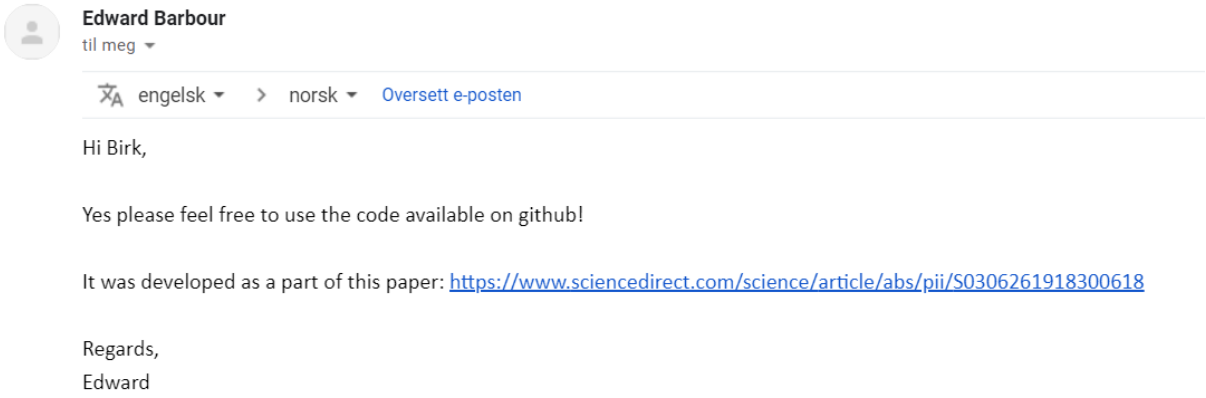
Dielectric Insulation Voltage	6,000 V DC max
Operating Temperature	-40 °C to 85 °C





Sunceco, Inc. | www.sunceco.com | email: info@sunceco.com

Permission to use code



Edward Barbour
til meg

engelsk > norsk Oversett e-posten

Hi Birk,

Yes please feel free to use the code available on github!

It was developed as a part of this paper: https://www.sciencedirect.com/science/article/abs/pii/S0306261918300618

Regards,
Edward