Norwegian University
of Life Sciences

Master's Thesis 2019 30 ECTS
Faculty of Science and Technology

Methodology for Assessing Short-term Flexibility in Demand-side Assets

Martin Haug
MSc. Environmental Physics and Renewable Energy

This page is intentionally left blank.

Preface

This piece of paper means that my studies in Ås has come to an end. It has been a challenging, yet thrilling journey of my life and this last semester has not been an exception.

First and foremost, I would like to express my gratitude to my supervisors Heidi Samuelsen Nygård and Stig Ødegaard Ottesen, whose help and guidance has been essential for the shaping of this thesis, by providing constructive and encouraging feedback. Thanks to Heidi for always keeping the door open, to always support and guide a bewildered student. Thanks to Stig and eSmart for the warm welcome. I am grateful for getting the opportunity to dig into this exciting research field in my last semester.

I want to thank my family for all the love, care and support throughout the years. Furthermore, I would like to let all my fellow students know that all these years would not have been as rich without you.

Some extra appreciation is given to the coffee culture in TF1-211, joy and weird humor at Nylenda and the many sunsets behind the oak tree which is soon to be world famous in Christmas letters around the world.

My last wish is for all people to be aware of the consequences of their needs and demands, and to be humble about the resources they do and do not have.

Ås, Dec 16th 2019

Martin Haug

Summary

Global electricity grids, and especially the distribution grids, encounter new challenges during the transmission to a sustainable energy chain. Decarbonization involves electrification and a massive deployment of variable renewable energy sources, which ultimately increase the complexity at the demand-side of the grid. There is a growing need to promote demand-side flexible power and to actively utilize it to deal with an anticipated increase of local congestions and ramping problems. Local flexibility markets have emerged to provide a platform where the distribution grid operator or another flexibility buyer can activate demand-side flexibility that is offered by prosumers, e.g. balance the grid.

In order for a prosumer to make its flexible power accessible on markets, new methodologies are needed. The main goal of this thesis is to develop a methodology for assessing short-term flexible power in a demand-side asset. Such a methodology has been developed for a generic flexible asset and consists of four stages: (1) load forecasts (2) physical asset models (3) estimation of available flexibility and at last (4) the shaping of a flexibility bid for flexibility markets. The thesis gives conceptual descriptions on how the methodology is implemented for each of five different flexible assets. Python is used as a tool for implementing the methodology, using the package Keras to make RNN forecast models and object-oriented programming to create an *Asset* class framework.

The methodology is applied on a real use-case scenario where multi-step RNN forecast models are created, using real consumption data for an asset that powers a cooling storage. Data are provided by ASKO (end-user) and eSmart (smart grid company). The forecast results seems promising even with relative short data, but must be optimized, tested on multiple test sets and include explanatory variables. Many assumptions had to be made for the asset parameters and the final hypothetical flexibility estimates were shown to be sensitive to these choices. Nevertheless, the methodology has been proven to work and is applied to a full demonstration of a bid procedure. Applied examples are also given for other assets, such as water heater and a battery.

The conclusion is that the methodology itself is stable and applicable to many different assets. Its results however, being the flexibility estimates, are prone to be very wrong if the constitutional stages in the methodology are weakly implemented. A strength is that each stage is flexible to be changed or improved without disturbing the flow of the methodology. For the methodology to work successfully, it is of utmost importance that accurate load (or production) forecasts and correct asset parameters are provided.

Sammendrag

Overgangen til et bærekraftig energisystem fører meg seg nye utfordringer for kraftnettet. Distribusjonsnettet vil oppleve en mer kompleks strømflyt som følge av elektrifisering og en massiv utrulling av uregulerbar, og til dels distribuert, strømproduksjon. Det er forventet en økning i lokale nettutfordringer i form av overbelastning og hurtig ramping, noe som gir et økt behov for å fremme og aktivt ta i bruk fleksibel effekt hos sluttbrukeren i operasjonen av distribusjonsnettet. Nye lokale fleksibilitetsmarkeder er i fremmarsj og har som mål å tilby en åpen plattform der sluttbrukere kan selge sin fleksibilitet til en nettoperatør eller andre kjøpere som trenger denne fleksibiliteten, eksempelvis for å avlaste nettet.

For å frigjøre sluttbrukerfleksibilitet til slike markeder, er det nødvendig med ny metodikk. Hovedmålet med denne oppgaven er å utvikle en metodikk for å estimere kortsiktig fleksibilitet i en distribuert strømkomponent, også kalt *asset*. En slik metodikk har blitt utviklet for en generell asset og består av fire trinn: (1) lastprediksjoner (2) fysisk modell av en asset (3) estimering av tilgjengelig fleksibilitet og (4) utforming av et bud mot et fleksibilitetsmarked. Oppgaven tar videre for seg hvordan metodikken kan implementeres for fem ulike asseter. Python brukes som et verktøy for å implementere metodikken, ved å bruke pakken Keras for å lage RNN-modeller og objektorientert programmering for å lage en *Asset*-klasse.

Metodikken har blitt anvendt på en reell asset som brukes til å kjøle et kjølelager. Forbruksdata er gitt av ASKO (sluttbruker) og eSmart (smart grid-selskap), og er brukt for å utvikle RNN-modeller til å prediktere assetens fremtidige forbruk. Modellene virker lovende, men må optimaliseres, testes på flere testsett og inkludere flere features. Det er gjort flere antagelser for asseten som har vært utslagsgivende for fleksibilitetsestimatene. Det har blitt vist at metodikken fungerer og den har blitt anvendt videre i en hypotetisk budprosedyre. Eksempler for implementering av metodikken på andre relevante assets er også gitt, nemlig en elektrokjele og et batteri.

Sluttkonklusjonen er at selve metodikken er stabil og anvendelig for mange forskjellige asseter, men fleksibilitetsestimatene kan bare bli like gode som parameterene og prediksjonene. Styrken til metodikken stegene kan endres uten å forstyrre flyten i metodikken, eksempelvis benytte en bedre lastprediksjonsmodell eller justere parameterene. For at metodikken skal fungere vellykket, så trenger den nøyaktige lastprediksjoner og riktige parametere for asseten.

Contents

List of Figures

List of Tables

Abbreviations

Abbreviation	Meaning
API	Application Programming Interface
LV/MV/HV grid	Low-/Medium-/High-Voltage grid
TSO/DSO	Transmission/Distribution grid operator (operators of the HV transmission grid and MV/LV distribution grids respectively)
(vRES) RES	(Variable) Renewable energy sources
DFS	Decentral/Demand-side flexibility sources
ML	Machine Learning
MLP	Multiple Linear Regression
RNN	Recurrent Neural Network
LSTM	Long-short-term memory
PV panels	Photovolatic solar panels
DA and ID market	Day-Ahead and Intra-Day market for trading electricity
DRM	Demand Response Management
MAE, MSE	Mean Absolute Error, Mean Squared Error
SoC	State of Charge
COP	Coefficient of Performance

Mathematical notation

Mathematical symbol	Meaning
$\vec{\hat{b}} = \left[\hat{b^{(1)}}, \hat{b^{(2)}}, \ldots, \hat{b^{(t)}}, \ldots, \hat{b^{(H)}}\right]$	Forecasted baseline
$\vec{b} = \left[b^{(1)}, b^{(2)}, \ldots, b^{(t)}, \ldots, b^{(H)}\right]$	True baseline
$E^{(t)}$	The energy storage level of the asset.
E_{min}/E_{max}	The minimum/maximum allowed energy level of the asset's energy storage.
$\Delta E_{range} = E_{max} - E_{min}$	Total flexible energy level range for the asset's energy storage.
P_{cons}	Electrical power consumption from the grid for the asset.
$\vec{P}_{max}$	The maximum possible power consumption of the asset.
$\vec{P}_{min}$	The minimum possible power consumption of the asset.
$P_{in} = eP_{cons}$	The share of consumption power that flows in to the asset's energy storage.
e	Efficiency factor for converting grid electricity to input power.
$P_{(dis)charge} = P_{in} + P_{losses}$	Net power flow in to the asset's energy storage.
P_{losses}	Power losses flowing out of the asset's energy storage.
$\Delta E^{(t)} = P^{(t)}_{(dis)charge}\Delta t$	Change of energy level of the asset's energy storage, due to charging.
$\Delta SoC^{(t)} = n\Delta E^{(t)}$	Change in the SoC level for an asset's energy storage, due to charging.

$n = \frac{1}{\Delta E_{range}}$	Normalization factor, convert from absolute energy level to SoC level for the asset's energy storage.
$P_{cons,SS}$	The consumption which leads to a steady state situation (no charging and no change in SoC level).
$\vec{\hat{F}}_p = \vec{P}_{max} - \vec{\hat{b}}$	Estimated maximum positive flexible power.
$\vec{\hat{F}}_n = \vec{P}_{min} - \vec{\hat{b}}$	Estimated maximum negative flexible power.
$\vec{F}_{committed}$ and $\vec{F}_{delivered}$	Committed (bought) and delivered flexible power.
$R^{(h)}_{delivered} = F^{(h)}_{delivered} - F^{(h)}_{committed}$	Error of delivered flexibility.
t	Current timestep/timeslot.
h or $t + h$, with $h \in \{1, 2, ..., H\}$	Forecasted timestep/timeslot.
H	Bidding horizon/number of timeslots.

Chapter 1

Introduction

This first chapter creates the framework, presenting the background and motivation behind the work of this thesis. Then a problem statement is formulated by means of one main goal and several sub goals. The tools, data, methods and use-case that are used to address the goals are presented at last.

1.1 Background

Understanding our timeline of energy can be helpful for understanding the energy situation of today. Energy is a real evolutionary drug. Humans did once go from being wanderers to evolve around agriculture, a "hack" of food supply enabling us to grow metropoles. Humans invented machines at a point, resulting in horses being replaced by horsepower. Suddenly, this "hack" of energy made a lot of cheap work available for us, through fossil fuels. From that point, energy usage only escalated. Industry and cities could expand remotely from the rivers and electricity was invented. Electricity, goods, transport, house appliances, and followingly improved health and wealth became achievable for many. Today, humankind has developed a very vulnerable relationship with the energy chain, which concern all sides of the globalized human society such as transport, health, water, food and communication to mention some. There are no doubts that we have built great societies, however they depend upon a stable and secure source of energy supply to maintain their vital functions.

In our age, the consensus about global warming is clear. There are huge social forces, demonstrations in the streets which demand immediate climate action now and the world leaders are taking responsibility, more or less. The reason lies in that around 80% of world energy consumption is fossil-based (per 2018, electricity sector excluded) [1]. Of the world's electricity production, around 66% is fossil-based [2]. This results in greenhouse gas emissions. Analyses estimate that postponing climate action will end up several times more expensive than immediate focus and investments in low-carbon solutions [3][IEA through [4]]. EU as an example has committed to cut greenhouse gas emissions by 80-95% by 2050, compared to 1990 levels, making it the fastest cutter [3].

Within the energy chain, two options are - to reverse our energy dependency by reducing energy demand - or to break the bond between energy and greenhouse gas emissions. Both are valid solutions and both are being pledged. Electrification is an example of reducing overall energy usage, since electrical technology is more energy efficient than fossil. Another plus of the electrification is that sectors are given access to clean electricity from renewable energy sources (RES). Thus electrification plays a vital role for cutting the bond between to greenhouse gases and the energy chain. World electricity demand is expected to rise by 62% by 2050 [2]. The fossil share of electricity production is expected to decrease from current 66% to around 31% by 2050, where solar and wind will constitute 48% [2]. Especially in Europe, wind and solar is expected to account for 80% of the electricity mix within 2050 [2]. This illustrates the massive deployment of new variable renewable energy sources (VRES) to come. On the downside, introducing new electricity demand and increasing the VRES share of electricity production will induce trouble for power grids, in many ways new to traditional power grid operation, outlined in the next paragraph. One prominent solution is to deploy smart grid technology with smart local flexibility markets. The work of this thesis falls in under these categories.

A new energy era for power grids

Power grids face increased intermittency and at the same time an increased sensitivity to intermittency [5]. At the supply side, power grids worldwide encounter an increased share of variable power production related to RES, where a significant share of it is at decentralized level unlike before [6]. On the demand side, we expect an increased high-intensive decentralized demand of electricity due to electrification and increased energy usage. All in all, these are good actions for our sustainable, carbon-neutral future, however they impose new issues for the electricity grid, issues that disturb the security of supply [5]. The complexity of the future electricity system will increase rapidly, particularly at distribution level, and this will result in more local congestions [7]. An increased VRES on

demand-side introduces a two-way power flow in the grid topography. In addition, the magnitude of the demand peaks are likely to increase, which initially anticipates grid capacity upgrades [6]. Quick changes in the power supply or demand which disturb the balance is also known as ramping. The new power grid trends will cause more frequent and intense ramping situations because of variable production, both at national level and more at local level - which will cause costly damages and blackouts unless the grid is made more flexible to handle it.

Balancing the grid have thus far been handled with traditional regulation methods. Regulation at the transmission level has many smart market-based mechanisms with a variety of backups, called reserves. A reserve is just another term for a major flexible source that can offer up- or down-regulation at transmission level when needed. Reservation and use of flexible reserves has mainly been a privilege for transmission system operators (TSOs) [8]. This have worked so far, but the new trends and the fact that the power flow changes from one-directional to bi-directional, requires more active approaches from the distribution grid operator (DSO) as well [8].

DFS markets

More local problems give rise to the idea that local problems must be solved locally. With digitization and new methodologies, the intelligent market-based operation methods at transmission level can be extended into the distribution grid and to the end-users. EU know this and has declared market-based congestion management as default for future operation of the grids, both for the TSO and for DSOs [3]. There are now many pilots and working cases on the rise in the field of market-based distribution grid operation. New smart flexibility market initiatives for DFS aim at offering local flexibility to DSOs for regulation of the grid and to others in need of flexibility [9]. NODES [10] and GOPACS [11] are two examples of platforms for such local flexibility markets. Utilizing DFS through local flexibility markets can be a key to shift and shed demand, providing a tool for solving local congestions and extreme ramping [4]. Demand-side flexibility can either regulate power up or down when it is necessary and the fine locational granularity of DFS is of uttermost importance [12]. The need for new DFS is well-documented. One nice and comprehensive article to read about this is *Flexibility in the 21st Century* by Cochran et al. [13]. Markets for decentralized flexibility that have the rightful design will both give added value to already existing DFS out there in addition to give incentives to further flourish DFS [8]. It is of importance that the design and development of such platforms are purposefully designed and is valuable for ALL participants. In addition to deploying smart and efficient marked-based operation methods, new capacity will be built in order to increase the shared pool of reserves which is a way of increasing the overall flexibility of a grid [4]. This in turn, will

expand the reach of flexible resources and probably add value to DFS with the right trade-off.

Management of decentralized flexibility in a building and trading on local flexibility markets requires expertise on the field. This is a task that smart grid companies often are engaged for. The smart grid company is called an *aggregator* if they trade DFS volumes on flexibility markets on behalf of a prosumer. It is of importance that the smart grid company has a methodology for assessing a building's flexibility in a precise manner. Being able to monitor and control flexible assets is necessary as well. eSmart is such a smart grid company, and ASKO is an owner of a building with flexible assets. In a reliable fashion, the smart grid company will offer the building's flexibility on a potential flexibility market on behalf of the building. NODES is an example of such a flexibility market platform.

1.2 Motivation

The recently mentioned need for distributed flexibility and flexibility markets makes this an interesting and new-born field to dig deeper into. The overarched goal is to identify and assess flexibility in demand-side assets so that it becomes accessible for smart regulation of the distribution grid. For that to happen, new methodologies are needed, a need confirmed by eSmart. The core motivation of the work in this thesis originates from this need. The success of flexibility markets depends on flexibility bids that are precise. Precise flexibility bids require precise flexibility estimates. In addition to estimation, there are challenges related to shaping of flexibility bids and verification of deliverance.

There are many who has done work on quantifying flexibility. An article by De Coninck & Helsen from 2015 [14] showed that there were no common metric or indicator for quantifying flexibility. They proposed a method to do so, using cost curves which indicate costs for deviating from the planned load. Barth et al. [15] proposed an optimization algorithm for quantifying flexibility by simulating all valid paths for the consumption throughout a day, but the bidding considerations are left out. Ottesen et al. [16] has proposed optimization models for bidding and scheduling of flexible demand-side loads. Much of the literature propose optimization algorithms and optimization models for intelligent load control. To my present knowledge, no literature looks into directly estimating short-term flexibility by using load forecasts with flexibility markets and bid shaping in mind. Although optimization methods may very well work, my motivation is to investigate a novel approach.

The work in this thesis is aimed at finding a methodology for assessing demand-side flexibility for flexibility markets, where some parts of the work can be beneficial to other applications. Accurate load forecasts are of great value not only for the methodology developed here, but also for cost-optimization problems. Estimates on available demand-side flexibility are of interest to anyone who might use it.

1.3 Problem statement

In an attempt to address some of the necessary technical challenges related to assessing flexibility, most of the focus in this thesis is to develop a methodology for estimating short-term flexibility in assets for the making of flexibilioty bids. As a reference for the work in this thesis, ASKO is used regarding which flexible assets to look at and NODES is used for the formalities around the local flexibility platform. The goals in the problem statement are formed in joint discussions with my supervisors Stig and Heidi, whose knowledge in the field and about research has been of great value. A bullet list with the main goal and sub goals of this thesis are presented below, in order to summarize the scope of this master thesis.

The main goal for the work of this thesis is as following:

(M) Develop a methodology to assess short-term flexibility for a set of various flexible assets in a building, in order to generate a flexibility bid in conformity with a local flexibility market platform. NODES and ASKO are used as point of references.

The main goal has been analysed and is divided into several sub goals:

(S1) Conceptualize the workflow and steps required to achieve the main goal.

(S2) Develop load forecast models that enable accurate predictions of asset consumption.

(S3) Develop a method to model an asset, its properties and constraints regarding flexibility.

(S4) Identify the flexibility of a flexible asset. Model and estimate their flexibility up to 6 timesteps ahead.

(S5) Suggest a bidding procedure, discuss the advancement in time and conceptualize a bid activation procedure.

1.4 Tools, data, case and methods

A methodology for assessing short-term flexibility in five selected assets have been developed. A lot of experimenting was done during this development process. Many of the choices throughout the work of this thesis is based on a use-case.

The use-case is a cooling storage for storing cold groceries in Vestby, Norway, which is drifted by ASKO. eSmart is a smart grid company which intend to analyse and utilize the flexibility in ASKOs flexible assets. ASKO has multiple flexible assets such as a cooling system for the storage, PV panels and a water heater. Data on historic consumption for these three assets are provided with a temporal resolution of at least 15 minutes. In addition, ASKO has a backup diesel generator at idle and do perhaps plan to invest in a battery bank.

The foundation is now set for which assets to look into at a conceptual level. The flexible assets listed in table 1.1 will stay in the spotlight for the rest of the thesis. A walkthrough on how to implement the developed methodology for each asset specifically, is included. Then a use-case will test the feasibility of the methodology on real data of ASKO's cooling consumption.

Table 1.1: List of flexible assets in the scope of this thesis.

Asset	Abbreviation	Type
Water heater	WH	Consumption
Machine room (for cooling a storage)	MR	Consumption
PV panels	PV	Production
Diesel generator	DG	Production
Battery	BA	Storage

An important fact-finding was that the flexibility of an asset is expressed by the magnitude it can deviate from its original load. Since we want to predict the future flexibility, the logical line of thinking is that forecasts of the load are needed. Recurrent neural networks (RNN) is a type of machine learning models and has been experimented with to make load forecasts. Python is used as the programming language and the Python packages Keras and TensorFlow are used to implement RNN forecast models. A variety of RNN settings have been tested and experimented with. In addition, there are four different strategies for making multistep forecasts, whereas the one called *direct multistep forecasting* has been implemented. A variety of hyperparameters for the RNN architecture is tested, however due

to computational complexity, the main focus is not to optimize them.

For modelling of an asset, a simplistic physical model of an asset with an energy storage has been made. During the implementation process, many assumptions on asset parameters must be done, because the information is unavailable or empirical tests was not possible to conduct. Object-oriented programming in Python is used to implement asset parameters in an asset class. The class is also used for flexibility estimates and to make proper visualizations of the estimated flexibility and the level of energy in the asset's storage.

før bakgrunnen kommer, bør intro sammenfatte klart målet med oppgave, hva som gjøres og hvordan. og hvilke verktøy som er brukt.hvilke ulike deler som består av hva.

Chapter 2

Theory

The theory chapter is structured in three parts. The first part is about the physical power grid and will present definitions on flexibility. The second part presents existing energy markets and the concept of novel flexibility markets, including NODES. The third part contains all theory for load forecasting and practical implementation of recurrent neural networks.

2.1 Electrical grids, power and energy

2.1.1 The physical power system

The pure purpose of power grids is to make sure electricity is transported from wherever it is produced to wherever it is consumed. It consists of a complex interconnected system of generators (or producers/supply) and loads (or consumers/demand). Some are even both producing and consuming power, named prosumers. Everything is interconnected through a system of high-voltage (HV) and low-voltage (LV) power grids and voltage transformation stations.

Power grid topology
The power grid consists of different voltage levels such as high voltage (HV) and medium/low voltage (MV/LV), associated with the transmission grid and distribution grid respectively. The transmission grid covers a large geographical area, typically a country or large state, with high voltage to reduce losses and ensure high transmission capacity. Distribution grids operate at lower voltage levels to

ultimately supply the end-user. This grid system forms a hierarchic system, with transmission grids on top, followed by distribution grids, buildings and assets, as illustrated in fig 2.1. For the rest of this thesis, the positive power flow is defined to go in the direction of decreasing voltage, indicating consumption. In general, negative power, or negative consumption, is regarded production. Transmission grids are operated by a Transmission System Operator (TSO), which is Statnett in Norway, while each distribution grid is operated by a Distribution System Operator (DSO), e.g. Hafslund in Norway.

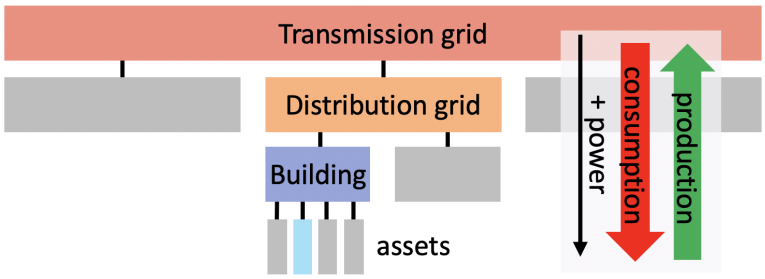


Figure 2.1: Illustration of a typical national power grid, including definitions of positive power flow direction, consumption and production.

A distribution grid contains several buildings beneath it, whereas each building usually connects through a metering device. These were all smart meters in Norway as per 1st of January 2019 [17]. A smart meter measures two-way power flow of what the building consumes and/or produces, used for quantifying bought and sold electricity. Digging deeper, a building could have several energy units below the smart meter, called assets. Each asset can be monitored by sensors, and such technology together with smart meters, ICT and control technology is a part of the digitization that enables intelligent DFS utilization.

Law of balance and inertia

The fundamental law of power systems is that net generation has to equal net consumption at all times, or at least over an averaging window. This is in order to maintain the system frequency, which is 50 Hz in most of the world. System frequency will rise or decrease in case of imbalance and it is the system operator's responsibility to keep it within an allowed deviation, e.g. $\pm$ 0.1 Hz. If this limit is exceeded, it can cause damage to equipment or even lead to blackouts. How sensitive a power system is to sudden balance disturbances, e.g. major errors in the power grid, can be characterized by inertia. A high inertia indicates a higher capability for the system to keep doing what it is currently doing, also referred to as transient frequency stability [18], which leads to a slower reaction to disturbances.

Rotating generators, such as hydro or thermal power plants, contribute with inertia to a grid with their momentum, whilst solar power does not. System inertia is good to have, but even grids with high inertia will eventually face frequency trouble in case of a sustained system imbalance, albeit in a less urgent manner. One of the downsides of a power system with a high share of VRES is the low inertia and the variability. Frequency is mostly balanced by the TSO, whereas the DSOs mostly ensure a stable voltage level and a reliable power flow in the LV grid. This has thus far been handled through good regulation of the HV grid.

2.1.2 Regulating the power balance

Having presented the importance of system balance, there is a logical necessity for every grid to have mechanisms for regulating the power balance. Such mechanisms are reflected by the flexibility of a grid. Flexibility in a power grid in general can be many things. It revolves around the inherited ability of a grid to handle unforeseen incidents and imbalances, e.g. sudden ramping. This ability can be possessed by all participants in the grid. Ulbig & Andersson [19] has proposed a definition on operational flexibility:

"Operational flexibility is the technical ability of a power system unit to modulate electrical power feed-in to the grid and/or power out-feed from the grid over time."

High flexibility will cause a grid system to withstand major error events or ramping situations, thus sustaining a stable system balance. Power flexibility is not something new, but some may recognize it as regulation power or reserves. However, the relatively new term decentral flexibility has received a lot of focus in later time. Traditionally, regulation of the system was mostly done by centralized generators, which matched supply against demand. That was usually sufficient for operation of the entire power grid, but does not withstand the new complexity at distribution level. Centralized flexibility will still play a major role in the future, however the need for intelligent operation methods at the distribution level with use of DFS is evident.

The location of flexible sources matters, as electricity needs to be transported through connections with capacity limits. Upgrading infrastructure, such as capacity of transmission cables and transformers, is an effective countermeasure to increase overall flexibility of a grid. The reason for this is that transmission capacity become higher and the reach of reserves is extended. It adds options for balancing the grid. A wider group of participants in need of or offering flexibility gets access to a larger shared pool of flexible reserves. On the downside, upgrading

infrastructure is expensive, and will furthermore only serve a minority of incidents having extreme peaks and not add value to the normal operation. This fact results in a worse capacity factor and is not a very socio-economical and acceptable operation of the grid.

Interconnectors
The power infrastructure in Europe has come a long way, and power grids in different countries are interconnected. The undersea cable NordLink between Norway and Germany is soon finished, and UK will have cables to both Denmark and Norway by 2022 (Viking Link [20] and North Sea Link [21] respectively). If most renewable production can be regarded variable and dependent in weather, such interconnectors will increase production capacity, the share of loads and stability. Better connections will increase the variety of both power source types and geographical location. As a result, the probability of coexistent production will decrease which in turn provides stable generation. As a simple example, suppose the northern Germany has a high demand of electricity and that their only power source, namely wind, is absent. With great interconnection, the demand would be covered by wind outside the British coast, south-German solar power and some French nuclear power, all well-balanced by Norwegian hydro power. This describes a scenario where grid management is done at transmission level, from the top. Good connections will be essential in the future, however there are opportunities growing from the bottom.

Pulsating end-users
The share of decentralized flexibility grows in speed with the demand-side complexity. The future consists of active use and integration of DFS, where the lack of wind power in northern Germany potentially could be covered from flexible prosumers within northern Germany. Interconnecting transmission capacity seeks to match the supply to the demand. The new and necessary tool is to control demand and match it against supply with pulsating and dynamic end-users. The idea is that prosumers regulate power in the grid with innovative integration of their flexible prosumption, which is one of the visions of smart grids. Smart grids provides a cost-effective alternative to infrastructure upgrades and aims at optimizing already built capacity. This results in raising the capacity factor of existing transmission lines and a more efficient operation of the grid.

Many of the traditional regulation mechanisms at transmission level are already market-based. Energy markets are considered cornerstones for maintaining balance between supply and demand in liberalized grid systems. They can be characterized as intelligent. However, they are restricted to the TSO only and not for the distribution grids. Distribution grids in past did not need such mechanisms in the

traditional "one-way power flow, centralized production, simple end-users"-grid. EU has declared marked-based solution as the standard operation method in the in the future, for distribution grids as well [3]. Before taking a dive into energy markets and novel flexibility markets, some definitions for flexible power must be settled.

2.1.3 Flexible assets and definitions

All use of the term flexibility will refer to demand-side flexibility for a building and its assets. Assets in a building, represented in figure 2.1, are flexible if they inherit the ability to deviate from their baseline. The baseline must be defined before flexibility can be quantified. The baseline is what would be "the plan" in the work of Peterson et al. [22]. The baseline is "the reference" in the work of Coninck & Helsen [14], suggesting it to be the load schedule solution taken from a cost optimization model. FLexible assets must be monitored with sensors and their power must be controllable. Some definitions regarding the flexibility of a flexible asset are now presented. The definitions are inspired by Coninck & Helsen [14] and Pinto et al. [23] with some modifications to fit the methodology in this thesis. Note that power is described in terms of consumption, whereas negative power, or negative consumption, is production.

- **Flexibility:** The magnitude of power the asset can deviate with from its baseline consumption.
- **Baseline consumption:** Referring to the originally planned consumption of an asset for the next H timesteps. This can be a load forecast or an optimized load schedule.
- **Positive flexibility:** The ability to **increase** power consumption relative to the baseline, thus providing positive flexible power. The upper boundary is denoted maximum positive flexible power. Upward flexibility is an optional term.
- **Negative flexibility:** The ability of an asset to **reduce** power consumption from its baseline (or increase production), thus providing negative flexible power. The lower boundary is denoted maximum negative flexible power. Downward flexibility is an optional term.
- **Flexibility space:** The feasible set of allowed choices of flexible power.

2.2 Energy markets

Energy markets are considered a cornerstone for maintaining balance between supply and demand for power. The shared Nordic power system has several energy markets to ensure system balance, such as the Day-Ahead (DA) market, Intra-Day (ID) market and reserve markets. In the markets, producers and consumers sell and buy their way into achieving balance, ranging from days to milliseconds before operation. These markets represent intelligent regulation systems. The reserve markets offer close to real-time power regulation with trade and activation of balance reserves. These tools are however currently limited to the transmission system level. With rising complexity on demand-side, there is a need for more active, intelligent regulation at distribution level as well. There are new local flexibility markets on the rise with a goal to expand intelligent market-based operations into the distribution grids. The goal is to integrate unrealized demand side flexibility for more precise regulation at distribution level. Upgrades of infrastructure, curtailment of RES generation and shedding high-intensity industry are local options to achieve local system balance. However, smart solutions with smart grids and ICT, along with connected energy markets will yield higher system efficiency, stability and reliability, and fill the gap between supply and demand. This enables the transition into a low-carbon society in the future [4].

2.2.1 Current power markets

Day-Ahead market (spot market) is the main market for most of the physical volumes that are traded in the physical electricity grid. Before 12:00, all major participants need to place bids and schedules of production and consumption for each hour the next day [24]. Then, NordPool settles a system clearing price, which is determined by a trade-off between demand and supply. Individual area prices based on bottlenecks will add or subtract to the system price for each affected area [25]. They apply to large regional areas, hence do not take local grid problems into account. The ID market is a power trading platform which is closer to the real-time operation than the DA market. Participants left in personal imbalance after the DA market closure, can achieve balance through ID market trading. The ID market closes an hour before operation time [24]. Further imbalances that occur in the hour prior to the operation time are settled in the reserve/balance markets. Here, participants with flexibility offer regulation power that can be activated from within 15 minutes or even seconds before operation time. Hence, reserve markets are essential for securing the temporarily balance between supply and demand. The

reserve markets can be divided further into primary market, secondary and tertiary markets, in which reserves must be activated automatically within seconds or manually within minutes or 15 minutes respectively [24]. Participants outside the market will be able to offer their reserves and be remunerated for their regulation services [26]. Reserve markets are merely platforms developed for TSOs with tools to tackle predicted and unforeseen critical grid events.

The emerging complexity at demand-side cause local problems at distribution level, which must be addressed by the DSO. The current market design with DA, ID and reserve markets are not aimed at operating distribution grids. In addition, the current, intelligent market-based operation methods at transmission level are simply not granular enough to solve the arising local problems[12]. To solve local congestions and other management issues related to the distribution grid require more active approaches from the DSO [8]. Local problems must be solved locally.

2.2.2 Mechanisms for solving emerging local problems

Innovative operation methods at distribution level is a highly active research field. There are alternatives which shows that deploying novel local flexibility markets is not the only solution. There are various ways to utilize DFS. USA as an example has had many years of experience with distribution grid operation methods. There is demand response management (DRM) which aim at controlling demand-side consumption. Two subcategories of DRM would be direct control or indirect control, both being so-called top-bottom approaches. The first, direct control gives the DSO full access to control a flexible asset at demand-side, even shed its consumption, under given constraints. An example of such a mechanism is to include a contract module for dispatchable consumption, where the end-user is remunerated by a DSO that gain full access to shred/control their asset. Direct DRM can be implemented in flexibility market platforms as well. Assessing short-term flexibility is important for direct DRM as well, as it will give the DSO knowledge on how much flexible power they have dispatched. The second sub category of DRM, indirect or intelligent control, nudge the end-user to change the consumption behaviour by means of price signals. Price signals may be added as tariff modules in the electricity contract between the DSO and building. A building is given incentives to actively exploit price variations, through a cost-optimized load control system of their DFS. The indirect method has some limitations because it requires planning and predictions of grid problems, at least a day but often weeks in advance. Indirect DRM may therefore have trouble to respond to more urgent grid events. In addition, price signals may not give sufficient incentives for end-

users to invest in added flexibility in their assets. It is important to find solutions which also promotes more DFS, because that is needed in the future. It is believed that flexibility markets will add value to DFS and flourish it.

EU has declared market-based congestion management as default for future real-time operations. Their reasoning is that, the alternative apporach, which is administrative and cost-based where participants are obliged to help and remunerated for costs and forgone profits, is difficult to apply for DFS. The estimates that are needed for costs and profits, for a vast amount of prosumers at demand-side, is too complex to get accurate and highly case-specific [9]. Such a top-bottom approach is hence favoured for the bottom-up approach where the slogan is to let the market do the job.

Local flexibility markets
From now on, local flexibility markets are the focus. When presenting the concept of novel local flexibility markets, the reader may notice it draws parallels to reserve markets. Flexible markets aim at extending intelligent marked-based operation methods all the way into the distribution grid and end-users, which now has smart control opportunities due to smart metering, IoT and digitization. The idea is that buildings bid their flexible power to the flexibility market platform. Here, DSOs and others who might need to buy local flexibility can activate that flexible power. In some cases, larger regions and even a TSO might need such flexibility as well, as with the example on northern Germany lacking wind production. DFS reserves are however small in volume, which could be unpractical on the market. Therefore, some smart grid companies specialize in aggregating small flexible volumes into bigger ones, e.g. a whole neighbourhood. A smart grid company possessing such a role is called an *aggregator*. Aggregator is also a term used in general for a smart grid company that assess and trade a building's flexibility on markets.

All the pilots and initiatives in the field of local flexibility markets are results of the rising complexity at distribution level. Local granularity is a key word. As mentioned, reserve markets are limited to the transmission level, in addition to be restricted to major flexible volumes. DFS has a precise location in the distribution grid, which is important. Active use of DFS provide finer granularity for DSOs to solve local problems.

Initiatives in Europe
There are many initiatives in Europe that investigate flexibility markets as a tool for local grid operations. Some are pilots, however some have already been deployed at national level, such as GOPACS in the Netherlands, which is already proved valuable. Radecke, 2019 gives a nice overview of pilots and working cases

of market-based DFS solutions in Europe [9]. Many of the proposed markets also create incentives to utilize unused potential DFS. Some examples of pilots and operative flexibility markets in Europe are

- NODES (universal, pilots in Germany, Norway and soon U.K.)
- GOPACS (operative in the Netherlands)
- Bne Flexmarkt (Germany)
- SINTEG (multiple projects in Germany)
- Piclo (U.K.)

In the future, there could probably be many more market operators in competition with each other. In addition, each scenario probably requires a specialized market design in order to be optimal for the case. However, the general concept of a flexibility market platform seems to be set. NODES, one of the many solutions for flexibility markets, is used further as a point of reference for the formalities around flexibility market design, operation and flexibility bids.

2.2.3 NODES - A fully integrated marketplace for flexibility

NODES emerged as an initiative by NordPool and Agder Energi to address the concurrent challenges that impact distribution grids. The information in this section is based on a NODES white-paper [12], unless other references are cited.

NODES is "*an universal platform for local, flexible electricity markets with features allowing for connecting to other markets*". It aims to increase the use of decentralized flexibility, as the European ID and DA markets alone do not provide sufficient granularity for local congestion management nor allow integration of DFS. It also aims at increasing the amount of available DFS by adding value to it. A NODES platform puts local flexibility as **products** on a shelf - up for take for buyers. The product, or a flexibility bid, is tagged with a location and includes a price, a baseline, the amount of offered flexible power and a duration.

The market-design

The design of the NODES marketplace and its market players is illustrated in

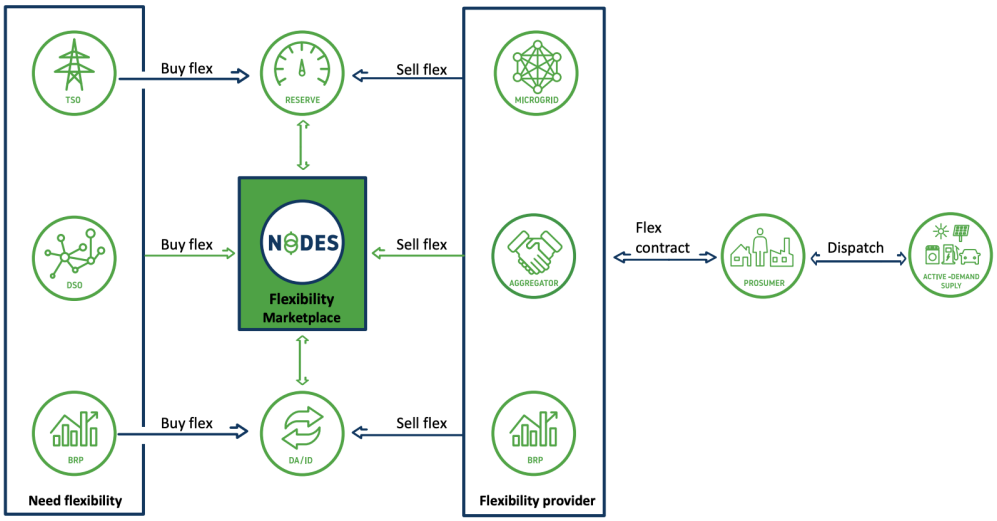


Figure 2.2: NODES marketplace and its various market players, mainly the flexibility providers on the right, and the ones who would need the flexibility on the left. Graphic from NODES whitepaper [12].

figure 2.2. The platform, as it is universal and meant to fit many scenarios, must be tailored to fit each unique scenario, in close cooperation with some thought market players. Fundamentally, the platform needs someone who can offer and someone who needs flexibility, e.g. prosumers and DSOs/TSOs respectively. A flexibility provider can be a smart grid company (or aggregator), with access to the flexible assets of a prosumer. They create a flexibility product and bid it into the platform. The product can then be bought by either of the flexibility buyers. If bought and activated, the flexible power should be dispatched accordingly by the provider. It could be positive of negative flexibility. Verification of delivered flexibility happens through the same platform.

The relevant market players included in the scope of this thesis are the DSO, aggregator and a prosumer. These are shown as circles in the figure. A setup with these market players is relevant for the use-case in this thesis, with the prosumer being a medium-sized industrial building possessing flexible assets. A piloting NODES platform often start out this simple, before it eventually incorporates more market players and extend the platform, in everyone's interest.

The aggregator will be important to bring DFS to the market. Aggregating smaller DFS volumes makes DFS more accessible. In addition, the aggregator will be responsible for flexibility estimation, bidding to the NODES market platform, dispatching of activated flexibility and verifying the delivered flexible power. NODES

will be operating the market platform and offer an Application Programming Interface (API) for trading. Both the flexibility buyer and the provider must be able to communicate with the API.

Advantages of NODES
The NODES market platform, as well as other similar platforms, serves multiple benefits. The first big advantage is that anticipated costly grid investments can be avoided. Secondly, local congestions can be solved more precisely. Although many of the new consumption assets impose problems, such as EVs, high-intensive appliances and demand-side RES, they also provide possibilities which can be taken into full advantage by flexibility aggregators and NODES.

NODES, in addition to other flexibility markets, claim to give incentives for buildings to promote and make use their potential flexibility, by giving their flexibility and increased value. Suppose that a building uses DFS for their internal use to exploit price variations. With NODES in addition, the building has more options to make profit from their flexibility. In addition, NODES platform want to expand local flexibility products into the reach of TSOs and other buyers that might need decentralized flexibility, thus further raising the value of DFS. A broader set of buyers means a higher value for the DFS. Different buyers also often need flexibility at different times. If the need would be coincidental, the flexibility will be used where it is of most value - ideally where it is most needed. Many possibilities for making profit of DFS will make it lucrative for prosumers to further realize unused, potential flexibility.

Another important feature of the NODES market platform is that it can connect to other markets in the future, such as the ID-, DA- and reserve markets. That would mean that a flexibility provider can access reserve markets more easily, so that they may buy back some balance. Suppose a flexibility provider is left in imbalance due to activation of its flexibility. The idea is that they should be able to automatically re-balance their portfolio through cheaper trading in other markets and still make profit. NODES, as an operator of the market, will make sure that all bids and activations do not cause new troubles elsewhere in the grid. All in all, NODES do not aim to replace any excising markets, but merely complement them to fully sustain a flexible smart grid all the way to the prosumers.

Working use-cases
The market-design of NODES is highly adaptable to different situations, locations and conditions to resolve a diversity of cases. The platform has already proven to be beneficial in real use-cases deployment both in Norway and Germany [27]. In Germany, NODES is used to relief an overloaded 110kV line, using flexibility

that is localized and exploited on the LV grid. In Norway, a potential overloaded transformer has postponed new investments, thanks to the NODES platform using flexible resources beneath the transformer.

Market flexibility products
Most of the market-based flexibility platform initiatives in Europe, including NODES, define a flexibility product to be the deviation from the baseline, either by consuming more or less than what was planned. Some have remuneration by availability, where flexibility providers get paid to have their flexibility at standby, similar to direct DRM. Others have remuneration by activation, where providers get paid per single flexibility activation. NODES and a few others, offer both remuneration methods [9].

NODES does not provide a specific product shape. Bids can look different for different flexibility buyers and for different use-cases. However, NODES suggests a modular design of a flexibility product. NODES support a contract to offer direct control of flexible asset, however the scope of this thesis focus on the competitive flexibility market platform. A product on this platform must at least consist of a baseline, offered flexible power, a time indicator, a price and the grid location of the prosumer. Resolution of the bid offers can be adjusted, but 15 minute resolution is often used. The focus of the work in this thesis will be on estimating the baseline and offered flexible power.

The forecasted baseline and the flexible power are constituted of several timeslots, denoted h. They indicate different times in the bidding horizon H. The bids in each timeslot could have several bid shapes. Some traditional bid shapes in the traditional energy markets can be linear, stepwise or block-based. A block bid consist of a constant volume and price, allowing for no deviation (also referred to as full activation), whereas a step-wise bid consists of multiple block bids. A linear bid consist of a continuous range of flexible power that can be bought.

2.3 Modelling

This section will present theory for timeseries, sequence forecasting models and especially recurrent neural networks. It will focus on practical implementation and application of RNNs for timeseries forecasting in Python.

2.3.1 Timeseries modelling

Sequences

A sequence, including timeseries, is data that are structured in a certain order, where the order matters. Moreover, a timeseries is data structured along a time axis, where a value is most likely to depend on prior values and to affect successive values. Timeseries can be represented mathematically as $\vec{x} = x^0, x^1, \ldots, x^t, \ldots, x^{T-1}, x^T$ starting at 0 in order to be Python index friendly. x^t represent a value of the sequence at timestep t, for a total of T timesteps. If there are multiple timeseries in the dataset, they can be distinct by a subscript, starting at 0 and counting, $\vec{x}_0^t, \vec{x}_1^t, \ldots, \vec{x}_i^t, \ldots, \vec{x}_n^t$ where n is total number of timeseries. Such a dataset containing multiple sequential features will form a multivariate dataset and can be nicely structured in a matrix, such showed in figure 2.3. The figure also gives some basic terminology for multivariate datasets, often used in machine learning, for later reference. Here, timestamps are optionally included in the index, in practise by using pandas DataFrames in Python. y also represents a timeseries and is the target we want to forecast in the future.

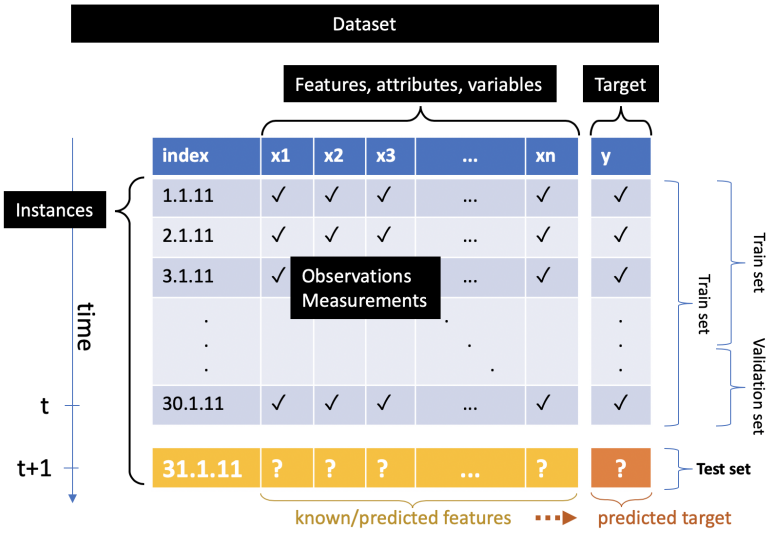


Figure 2.3: The terminology of a dataset used for creating machine learning models, here presented in a dataframe. This multivariate dataset has n features along the columns and has timestamps as instances along the rows, making it a timeseries. The figure also shows how the data is usually divided into train, test and validation splits.

From sensors to timeseries

Timeseries are not continuous because they stem from sensors that measure dis-

crete signals. How well they represent a continuous measurement depends on the temporal resolution, Δt, of the measured timeseries. Power is originally an instantaneous value, measured in W or kW. If power is measured each minute, these measurements could either be momentaneous measurements at each minute or the sensor could be sophisticated enough to provide an averaged value over the minute. Either way, the power is not momentaneous as it is assumed to represent a whole minute. The unit becomes kWh/h, as in an averaged power.

Downsampling is an expression that means to resample the temporal resolution of a timeseries to a lower resolution. For example, a timeseries of 1 minute resolution can be downsampled to 15 minute resolution. That is done by taking the average of each of the 15 1-minute measurements.

Sequence forecasting
Sequence forecasting, or sequence predictions, can be done by the means of various methods. ARIMA models is a well-established and widely used timeseries forecasting method. Another option is to make use of a multiple linear regression method (MLR) for forecasting [16]. More novel methods are deep learning methods in the field of machine learning (ML), such as neural networks, where especially recurrent neural networks (RNNs) are specifically designed for sequences.

According to Shi et al [28], RNN models have been shown to perform better at load forecasts compared to state-of-the-art techniques of ARIMA and SVM models. Others may mention they are equally good, which conforms with the well-known fact that there is no outstanding ML model to each unique forecasting problem. The promising potential of RNNs and the fact that it is a quite novel approach for forecasting load is the motivation for further exploring RNNs to perform the forecasting tasks in this thesis. The next sections present the process of building machine learning models, followed by RNN theory.

2.3.2 General machine learning

ML is a field of data science, which differ from traditional programming algorithms in one specific way. Instead of making rules to use on input data in the quest of finding answers, ML aims at using data and answers, in order to learn the rules. These rules can later be used to forecast future values.

The literature usually divide ML into three main subfields: supervised learning, unsupervised learning and reinforcement learning. Supervised learning is the relevant subfield for the work in this thesis, e.g. for making load forecasts. It describes

modelling machine learning models that are trained on datasets where the target is known, in order to predict future targets.

Supervised learning
One of the main goals of supervised learning is to learn patterns in a historic dataset such that we can make correct predictions and decisions in the future. To learn these patterns, the algorithm needs to have features and the solution (target) as input. It learns trends between the feature dataset and the target, ref $x_1, x_2, ..., x_n$ and y respectively in figure 2.3. Once the model is correctly trained, predictions can be made for the test set target, or unknown target, e.g. a future timestep. The features in the test set must be known in order to predict the final target.

There are several models in the field of supervised learning, such as multi-linear regression (MLR), logistic regression, artificial neural networks (ANN), recurrent neural networks (RNN), decision trees, random forests, etc.

Steps to creating a model
Figure 2.4 illustrates the steps in creating a supervised machine learning model.

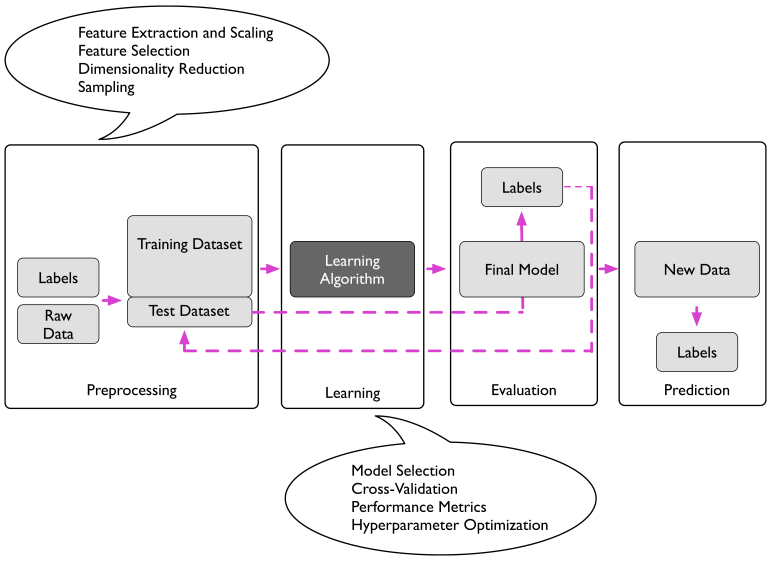


Figure 2.4: The process of building a machine learning model. Figure from book *Python Machine Learning*, s. Raschka, V. Mirjalili [29].

Preprocessing of raw data and quick analyses
After data has been gathered, one would very much like it to be perfect. However,

this is rarely the case. It either contains holes or errors - meaning that there are missing values for some periods and variables or that the observations are be wrong, respectively

As a data scientist, understanding the data, making sure that it is correct and preprocessing it is as important as making machine learning models itself. Garbage in usually means garbage out. There are many different tools to analyse the raw data; by the means of correlation plots, pair plots, histograms, checking against physical relations and checking against assumed statistical distributions.

Regarding missing data, there are several techniques that can fix it. Neither technique is outperforming another. Depending on the problem, the data scientist should investigate which technique yields the best results. One technique is to remove rows (instances) or features (columns) which has missing values. Another technique would be to predict the missing values, by the means of imputation, interpolation or others.

Lagged values as features:
When creating lagged values as new features to the model, the target sequence is taken, and shifted k timesteps ahead, $x^t_{lag-k} = x^{t-k}$, for all t. One problem will arise, namely that the new lagged feature lack values for the k first timesteps. This is easily solved by removing the first k observations. This process can be done for several ks.

Exogenous time variables:
These are features based on the timestamp of an observation, and i.e. the year, month, day, hour, minute, the day of the week, week of the year and so on. It could provide valuable structural information of seasonal and time-dependent trends, if any.

Learning
Preprocessed data is split into a train and test set, perhaps a validation set before input to the learning process. The model is trained with a learning algorithm. This step lays the foundation for which model should be selected and what model parameters should be chosen, based on model performance. Cross-validation is a method for assessing the performance of each model, by validating each model to unseen validation data to avoid overfitting. Learning is individual for each and one machine learning type.

Evaluation
Evaluation for model is done to measure the performance of the model and how good its forecast results are. Evaluation metrics are provided later in the multi-step

forecasts section.

Overfitting
Overfitting is the case when a machine learning model is not generalized enough to perform well on new unseen data. This happens when a model is trained very well to training data, often prone to its noise, but results in bad forecasts for test data.

Final prediction
This is the step for real-time forecasting. The model is fully trained and in operation. When new data become available, the model could be retrained with the new observations.

2.3.3 Recurrent neural networks

This section presents theory for RNN with focus and practical application. RNNs can be implemented in Python with the packages Keras and TensorFlow. The theory includes a high-level explanation of model architecture and the different parameters of the model in addition to how the training and test data should be set up for training and forecasting.

All of the theory on recurrent neural networks in this subsection is based on the two books *Python Machine Learning* by S. Raschka and M. Vahid [29] and *Deep Learning with Python* by F. Chollet [30], unless else is mentioned.

Recurrent neural networks (RNNs) are widely used in many sequence applications, such as:

- classification of documents or timeseries, e.g. determine topics or authors of books
- comparison of timeseries, e.g. investigate the relations
- sequence-to-sequence learning, much used in language translation
- analysing sentiments, e.g. classify the mood of a text or music (happy or sad)
- timeseries forecasting, e.g. predict electricity consumption

RNNs can be quite complex with advanced mathematical relations. The process

and architecture of learning and prediction in RNNs do require time and effort to understand. Yet, RNNs can be fast and easy to develop in Python using developed packages such as Keras [1]. The tools offer a kind of "plug-n-play" for building neural networks with set standards, while still offering access to advanced modification settings. In this way, packages in Python require little in-depth knowledge about neural networks, yet offering their power.

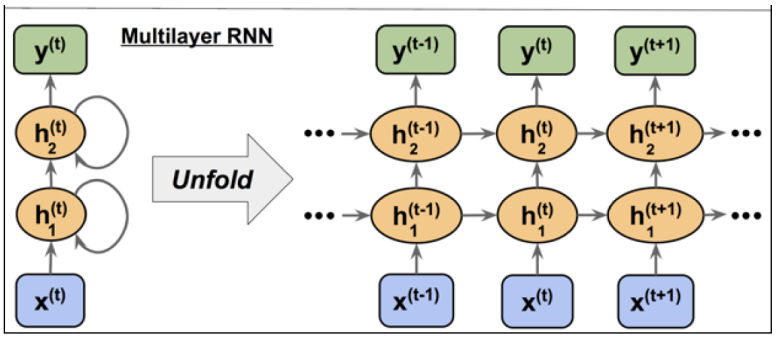


Figure 2.5: Architecture of a multilayer RNN, where the arrows indicate flow of data. Figure from book *Python Machine Learning*, s. Raschka, V. Mirjalili [29].

This explanation of RNNs and how they work is meant to avoid the complicated equations. More technical and thorough information on the topic may be found in the ML books suggested above. The following theory focuses on the practical application of RNNs for timeseries forecasting. As an example, the architecture of an RNN model with two layers is illustrated in figure 2.5. Here, x represents the input, being the features fed into the network which consist of two layers, h_1 and h_2, which makes the architecture. y is the output, or the target. All nodes are connected with weights, through functions and so on, where the weights are learned accordingly during training, just like an ANN. The most important difference with RNN over an ANN is the connections from prior timesteps, $t-1, t-2, \ldots$ to the current timestep, t, which allow the model to have memory from past results, to influence the next results.

All in all, the framework for an RNN is to build a dataset containing features and a target, construct the RNN architecture with optimal number of hidden layers, consisting of layers such as LSTM, GRU or Dense. Choosing between either of those results in different properties of the architecture. The layers are explained here:

[1] https://keras.io/getting-started/sequential-model-guide/stacked-lstm-for-sequence-classification (accessed 12/12/2019)

LSTM (Long-short-term-memory) unit: Type of node unit, designed to overcome the vanishing gradient problem. It designed to carry information across many timesteps and preventing older signals to completely vanish, hence it contributes to the network by giving it a longer memory of the past. It is complex and combines different functions with different input and output combinations.

GRU (Gate Recurrent Units): A bit simpler architecture than LSTM, and is more computational efficient, due to fewer parameters. GRU layers may have better performance on smaller datasets.

Dense layer: Ultimately takes in all inputs from the previous layers and outputs a forecast, which is either a value or an array of values in the case of a multi-output.

Hyperparameters
In addition to building the architecture, there are hyperparameters such as batch size, number of epochs to train on and learning rate that influence model performance. The construction of data is of much importance as well. As with much else in the field of machine learning, there is no superior model to all problems, regarding choice of model, its architecture, its hyperparameters, data preparation etc. Therefore, finding a good model is experimentation itself and unique to each problem.

Optimizers
An optimizer is needed for training an RNN model and needed for finding optimal weights of a model during learning. Various optimizers can be chosen, such as adam and RMSprop. A brief introduction to optimization, with a comparison between adam and RMSprop can be found on the webpage in the footnote [2]. RMSprop is a simpler version of adam. Adam has been most used in the thesis and the standard optimizer for RNNs in Keras for Python [3]. More on the adam optimizer can be read in the article by Ruder [31]. Optimizers affect the success of the learning process and whether or not the optimal solution is found. It also affects the time of the learning.

Preparing the dataset for RNN training
During training of an RNN model in Python, the Keras model expects the training input data to have the following 3-dimensional shape:
[**samples, timesteps, features**]

This 3-dimensional matrix must be generated from the basis 2-dimensional dataset,

[2]https://blog.paperspace.com/intro-to-optimization-momentum-rmsprop-adam/ (accessed 12/12/2019)
[3]https://keras.io/optimizers/ (accessed 12/12/2019)

like the one shown in figure 2.3, which has the shape [**samples, features**] . The basis 2-dimensional dataset represents the whole training and test data foundation which has been prepared and preprocessed. When the RNN model is trained in Keras, the *timesteps* dimension is introduced. This dimension represent how long the memory of the RNN network should be.

During training, many smaller batches of the training data, having the 3-dimensional shape, is fed into the network. Each batch generates a line of predictions in the train set. The amount of *samples* is set by the *sequence length* parameter and the amount of timesteps to form is set by the $N\_lOOKBACK$ parameter.

As an example, a batch with input shape [1100, 7, 1] means that a training batch of 1100 observations from the training set is chosen, in a kept sequential order, is fed into the training network. It has 1 feature in the training set, e.g. the consumption. 7 timesteps mean that the model shall use all 7 steps in the foundation for learning and predicting.

Each training batch fed into the learning algorithms, results in a forecast being made. This is compared against the true value. Based on this comparison, the learnable weights of the RNN model are adjusted in order to improve the next forecasts, through backpropagation.

When making a prediction for the test set, Keras wants only on batch with the 3-dimension shape. For final prediction, sample=1 and the input data shape is namely [1, timesteps, features]. The output forecast will be [1, 1]. If the model is created so to output a multi-output forecast of H steps, the shape of this input matrix would have to be [H, timesteps, features]. It will then output an array of forecasted values of dimension [H, 1]. This array is a forecast for the forecast timestep made for all timesteps, $t + 1 \forall t \in \{testset\}$.

In order to create many batches for iterative learning for the whole train set and to finally make predictions for the whole test set, various *batch-generators* are used. In the work of this thesis, a batch generator has been borrowed from an open Git repository [4], and the others are included in Keras, named TimeseriesGenerator.

Scaling:
In order to improve model performance, RNNs benefit from having data scaled to the range between 0 and 1 or standardized. This can be done using *MinMax-Scaler()* provided by the sklearn package for Python.

[4] https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/23_Time-Series-Prediction.ipynb

Encoding categorical features to dummies:
Dummies are needed to avoid letting the model think that a Sunday (value 6) has a higher value than Monday (value 0), and such, dummies is needed to be made, by constructing a new feature for all of the values in the old feature, i.e. Monday,...,Sunday are the new dummy features which have values 0 or 1, where a 1 assigns this feature to the given observation. Dummy creation can be done easily with pandas in Python.

2.3.4 Multi-step forecasting

As to this point, models were thought to predict only one timestep ahead. The goal is to make forecasts for multiple successive steps ahead. That means forecasts each of the forecast timesteps, $t+1$, $t+2$, ..., $t+H$, that we would like to forecast. A summary of various strategies for multi-step time series forecasting is given by Bontempi et al. [32] and reviewed here:

1 *Direct Multi-step Forecast Strategy:* One unique model is made for each forecast timestep.

2 *Recursive Multi-step Forecast Strategy:* The same one-step model is used recursively, where the prediction for the prior time step is used to predict the next one.

3 *Direct-Recursive Hybrid Multi-Step Forecast Strategies:* Combines the two above strategies, e.g. creating individual models for each timestep ahead, where each model also uses predictions from models for prior timesteps. This may help overcome limitations of (1) and (2).

4 *Multiple Output Forecast Strategy:* Only one model is created and trained to make a forecast for all timesteps at once.

(1) Direct Multi-step Forecast Strategy
For the work in this thesis, only the direct strategy has been implemented properly. For this strategy, multiple models are created in order to specialize on each of the lead timesteps that are to be forecasted, h. This happens relatively straightforward. Many one-step models are used. Each model will use the same train set, X. There will however be different targets array per model, that are shifted one and one timestep from each other in lead time.

Evaluating multi-step forecasts
These evaluation metrics are used on regression problems, which is relevant for this thesis. The most common ones are Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mean Square Error(MSE), Root Mean Square Error (RMSE) and R Squared. The ones used in this thesis are mathematically expressed as following:

$$MAE = \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{n} \tag{2.1}$$

$$MSE = \sum i = 1^n \frac{(\hat{y}_i - y_i)^2}{n} \tag{2.2}$$

A lower MSE or MAE value is a better score. Computing the score for each forecasted timestep is often helpful. It gives valuable information on how a model performed on each seperate forecasted timestep. A summarized score can be computed as the average score across all forecasted steps.

2.3.5 Techniques for fighting overfitting

Some popular techniques to fight overfitting in neural nets are as following.

Validation data:
During each training iteration, the model performance can be checked against unseen data, named validation set, as seen in figure e2.3. Validating the training on an unseen validation set rather than the train set, will cause to model to not try overfit parameters for the train set.

Early stopping:
This technique is applied during model training. It monitors the validation score of each epoch, and if the validation score has not improved during a number of successive training epochs, called *patience*, it stops the training. Early stopping is often used together with model checkpoints, which stores the latest best model. Using early stopping and model checkpoint enables us to reload the best model from the point it stopped improving with respect to validation loss.

Dropout:
Dropout is a technique to randomly drop a certain share of units (and learnable weights) in a layer and in such way "thinning" the neural network. Their aim is to prevent units to be prone to overfitting. In Keras, the Dropout method is applied by adding a Dropout layer after a layer that shall be thinned, e.g. after a LSTM layer. Correspondingly, a share of some random units will be dropped by

setting their unit output weights to 0, thus forcing use of other weights. The Dropout technique has in one case shown clear improvement over other regularization techniques [33].

Some techniques work better than others in certain cases, and sometimes the best result comes from a combination of them. Various techniques should be used with prudence in order to find the optimal method for the specific problem.

Chapter 3

Preliminary methodology for assessing short-term demand-side flexibility

This chapter explains the concept of the developed methodology for assessing short-term flexibility for an asset. It starts with a light explanation to let the reader get familiar with the overall methodology. Then, the methodology and its constitutional stages is thoroughly explained for a generic asset, including a bid event line example to demonstrate its uaage during time advancement. The last part involves more specific descriptions on how the methodology is thought to be implemented for each of the five chosen flexible assets that were presented in table 1.1.

3.1 Presenting the methodology

To this point, various theory has been presented, which is now finally stitched together into a streamlined workflow, which is the proposed result to the main goal **(M)** described in the introduction. The methodology proposed in this chapter aims at assessing the short-term flexibility in flexible assets in a building, where the ultimate purpose is to construct valid flexibility bids into a flexibility market. The process ends the bid being either activated or not, before the methodology repeats. For this methodology, the NODES flexibility market platform serves as a point of reference. An illustration of the workflow for the preliminary methodology is shown in figure 3.1. This figure will be referred to frequently. As illustrated, the workflow is composed of four distinct stages.

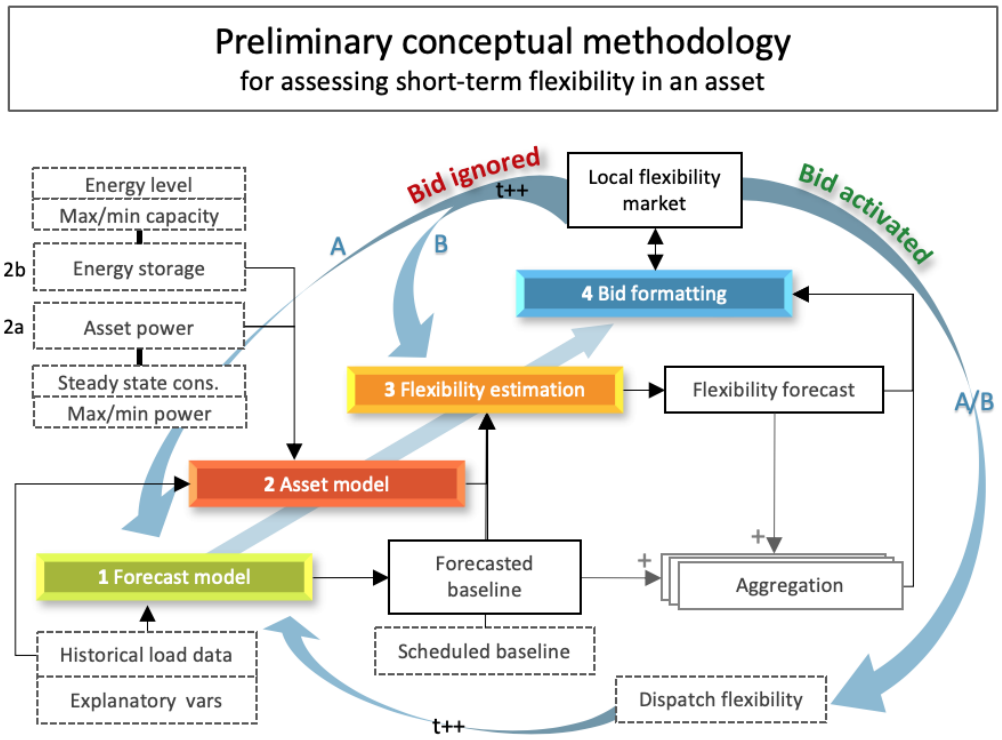


Figure 3.1: Workflow and the stages of the preliminary methodology for assessing short-term flexibility in a flexible asset.

A brief description of the stages

A generic, high-level description of the preliminary methodology will now be presented, before each stage is explained in detail later. The methodology involves the use of timeseries data, forecasting models such as RNNs, physical models of the assets, object-oriented asset modelling in Python and bid formulations.

As we will see, and as previously defined, the flexibility is based on the baseline consumption, and therefore a **load forecast model (stage 1)** is the first stage in making flexibility estimates. The load forecast model output a *forecasted baseline*. In some exceptions, an asset does not require a complicated forecasting model in order to know its future baseline. An example is an optimized load schedule for battery, where the forecasted baseline is simply a *schedule* or the plan. In other cases, the future baseline must be forecasted. The electrical consumption is in many occasions influenced by external parameters, e.g. temperature or building activity. A load forecast model uses *historical timeseries* data as input, in addition to any *explanatory parameters* which may improve predictive power of the forecast model. The goal is a model with high predictive accuracy for unseen data points, especially the first H timesteps in the bidding horizon, e.g. $H = 6$. A forecast model is developed and once a good model has been chosen, it outputs a forecasted baseline, $\hat{\vec{y}}$ or $\hat{\vec{b}}$.

Flexibility itself lies all in the asset and its characteristics, thus calling for an **asset model (stage 2)** . The asset has either flexible consumption, generation, storage or a combination of the three. In addition to a *baseline forecast* provided from stage 1, it is necessary to investigate and quantify the inherited properties and constraints of the asset regarding its *power* (2a) and *energy storage* (2b). The *maximum*, *minimum* and *steady state power*, P_{max}, P_{min} and $P_{cons,SS}$ respectively, are important for determining the magnitude of flexible power. Energy storage constraints puts limits to the available flexibility regarding time and volume. Provision of an *energy level* estimate, SoC, and the *energy capacity limits*, E_{min} and E_{max}, are necessary. This methodology does not provide a method to quantify such asset parameters, which may be a complex task to do. The asset parameters are nevertheless crucial input for the asset model in the task of estimating flexibility. A State-of-Charge (SoC) modelling framework is used to physically model a flexible asset with its parameters, power flows and states. Implementation is done by means of object-orientated programming in Python.

Until this point, all necessary values are provided in order to calculate **flexibility estimates (stage 3)**. The forecasted baseline is input to the asset model and

combined with the asset parameters, in order to calculate of the *flexibility forecast*. A simple, yet effective way to assess the complete range of available flexibility is to calculate the upper and lower flexibility limits, denoted maximum positive and maximum negative flexibility respectively. When this stage is done, the maximum positive and negative flexibility, $\hat{F_p}^{(h)}$ and $\hat{F_n}^{(h)}$ respectively, is estimated for each timestep in the bidding horizon. Simulation of the energy content levels reveal if the energy capacity limits are reached when activating F_n or F_p. The estimated flexibility is therefore shredded accordingly, if energy storage limits are reached. This stage also visualizes the flexibility estimates and the possible energy storage trajectories different choices of flexibility activations.

After available flexibility has been estimated, the next process is to **format a flexibility bid (stage 4)** to be offered on the *market*. Formulation of a bid has many alternatives and should be developed with close cooperation with the respective participants. One choice is for the DSO to fully activate all negative flexibility at once, which will probably completely discharge the assets energy storage. Another choice could be that the DSO activates just a portion of a bid, and thereby extending the duration. Another choice is for the building itself to withhold their flexibility fully or by parts, which they can freely do, for example if they believe the revenue is higher a couple of hours later. Since there are many opportunities, the bid formulation stage must be done with prudence in order to find the optimal solution for all parts. The bi-directional arrow between *bid formatting* and the *Local flexibility market* indicate that the flexibility buyer can give feedback to the aggregator regarding the bids.

When a flexibility product has been offered to the market platform, there are two possible **outcomes**. An **ignored bid** means that the bid is not bought by any flexibility buyer and the building must prepare new flexibility estimates for the next bidding period. t++ symbolize incrementation of time, indicating advancement to the next bidding period. *Pathway B* represent an update of flexibility estimates without retraining of the forecast model, whereas *pathway A* represent a full update of flexibility estimates including retraining of the forecast model. Typically, the frequency of model retraining needs to be found by experimentation and is limited by computational complexity. In the case of an **activated bid**, the building is committed to dispatch the bought amount of flexible power. The success of this dispatch process relies on the asset control system. As the dispatch process goes on, no bids are made until the end of the dispatch period. The process of making new flexibility bids then starts over with a repeated workflow cycle. One exception is in the case of rebound effects and rest time, which mean that the flexible asset must recover from the dispatch process. Monitoring of the dispatch process will be used in the aftermath to determine the level of dispatch success and potential

penalty fees.

Side notes:
Aggregation: Currently, it is unknown whether the flexibility buyers at the NODES market platform would like individual flexibility products from each asset or an aggregated offer of available flexible power frmo mutliple assets. It could also be that different participants are in need of different bid aggregation levels, thus reflecting the need for multiple bid offers to NODES tailored for different flexibility buyers. Whether or not an aggregation must be done by the flexibility provider itself or on the NODES platform is currently an open question.

Optimised load control: It should be mentioned that this model does not provide any decision-support nor instructions for smart control system. The workflow has as a goal to generate bids from flexibility forecasts, conducted from combining high-accuracy load forecast model forecasts together with information about asset properties and state. If the building is subject to an optimized smart control system, the baseline in stage 1 could be replaced with the optimized load schedule.

3.2 In-depth explanation of the methodology

3.2.1 Preparation

Parameters such as the temporal resolution of a timeslot, Δt, and the amount of timeslots in the bidding horizon H, should be defined first. Δt could be one hour or less. The bidding horizon decides the total number of timesteps, or time slots, in a bid such that $h \in [1, H]$. The temporal resolution determines the time between each timestep, or time duration of each time slot. The product of Δt and H is the total time duration of the whole bid (in hours). Temporal resolution could in theory be set freely, however a lower limit is in practice set by sensor delays, computational time and the temporal resolution of the timeseries. Unless anything else is mentioned in this short-term bidding methodology, $\Delta t = 1$h and $H = 6$ timesteps as standard.

3.2.2 Forecast models (stage 1)

This methodology will proceed with a direct multistep RNN model, which was presented in the theory chapter. All data is preprocessed in order to comply with a temporal resolution of 1 hour and the model is set to forecast 6 steps ahead.

Equations
The goal is to predict the baseline consumption of the asset. Historical timeseries of baseline consumption is used as a target $\vec{y}$ when training an RNN forecast model and its lags and additional explanatory variables are included in the feature set.

The model outputs a vector of forecasted consumption values, $\hat{\vec{y}}$, which is henceforth denoted as the forecasted baseline, expressed as following,

$$\hat{\vec{b}} = \left[b^{(\hat{h})}\right]_{h=1}^{H} = \left[b^{(\hat{1})}, b^{(\hat{2})}, \ldots, b^{(\hat{t})}, \ldots, b^{(\hat{H})}\right] \tag{3.1}$$

The vector of true consumption values $\vec{y}$ is henceforth denoted as the true baseline, expressed as following,

$$\vec{b} = \left[b^{(h)}\right]_{h=1}^{H} = \left[b^{(1)}, b^{(2)}, \ldots, b^{(t)}, \ldots, b^{(H)}\right] \tag{3.2}$$

Units are kWh/h. The true baseline is available in historical data, in both test and train set. In real-time operation, the true baseline will tick in as datapoints if bids keep getting ignored, however that is not the case with bid activation. In the time after bid activation, it is impossible to know what the true baseline would be because there is a flexibility dispatch going on. That should kept in mind for later.

Visualizing multistep forecast plots

Multistep forecasts can be visualized as in figure 3.2. It may take some time to fully understand what it represents; therefore, a thorough explanation is given. The plot shows multi-step forecasts, $t + h$ with $h \in \{1, 2, ..., H\}$ with H=6, made at *all possible* timesteps in the timeseries, $\forall t$. Ergo, there is one plotline for each forecast horizon step h. Each of the red lines represent different forecasted steps, t+1, t+2,..., t+6. All the forecast lines aim at replicating the true line. They do however have different data foundations. Each forecasted value on the t+1 graph is based on all previous data points. Each forecasted value on the t+6 graph does not have its 5 prior values available for forecasting, thus yielding a different forecast foundation. Ideally, all the forecast lines should thus be identical to each

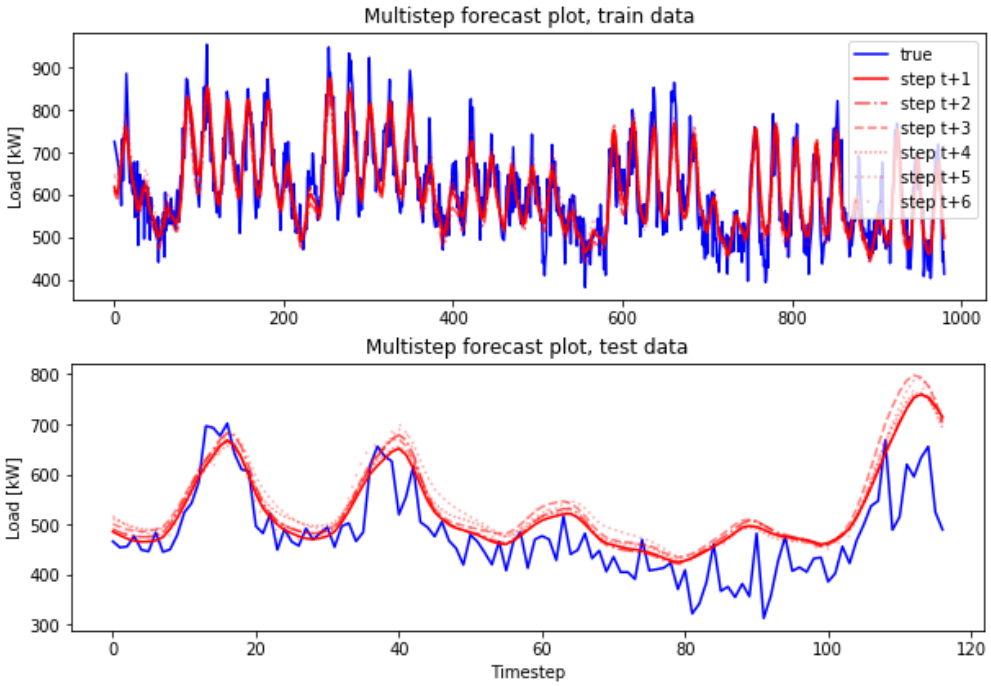


Figure 3.2: An example of a multistep forecast plot. One line represents forecasts at all current timesteps, t. Each line represents each forecast step in the bidding horizon, $t + h, \forall h$. To get an idea of how this plot is made, refer to the text and figure 3.3.

other and equal the true baseline in blue. The following figures may provide a more clarifying explanation.

Each of the graphs in figure 3.3 shows a multistep forecast made at *one single* timestep, in contrary to the previous figure. This is how forecasts look like in real time forecasting. A single multistep forecast lays the foundation for estimating flexibility. Start at figure 3.3 (a) with t=800. Forecasts, $\hat{b}^{(h)}$, are being made for each of the timesteps in the bidding horizon $h \in \{1, 2, ..., 6\}$, here step 801 through 806. They are each represented by red dots. The true baseline, $b^{(h)} \forall h$, is also shown. As time advance from figure 3.3 (a) to 3.3 (b), the forecasted red dots are updated on new available data, but still represent forecast for t+1, t+2,...,t+6, but with t=804. Trajectories for each of the red dots are drawn as time continues to advance from 3.3 (a) through 3.3 (d). The drawn trajectories illustrate nicely how figure 3.2 was made.

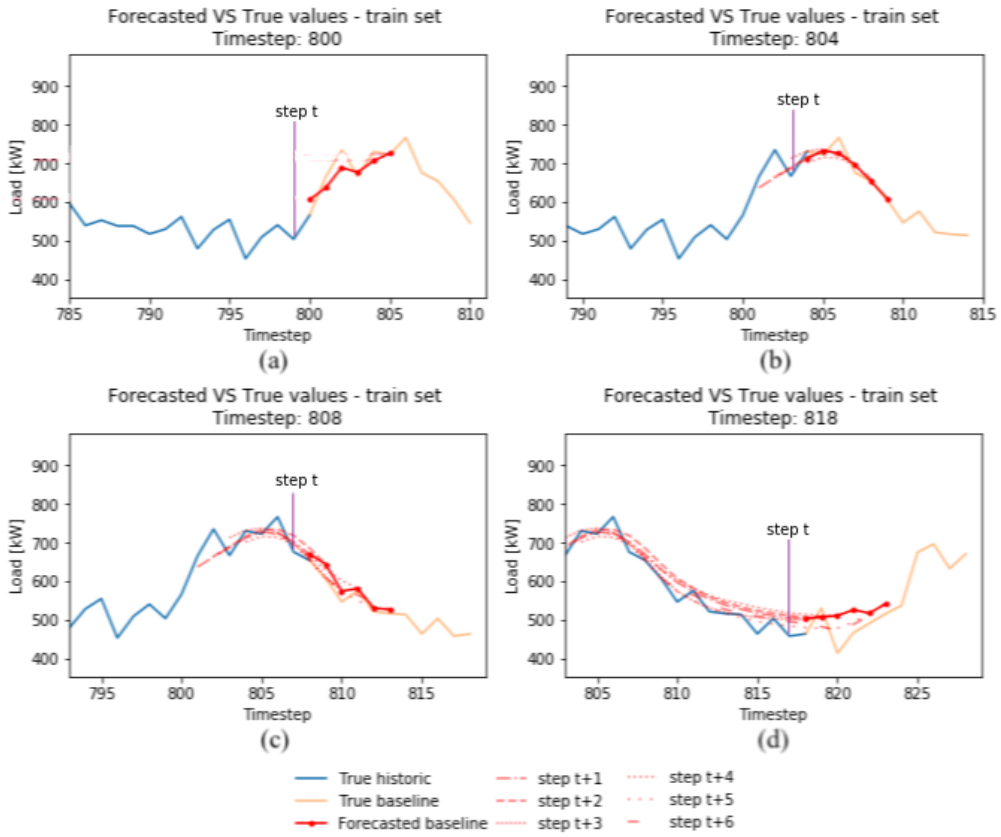


Figure 3.3: Real-time multistep forecast plots for different points in time. The red dots represent the forecasted values in the bidding horizon. True baselines are also plotted. current timestep= (a) 800 (b) 804, (c) 808 and (d) 818. As time advance, trajectories for each of the red dots that represent each forecast timestep, h, has been drawn to give an idea of how the multi-step forecast in figure 3.2 is made.

3.2.3 Asset model (stage 2)

Asset model with a State-of-Charge framework

A State-of-Charge (SoC) model framework is introduced as an unite framework for measuring the current charge level of an energy storage. The SoC approach has borrowed inspiration from Ottesen [16], Barth et al. [15] and Ulbig & Andersson [19]. The concept is introduced in order to avoid the confusion regarding positive and negative energy in energy storages, e.g. cold versus heat reservoirs. The SoC model normalize an assets energy capacity and current energy level to the SoC range between [0, 1]. Normalization is not implemented in this work. The normal-

ization factor provides a conversion factor, n, between ΔSoC and absolute energy ΔE. Along with the use of SoC, the terms *charging* and *discharging* will be used instead of increasing and decreasing energy content. Charging of an asset will increase the SoC, and vice versa for discharging. The asset consumption which leads to a stable unchanged SoC level is defined as the steady state consumption, denoted $P_{cons,SS}$. An increase or decrease of consumption relative to the steady state consumption results in charging or discharging the storage respectively. Cooling storages are charged when they actually decrease their energy content, because of added electrical consumption. That is the purpose of the SoC framework.

The normalization factor, n, is defined to be

$$n = \frac{\Delta SoC_{range}}{\Delta E_{range}} \xrightarrow{SoC \in [0,1]} \frac{1}{\Delta E_{range}} \tag{3.3}$$

where $\Delta E_{range} = E_{max} - E_{min}$ is the allowed range of energy levels in the storage. If the energy storage is thermal, then ΔE_{range} will be provided by equation 3.20 by inserting the maximum and minimum allowed temperatures. For a cooling storage, ΔE_{range} will be negative and $n < 0$. In this thesis, n is set to be either +1 or -1, which means that SoC is still measured in kWh and not normalized to the range [0, 1]. Further in this work, a $n = -1$ is used for the cooling storage and $n = +1$ is used for the other assets.

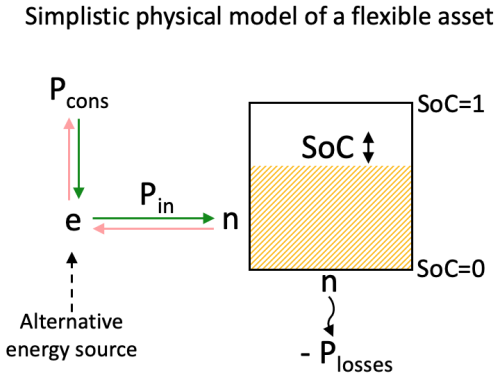


Figure 3.4: A sketch of a general physical model of a flexible asset with a flexible energy storage. The green arrows, and not the light red, indicates positive power direction.

A generic physical model of a flexible asset with a flexible energy storage is shown in figure 3.4. The SoC level and the relevant power flows are indicated. P_{cons} represents the electrical power consumption of the asset. The share of consumption that interacts with the storage is expressed

$$P_{in} = eP_{cons} \tag{3.4}$$

with $e > 0$ being a power conversion efficiency factor and can be lower or greater than 1. P_{in} can be electrical or thermal power depending on the energy storage being chemical (battery) or thermal respectively. Often, $e > 1$ for heat pumps or compressor systems, then often called coefficient of performance (COP). Or $e < 1$ for chemical or direct thermal power conversion. A positive P_{in} will in isolation increase the SoC level. In case of a cooling storage, a positive P_{in} do actually represent a negative power flow pure physically, but this confusion is now avoided. Another factor is the power losses, P_{losses}, which in isolation cause the SoC level to decrease. The loss power could represent an intended outgoing power flow to an external purpose, e.g. heating offices. It also includes unwanted energy storage power losses, for example heat losses. P_{losses} exclude electricity that flows back to the grid, as this is included in P_{in}/P_{cons}. The physical model also includes an option to have an alternative power production source, e.g. diesel generator or PV panels.

The net power flow into the energy storage is denoted as the charging power, $P_{(dis)charge}$. It is defined as the net power flow that interacts with the energy storage's system borders, expressed as following

$$\begin{aligned} P_{(dis)charge} &= P_{in} + P_{losses} \\ &= eP_{cons} + P_{losses} \end{aligned} \tag{3.5}$$

The charge power is important for the evolution of the energy storage level. A change in the SoC and energy level over time happens when the charge power is either positive or negative, expressed

$$\frac{dSoC}{dt} = nP_{(dis)charge} \tag{3.6}$$

Integrating at both sides from one discrete timestep to the next timestep yields

$$\Delta SoC = n \int_{t}^{t+\Delta t} P_{(dis)charge}(t)dt \tag{3.7}$$

which for discrete timeseries simplifies to

$$\Delta SoC^{(t)} = nP^{(t)}_{(dis)charge}\Delta t \tag{3.8}$$

where $\Delta SoC^{(t)}$ is denoted the change in SoC level at the timestep t, due to the charge power at that timestep, $P^{(t)}_{(dis)charge}$. Δt is the temporal resolution, the duration of a timestep (in hours).

Table 3.1: Table of asset parameters that must be determined in order to fulfill stage 2 of the methodology.

Symbol	Parameter	Unit	Description
Δt	Temporal resolution, positive integer	hour	example: Δt=0.25 imply 15min resolution
$\vec{P_{min}}$	a minimum possible asset power (2a), array of length H	kW (kWh/h)	
$\vec{P_{max}}$	a maximum possible asset power (2a), array of length H	kW (kWh/h)	
E_{max}	Maximum energy capacity, integer	kWh	
E_{min}	Minimum energy capacity, integer	kWh	
SoC_0	An initial SoC value, $\in [0, 1]$	-	
$\vec{\hat{b}}$	Forecasted baseline, array of length H	kWh/h	provided from stage 1
$P_{cons,SS}$	Steady state consumption, array of length H	kWh/h	is equal to for ex. 0 or the baseline

Having $P_{(dis)charge} = 0$ will have the following implication

$$P_{(dis)charge} = 0 \implies P_{cons} = P_{cons,SS} \implies \Delta SoC = 0 \tag{3.9}$$

where $P_{cons,SS}$ is the steady state consumption, defined to be the consumption that leads to an unchanged energy storage level. A steady state situation will according to eq. 3.5, lead to

$$P_{losses,SS} = -P_{in,SS} = -eP_{cons,SS}$$

The losses are assumed to not be dependent on the consumption, namely $P_{losses,SS} = P_{losses}$. Using the above equation with a reformulation of eq. 3.5, $P_{(dis)charge}$ can also be expressed

$$P_{(dis)charge} = eP_{cons} - eP_{cons,SS} \tag{3.10}$$

Asset model parameters

During the development of the methodology, it is found that the asset parameters given in table 3.1 must be determined. In order to determine some of these parameters, it may be necessary to conduct experiments and tests of the asset, e.g. step-tests.

The forecasted baseline, ref eq. 3.1, is provided by the forecast model in stage 1. The minimum power $\vec{P_{min}}$ sets a lower limit for the minimum possible power consumption of the asset. It can vary for different timeslots and is expressed

$$\vec{P_{min}} = \left[\hat{P}_{min}^{(h)}\right]_{h=1}^{H} \tag{3.11}$$

Similarly, the maximum power $\vec{P_{max}}$ sets an upper limit for the power consumption of the asset, and is defined as

$$\vec{P_{max}} = \left[\hat{P}_{max}^{(h)}\right]_{h=1}^{H} \tag{3.12}$$

The upper and lower power limits are found empirically or provided by the respective asset. Note that negative values always indicate production and not consumption. The charge power is calculated according to equation 3.10.

Object-oriented programming for asset modelling
For the implementation of an asset model, a class called *Asset* has been made in Python. Making a class is beneficial, because one can create multiple objects from it, e.g. many assets. The asset parameters defined in table 3.1 are provided as input to the asset class, and will constitute the initial attributes (class variables) of the asset object. The class consists of many attributes and methods (class functions). The first important method, *add_energystorage()* involves the possibility to attach an energy storage to the asset model, if it has any. The inputs to this method is the energy storage parameters of the asset. The second and last important method is *make_flexplot()* which makes the final flexibility estimates and creates a flexplot. The flexplot is presented shortly.

The whole Python script for the class *Asset*, with commentaries and documentation, is included in Appendix B.2. The bottom of the script show some example usage of the class, used for the making of the flexplots throughout this thesis.

Further work yet to be done is to implement a class for an aggregated group of assets, e.g. *Building*, in order to provide aggregated flexplots and flexibility estimates.

3.2.4 Flexibility estimation (stage 3)

Flex equations
The flexibility lies in the ability to deviate from the baseline. Based on the baseline

forecasts $\vec{\hat{b}}$, the forecasted maximum **positive** flexible power $\vec{F_p}$ is defined to be

$$\vec{\hat{F}_p} = \vec{P_{max}} - \vec{\hat{b}} = \left[P_{max}^{(h)} - \hat{b^{(h)}}\right]_{h=1}^{H} \tag{3.13}$$

Similarly, the forecasted maximum **negative** flexible power is denoted

$$\vec{\hat{F}_n} = \vec{P_{min}} - \vec{\hat{b}} = \left[P_{min}^{(h)} - \hat{b^{(h)}}\right]_{h=1}^{H} \tag{3.14}$$

These values represent the estimated flexibility at each forecast timestep, h, in the bidding horizon H. Units are in kWh/h.

So far, the flexibility estimates do not take into consideration any limits set by the energy storage or potential time restrictions. In the implementation program in Python, the ultimate estimated available flexible power is restrained by energy storage capacity limits, as soon illustrated.

Flexplot - visualizing available flexibility and energy storage trajectories Visualization provide better understanding for humans and is the first stage of decision support, on the road to a fully intelligent and automatized system. Plots have been developed to visualize the estimated available flexibility and all possible impacts on the energy storage for all possible estimated outcomes. Simulations for the energy storage trajectories will reveal when activation of flexible power will result in hitting any energy storage limits. That is taken into consideration and the estimated available flexible power is shredded if simulations indicate that energy storage limits are reached. The plot is henceforth referred to as a *flexplot*, and an example flexplot is shown in figure 3.5.

Passive charge schedule

For the explanation of this flexplot, a Pixii battery [34] with the following specifications and assumptions is used:

- Temporal resolution is 15 minutes, $\Delta t = 0.25$
- Max og min power, $P_{max} = -P_{min} = 30\text{kW}$
- Energy storage has a capacity of $E_{min} = 0\text{kWh}$ and $E_{max} = 65\text{kWh}$
- Steady state consumption equals zero, since there are no power losses in battery, $\vec{P}_{SS} = \vec{0}$
- Forecasted baseline = inactive load schedule, $\vec{\hat{b}} = \vec{0}$

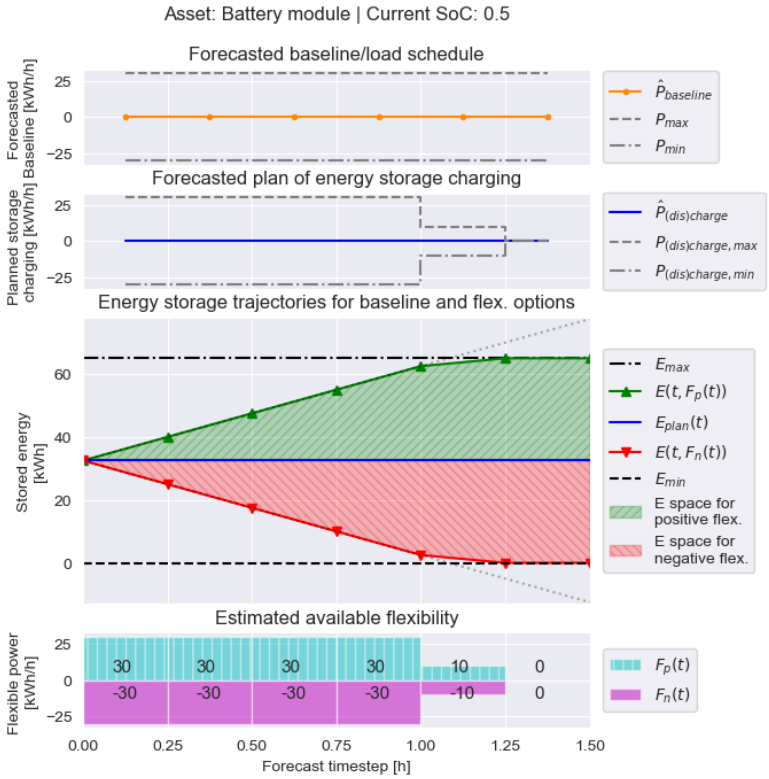


Figure 3.5: An example of a flexplot, here using a Pixii battery as reference, with a passive load schedule.

- There are no conversion losses, e=1
- The battery has an initial SoC of 0.5 (32.5 kWh)

Explanation of the flexplot

The flexplot consists of four subplots which all share the same time axis shown at the bottom. The values on the time axis indicate time from the start of the bidding period for this estimate, with hours as unit. The units on the y axis are kWh/h for power and kWh for energy storage. The upper subplot is of the forecasted baseline power along with maximum and minimum power. The second subplot shows calculated planned charging power of the energy storage, based on the forecasted baseline, using eq. 3.10. If it is zero, it means the SoC level is planned to be unchanged. A positive or negative charge power will lead to an increase or decrease in the energy storage level respectively. The plot also indicates the maximum and minimum limits for charging of the energy storage, based on

the max and min power. The first and second subplots do in this case look similar, because of the chosen example (no storage losses), but this does not have to be the case (as examples will show).

The third subplot is of importance to visualize what will happen with the energy storage level. The y axis represents stored energy. The limits of the energy storage are indicated. The graph reveals what is going to happen with the energy storage level for different choices of asset consumption. The blue solid line shows the planned energy trajectory, $E_{plan}(t)$, which is a simulation of energy level when following the forecasted baseline consumption. It is the path when choosing not to activate any estimated available flexibility. For this specific example, the planned energy storage trajectory remains constant, because of the planned charging power being zero. However, the planned baseline could very well involve charging, as shown later. Explaining the remaining content of this subfigure is postponed to after explaining the fourth subplot.

The fourth subplot ultimately reveals the estimated flexibility, represented by the estimated max positive flexible power in striped cyan and estimated max negative flexible power in solid purple. The estimates correspond to the difference between the forecasted baseline and the power limits, as stated by the equations 3.13 and 3.14. The estimated flexibility here are symmetric because of the chosen symmetric case. It is 30 kWh/h both in the negative and positive direction. After timestep 1.00 it sinks to 10 kWh/h and then to zero even, despite that the asset still has a lot of choices to deviate from its baseline consumption, according to the first subplot. The reason is that the energy storage limits are reached and impose a shredding of the estimated flexibility.

Choosing to activate either $F_p(t)$ or $F_n(t)$ results in a change to the planned consumption, planned charging and the energy content. Referring to the energy storage subplot, the resulting energy trajectories of activating either all $F_p(t)$ or all $F_n(t)$ is plotted in green or red respectively. These trajectories are represented by $E(t, F_p(t))$ and $E(t, F_n(t))$ respectively. The fact that the energy storage limits are reached at timestep 1.00 is what impose the reduced flexibility seen in fourth subplot. The dashed light grey line in the third subplot shows how the trajectories would be without considering energy storage limits.

Code snippets for creating this flexplot is found at the bottom of the *Asset* class script in Appendix B.2, with *EXAMPLE 1* as reference.

Active charge schedule

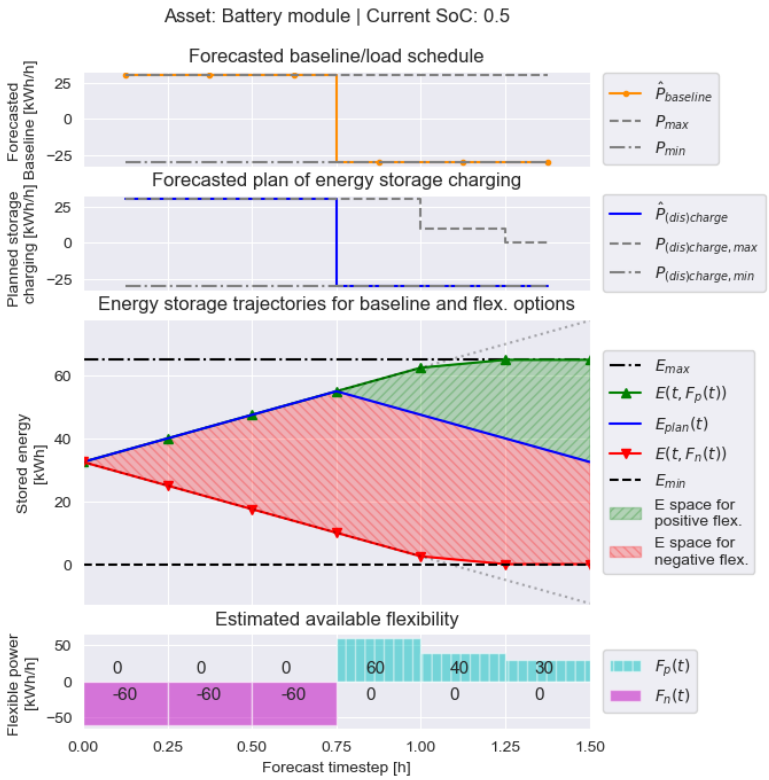


Figure 3.6: An example of a flexplot, here using a Pixii battery as reference, with an active load schedule.

Now, a flexplot for a battery with an active planned energy storage charging is showcased, and is shown in figure 3.6. The exact same battery as above is used as an example. The battery does not however have a passive plan, but instead a planned active load schedule, as followed

- Forecasted baseline = active load schedule,
  $\vec{\hat{b}} = \big[+30, +30, +30, -30, -30, -30\big]$ kWh/h

The load schedule could be for any reason, e.g. cost optimization or needs. For this case, the baseline power still equals charge power, as the first and second subplot indicate. As the blue solid line indicates, the planned energy trajectory

looks different now, due to planned charging. In addition, the flexibility estimates are different. The forecasted baseline that is entered to the flexibility platform is what lays the basis for calculating the flexible power. Therefore, the estimated flexible power is now -60 kWh/h in the first 3 timeslots. The plan was to charge the battery with +30 kWh/h, while there is still an opportunity to discharge -30 kWh/h. The resulting negative flexible power is the difference of - 60 kWh/h. There is no available positive flexibility for the first 3 slots, because the maximum possible power/charge is already planned. This can be seen in the energy plot as well, where the planned trajectory already follows the path of maximum positive flex, which is 0. The last 3 slots offers positive flexible power. The fourth slot is +60kWh/ and the fifth has been imposed restrictions on because energy limits are reached. Still, after the limits are reached, the sixth slot contain +30kWh/ of offered flexible power. That is because the battery can have a consumption of 0kWh/h, when -30kWh/h is planned.

Code snippets for creating this flexplot, figure 3.6, is found in Appendix B.2, with *EXAMPLE 2* as reference.

3.2.5 Bid formatting (stage 4)

Based on the estimated flexible power, the ultimate bid is entered to the flexibility market platform. It consist of H timeslots, each slot containing both a bid for positive and negative flexible power, as seen in figure 3.7. The bid should comply with the flexibility product presented in the section under*NODES Flexibility product*. Pricing of each flexibility bid slot is important, but is left out of the scope of this thesis.

The bid in each slot could be linear, absolute or even contain a minimum activation, e.g. an absolute value above 30kWh/h. For the estimations in this thesis it is assumed an activation of the maximum bidded available flexible power, referred to as a full activation. Later, some alternatives will be briefly discussed, e.g. activating half the power of a bid.

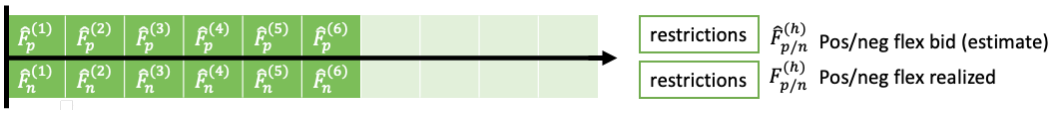


Figure 3.7: Example of a how a flexibility bid that is entered into the flexibility platform, may be illustrated. It is constituted of several slots in the bidding horizon $h \in [1, H]$, here with $H = 6$.

3.2.6 Aftermath and error measures

There are two different **outcomes** after the bid is formatted and entered into the flexibility market platform. When a bid is *ignored*, we *expect* the realized consumption to follow the forecasted baseline,

$$\hat{P}_{cons}^{(h)} = \hat{b}^{(h)} \tag{3.15}$$

The *realized* consumption, however, does in fact become

$$P_{cons}^{(h)} = b^{(h)} \tag{3.16}$$

These values enables us to calculate the error of the baseline forecast, using either of the evaluation metrics mentioned in the section *Evaluating multi-step forecasts*.

When a bid is *activated*, the bought flexible power can be denoted $F_{committed}^{(h)}$. The realized consumption is anticipated to be equal to the forecasted baseline power with the added committed flexible power that is expected to be dispatched, expressed

$$\hat{P_{cons}}^{(h)} = \hat{b}^{(h)} + F_{committed}^{(h)} \tag{3.17}$$

During the influence of the activated flexible power, the realized baseline, $\vec{b}$, will not become available, but the realized consumption $P_{cons}^{(h)}$ will be. That enables us to calculate the **delivered** flexibility, $F_{delivered}^{(h)}$, which is expressed

$$F_{delivered}^{(h)} = P_{cons}^{(h)} - \hat{b}^{(h)} \tag{3.18}$$

The failed deliverance, or error of flexibility deliverance, can then be expressed

$$R_{delivered}^{(h)} = F_{delivered}^{(h)} - F_{committed}^{(h)} \tag{3.19}$$

where the delivered flexibility is found with eq. 3.18 and the estimated flexibility is the flexible power that is bought and committed.

There is a distinction between two situations - backtesting and real-time use of the model. During backtesting, the true realized baseline is available, and assumptions on flexibility activation is done. This is opposed to real-time usage where the forecasted load never will see its true realization play out. Everything of this thesis must be based on backtesting, because of the lack of opportunity to conduct practical tests and real application of the methodology. Therefore, the results will be hypothetical. The generic explanation of the methodology is finished. Now, its usage during time advancement will be demonstrated.

3.2.7 Bid event line - time advancement

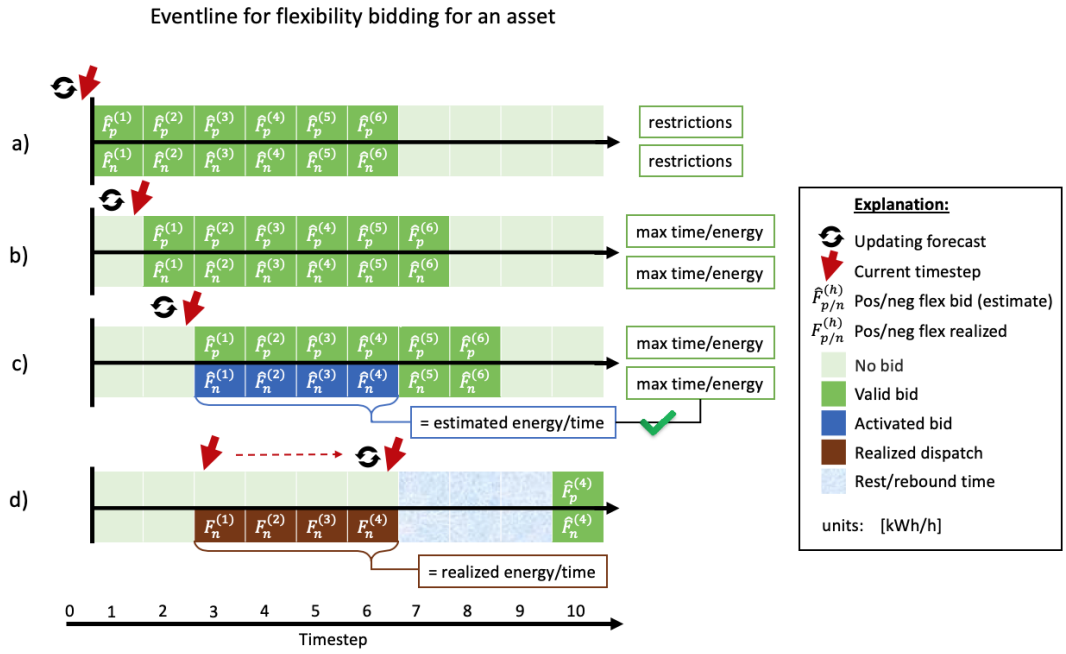


Figure 3.8: Example of a how a line of bidding events may look like during time advancement. Each subfigure represents successive events of bidding, where each event involves assessing flexibility estimates by means of the methodology. The flexibility bids in (a) and (b) are ignored. In (c), a part of the bid is thought activated, followed by (c) the dispatch process.

This section is dedicated to explaining the methodology in action during time advancement and in a line of bid events. When a flexibility bid is entered to the flexibility market, it is either ignored or activated. As time advance, the methodology will repeat itself in order to make flexibility estimates for new bidding

horizons.

To demonstrate how the methodology works during time advancement, a conceptual demonstration of the bidding procedure has been designed. It is designed to show what happens in both outcomes of the methodology. For this purpose, 4 successive bid events are created. Figure 3.8 shows 4 flexibility bids in the bidding procedure. Each bid represents an event, where each event involves the use of the methodology for estimating flexibility to create the illustrated bids. Each event is as follows:

- Event (a): According to the methodology, a flexibility bid is made and entered into the platform, but the bid is ignored. We proceed to the next timestep.
- Event (b): Flexibility estimates are yet again formulated into a flexibility bid and ignored again. We proceed to the next timestep to prepare for the next bidding period.
- Event (c): The new flexibility bid that is entered is bought. As an example, the four first 4 slots for negative flexibility are activated, however this is only illustrative. The activated flexibility must match the given time and energy constraints. We follow path *bid activated* in figure 3.1.
- Event (d): According to the methodology, the dispatch will try to deliver the committed flexible power in its best fashion, however it may fail. The values in the dark brown coloured slots, representing delivered flexible power, may therefore differ from the values of the activated flexibility. As the dispatch period soon comes to an end, there are some alternatives. Either, new flexibility estimates are done right away or there is a rest or rebound period.
- The same process goes on and the methodology is repeated.

This set-up is used as a standard for later practical examples.

3.3 Implementation of the methodology for selected assets

This is the third and last section of the chapter and will present specific descriptions on how the methodology can be implemented for the five following assets: Battery, diesel generator, PV panels, water heater (with heat tank or with alternative energy source) and machine room.

3.3.1 Thermal energy storages and heat losses

Before presenting the conceptual methodology for assets with thermal energy storages, some theory on thermal physics must be presented. Equations for the energy level and how heat transfer mechanisms change the energy level of a heat storage will therefore be presented. This section is aimed to support the modelling framework for flexible assets whose behaviour relies on underlying thermal physical processes, such as a cooling storage and a water-based heat storage. Knowledge about heat losses will also provide information on which heat loss parameters to include as explanatory variables in load forecast models.

Most of following theory on thermodynamics is from the book by Sonntag & Borgnakke [35].

Heat and transfer mechanisms
Heat is thermal energy and everything that have atoms with a temperature above 0 Kelvin possess heat. In practice, one focus on relative change of heat in an object. The change of thermal energy ΔQ (Wh) of an object with mass m (kg) subject to a temperature change ΔT (K), can be expressed

$$\Delta E = \Delta Q = c_p m \Delta T \tag{3.20}$$

where c_p ($\frac{\text{Wh}}{\text{kgK}}$) is the material-specific heat capacity of the object. The new thermal energy in an object subject for a change in thermal energy would simply be $E_0 + \Delta E$, where E_0 usually is a constant that is set freely. Wh is a practical unit for energy because electricity also uses this unit.

In order to change the thermal energy of an object, it needs to exchange heat power with the surroundings. Heat exchange happens through three heat transfer mechanisms: conduction, convection and radiation. These mechanisms are presented below, except for radiation, which can be neglected in the further work.

Heat conduction $\dot{Q}_{cond}$ (W) is heat that transfer through a material, i.e. wall, and can be described by Fourier's law of heat conduction,

$$\dot{Q}_{cond} = -kA\frac{\Delta T}{\Delta x} \tag{3.21}$$

where k ($\frac{\mathrm{Wm}}{\mathrm{Km}^2}$) is heat conductivity of the material, ΔT (K) is the linear temperature gradient through the material, A (m^2) the cross section and Δx (m) the thickness of material.

Convection is heat transfer by motion of a fluid itself and is very complex to describe in a mathematical way. The driver for convective heat flow is temperature gradients in a fluid, which cause particle movement due to pressure differences. A great temperature gradient will cause high convective heat transfer until at last, when all air is mixed up and shares a common steady state temperature. As an aimed example, consider two enclosed rooms filled with air, having a difference in temperature ΔT. They are connected with a closed gate. When the gate opens, the temperature difference, ΔT, will be a driver for heat convection through the door, q_{conv}, which can be expressed

$$\dot{Q}_{conv} \propto \Delta T \tag{3.22}$$

A thermal energy storage raises or lowers its energy content by interacting with the environments. In order to provide flexible electrical power, it is needed that $P_{losses} > 0 \Rightarrow P_{cons} > 0$. The reason is that stored thermal energy itself cannot be transformed back into providing electricity to grid. The asset can only deviate its consumption of there is a natural drainage of the thermal storage. Unwanted storage heat losses stem from a warmer or colder environment to which heat is transferred according to the equations 3.22 and 3.21. Thermal energy storages are therefore often insulated to avoid unwanted heat losses, reducing the coefficient of heat conduction. There should be mechanisms to avoid unwanted convective heat losses as well. Ideally, power losses should fully be going to its purpose, e.g. heating offices.

3.3.2 Implementation for batteries

Batteries big advantage of being charged and discharged with ease and precision, makes them the most ideal resource for flexibility and grid balancing purposes, when ignoring costs. From now on, a high-capacity battery will be considered. Although still being a more expensive alternative to other flexibility sources, utility-scale batteries are a serious competitor for offering flexibility services in the energy

chain. By 2050, around 360 GW of batteries are expected to be a part of the world's power grids, and they expect to become a cost-competitive choice for load shifting from the mid 2020's [2]. A commercial battery available on today's market can offer 65 kWh of electrical energy storage with a 30 kW power conversion, packed in a half-tonne fridge-sized cabinet [34]. As batteries degrade over time with different use patterns, current research is being done to gain a better understanding of operational related costs. For this thesis, a simplistic battery model is to be used, as presented in methodology chapter.

Batteries do perhaps offer the simplest and easiest source of flexibility and has a smooth implementation of the conceptual model. A battery was chosen as an example for the flexplot explanation.

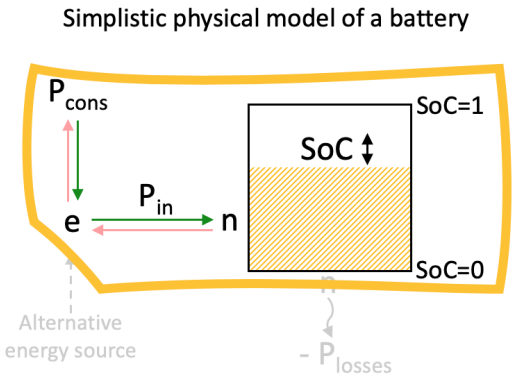


Figure 3.9: Simplistic physical model of a battery as a flexible asset.

A physical model of a battery is shown in figure 3.9. A battery is subject to decisions, and the load schedule must be actively set, for example in order to minimize operational costs. The forecasted baseline is therefore set by a load schedule. Steady state consumption is always equal to 0 for batteries, by assuming it has no storage losses. The charge power, P_{charge} will thus be equal to the P_{in}, which transform power from the grid with a certain efficiency, $e \in [0, 1]$. The charging and discharging of a battery can be expressed

$$P_{(dis)charge} = P_{in} = eP_{cons} \tag{3.23}$$

The action of storing energy in the battery for later extraction is associated with an efficiency loss, expressed by applying the above equation twice

$$P_{cons,extraction} = e^2 P_{cons,storing} \tag{3.24}$$

Inserting the given parameters into eq. 3.8 yields an expression for increasing the energy level, or SoC

$$\Delta SoC^{(t)} = n\Delta E^{(t)} = nP_{in}^{(t)}\Delta t = neP_{cons}^{(t)}\Delta t \tag{3.25}$$

When a battery is charging, it is consuming power, $P_{cons} > 0$ and ΔSoC would have to be positive. If they are negative, they indicate discharging and power production. The maximum and minimum power of the battery is usually provided by the manufacturer. Estimation of flexibility is next. It has already been demonstrated and was shown for the case of both a passive and an active load schedule, figure 3.6 and 3.5 respectively.

As with the passive battery, the figure reveals that fully activating the maximum amount of either positive or negative flexibility, results in hitting the boundaries of the energy storage limit within 5 hours. The flexibility the last hour is then shortened from this fact.

3.3.3 Implementing a diesel generator

A diesel generator generates electricity from running a combustion engine fuelled by diesel. It is ideal as a power backup, in case of power outages. Many diesel generators can also be fuelled by other liquid energy carriers, such as gasoline, bio diesel and colza oil, with different efficiencies.

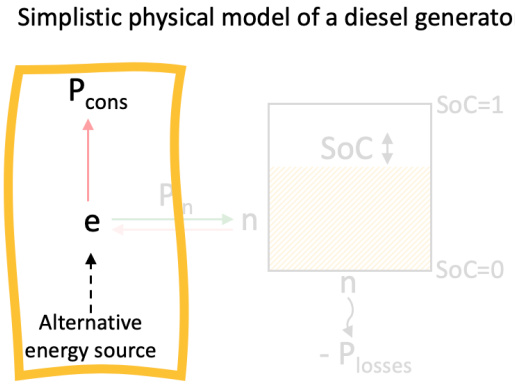


Figure 3.10: Simplistic physical model of a diesel generator or of a PV panel.

A physical model of a diesel generator is shown in figure 3.9. A diesel generator is always at backup to provide the alternative energy source as electricity to the

grid. It could have a minimum and maximum run time, and perhaps a rest time between runs. The forecasted baseline that is entered into the flexibility platform is simply equal to 0. The maximum power is determined by the specifications of the engine and how much electrical power it can deliver. The negative flexibility is directly quantified as the maximum power and this information could therefore be directly used to make bids. All the first stages of the methodology model could be skipped, and bids are entered directly to the platform. The diesel generator may have a minimum power, thus the negative flexible power is

$$-F_n \in \left[P_{max}, P_{min}\right]$$

3.3.4 Implementing PV solar panels

Photovoltaic solar panels (PV panels) have the unique ability to transform irradiance to electrical power. They do so when they get lit by both direct solar rays and diffuse light. A typical commercial PV panel has an efficiency around 15-19 % [36]. Solar power cannot be controlled, except from curtailing the production which could provide positive flexibility, for example in the case of an extreme grid emergency. However, smart and interconnected flexibility markets aim at avoiding such curtailment. Alternatives to curtailment should always be sought, such as charging other flexible assets in the building instead.

PV solar panels could provide flexible generation and their flexibility lies in curtailment of production. In the use-case, solar production assist to supply cooling power which correlates with temperature and mostly sun. PV curtailment may therefore lead to a high net building consumption that exceeds power tariff limits, thus resulting in a very high cost of the flexibility.

The PV panel asset can be modelled the same way as a diesel generator, as seen in figure 3.10. The forecasted production is entered to the flexibility platform as the baseline. The maximum negative flexibility is to curtail all solar production. For this reason, the solar power production must be forecasted. This is somewhat easy on sunny and complete shaded days, but become tough on days with clouds that come and go. That applies to the flexibility estimates as well. This asset does not have an energy storage or asset models, and the flexible power could directly be used to make flexibility bids. PV panel is not further investigated in this thesis. The implementation is clarified.

3.3.5 Implementing a water heater

A water heater consumes electricity to heat up an energy carrier, often water. It could have a tank for the energy carrier that stores thermal energy. Pipes transport the energy carrier to ultimately heat up whatever needs to be heated externally. If the thermal energy storage is of a significant size, energy-wise, one can expect it to provide flexibility. Another source of flexibility can also be if the electrical consumption has the opportunity to be replaced with another energy source. A heat tank consisting of water can store a lot of energy. The specific heat capacity of water is around 4.2 kJ/kgK, in the liquid phase range, at 1 atm pressure [35].

We look at two cases. The first case is to look at a water heater whose consumption can be totally replaced by an alternative energy source. The second case is to look at a water heater with a flexible thermal energy storage, which is providing the flexibility. In practice, the both can be combined. Using an alternative energy source will not result in any rebound effects, whereas discharging a thermal storage will lead to a rebound effect to recover after activation period.

The first stage, settling the baseline forecast, must be done for either case. Water heaters use electrical power to provide a heat power flow to an external purpose, which is considered a part of P_{losses}. From now on, it is assumed that the consumption equals steady state consumption

Predicting the baseline may require the creation of a load forecast model or, if the consumption is very stable, an average or repeated value can be used. One can expect the consumption to be influenced by the things that influence P_{losses}, such as conduction losses through the isolated walls and parameters behind its external purpose, e.g. office space heating. Investigation of such processes may therefore be helpful.

The ability to deviate from the consumption must be determined, which sets P_{min} and P_{max}. The successive stages will differ for the case of an alternative energy source versus a heat reservoir. The case with a thermal storage will be treated first, followed by the case with an alternative energy source.

Water heater with thermal storage

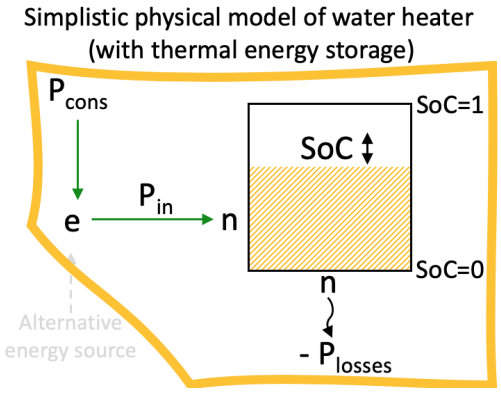


Figure 3.11: Simplistic physical model of a water heater with a thermal energy storage as a flexible asset.

A physical model of the water heater system with the heat reservoir can be seen in figure 3.11. P_{max} would be some maximum value, most probably a constant and will make up the possibilities for charging the energy storage. P_{min} can be assumed to be zero from now on, which means total consumption shut off. The drainage of the energy storage is thereby totally dependent on $P_{losses} > 0$.

The flexible range of energy levels in the heat storage is given by the equation

$$\Delta E_{range} = n\Delta SoC = cm\Delta T \tag{3.26}$$

where $\Delta T = T_{max} - T_{min}$ is the allowed range of temperatures(K), c is the heat capacity (kJ/kgK) and m is the mass (kg), of the medium. The latter can alternatively be expressed $m = \rho V$ with ρ being the density (kg/m^3) and V being the volume (m^3) of the medium. It is assumed no phase transitions. Temperature restrictions, such as a minimum and maximum allowed temperature, T_{min} and T_{max}, sets a lower and upper boundary for the thermal energy content, E_{min} and E_{max} respectively. The level of energy in the storage cannot exceed these limits.

An approach is to set $E_{min} = 0$ and let the maximum allowed energy level be expressed

$$E_{max} = E_{min} + \Delta E_{range} \tag{3.27}$$

Now, all is set to estimate the flexibility.

Some notes

If $P_{min} = 0$ and the asset consumption is shut off, the asset will provide negative flexible energy and also discharge. As mentioned, the drainage will totally depend on P_{losses}. If the bottom energy level is reached, then the electrical consumption must be turned back on. From this fact, it is important to have good forecasts of the electrical consumption, assumed steady state. The forecasted baseline will only be able to provide a forecast of the potential new energy level after activation and the time of a completely empty energy storage. A problem is that, if the baseline forecast is wrong, especially when it has underestimated the forecasted steady state consumption, it will thereby underestimate the losses. This would result in an overestimated flexibility. It means that activation of that flexibility is expected to drain the storage in a certain manner, but the energy storage will in practice drain faster than anticipated. As a result, the flexibility activation will not be able to last as long as committed, as the consumption must be turned back on when the bottom energy limit is reached before time.

Some ways to avoid this to happen is to provide more accurate, or even overestimated, forecasts or subtract a share of the estimated flexibility as a safety margin. That may help prevent failed deliverance of flexibility.

The rebound effect is a consideration that should be investigated. The rebound must be analysed through experiments, and the rebound strength could perhaps be controlled. Notes on implementation of rebound effects are for the discussion later.

Water heater with alternative energy source

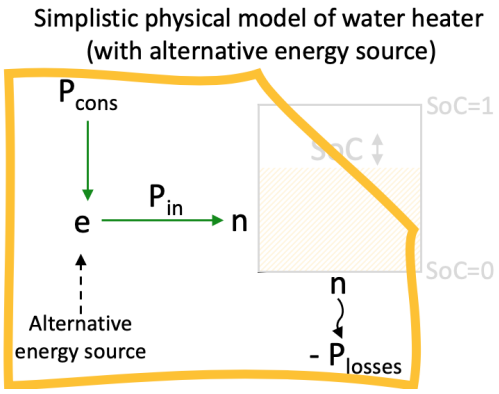


Figure 3.12: Simplistic physical model of a water heater with the opportunity to replace electrical consumption with an alternative energy source, as a flexible asset.

A simple sketch of the water heater system with an alternative energy source is shown in figure 3.12. A twist in this case is that P_{max} is equal to the baseline consumption of the asset. That yields no positive flexible power. P_{min} is simply zero. Thus, the negative flexibility lies in shutting off the electrical power. The alternative energy source still provides everything that is needed of power for the water heater to continue delivering its functions, in P_{losses}. If there are any time constraints for the alternative energy source, regarding max runtime or rest time after activation, this can easily be implemented. All is set to estimate the negative flexible power and make flexibility bids.

Some notes:
The alternative energy source may have time constraints, which are taken care of in the flexibility estimation or bid formulation. Rebound effects are not present, however there might be a rest time, in which a resting period must pass before the alternative energy source is yet again ready.

Aftermath:
The realized baseline will not be available and the asset has not any energy storage than can be drained. Therefore, the delivered flexibility is concluded to be perfectly fulfilled.

3.3.6 Implementing a machine room for cooling storage

A complete conceptual description for implementing the preliminary methodology on a machine room asset is now presented, including the steps for the making of a forecast model. Figure 3.13 shows a simplistic model of the machine room asset with its power flows and the SoC level of the cooling storage.

The first stage in the methodology is to create a load forecast model that can make predictions for the future baseline consumption. Step one is in many ways the same as forecasting the heat losses, since the baseline is assumed to be the steady state power, $P_{cons} = P_{cons,SS} \Rightarrow P_{(dis)charge,SS} = 0$. The ability to be flexible is to either blow up or ease down on electricity consumption from the forecasted baseline, thus charging or discharging the thermal storage respectively. According to the methodology, the second step would be to quantify the potential to deviate from the baseline. This must be done by analysing the ability to control the electrical consumption and finding P_{max} and P_{min}. In addition, the storage parameters E_{min} and E_{max} must be determined.

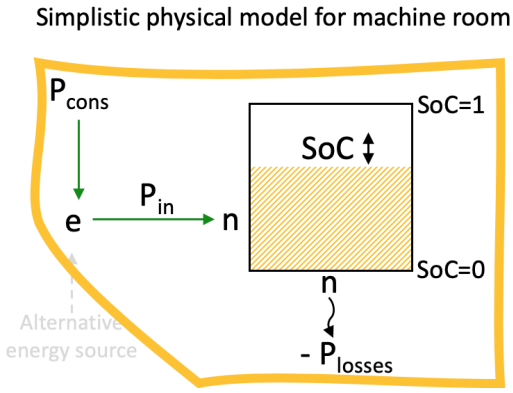


Figure 3.13: Simplistic physical model of the machine room and cooling storage as a flexible asset.

Settling the storage losses

Since understanding heat losses will increase predictive power for electrical consumption, heat losses for the cooling storage will now be analysed in the coming paragraphs.

Figure 3.13 is now referred to. The cooling power, represented by P_{in} (W), must equal the net losses, P_{losses} (W) to maintain the set-temperature and SoC level of the storage. There is convection losses through a potential open gate and conduction losses through the walls, roof and closed gates. The power loss is solely constituted by heat losses that can explained by the two heat transfer mechanisms, equations 3.22 and 3.21. Combined, this yields

$$-P_{in} = P_{losses} = \dot{Q}_{cond} + \dot{Q}_{conv} \tag{3.28}$$

One unit of electricity consumption P_{cons} (W) that is needed to create one unit of storage cooling P_{in} (W) is in fact lower, due to a usually high Coefficient of Performance (COP) of above 4, $e > 4$. Combined with the latter equation, this can be expressed

$$P_{cons} = \frac{P_{in}}{e} \tag{3.29}$$

To conclude, the electrical consumption becomes dependent on heat losses

$$P_{cons} = \frac{1}{e}(\dot{Q}_{cond} + \dot{Q}_{conv}) \tag{3.30}$$

The variables of the respective heat transfer mechanisms indicate that outside temperature will play a key role in both heat loss mechanisms. In addition, the frequency of open gates will be essential for the presence of convective heat losses.

Thus, these variables should be included as explanatory features in a forecast model in order to improve the performance and predictions of the consumption. e must not be determined, because it will be found by the RNN model. If the frequency of the gates can be modelled, this would be a valuable feature. Once a good forecast model is realized, good forecasts gives a good foundation for the estimating flexibility. Note that the load forecast also is a forecast of the steady state consumption.

stage 2 Next step in the methodology is to determine the asset parameters. A cooling storage for groceries needs to maintain a desired air temperature, called the set-temperature, T_{set}, which can be manipulated within an allowed deviation. Temperature restrictions are set for food safety regulations, under strict laws. The range of allowed temperatures will yield a range of a flexible energy storage levels. This makes the asset possess flexible power. One needs to decide the range of allowed temperatures. The lower and upper temperature limits, T_{min} and T_{max} will define the maximum and minimum allowed energy storage capacity, E_{max} and E_{min} respectively, according to equation 3.20. The other variables in the equation, heat capacity and mass, could be assumed constant or be variables. If they are variable, it is because groceries are moved in and out of the cooling storage. The resulting range of allowed energy levels will be a function of the grocery volume and perhaps other variables. One could create physical models incorporating grocery volume, but this may be a complex task. For the rest of this thesis, the combined heat capacity is assumed constant. Finding this constant must be done by experts, however, step-tests have been proposed to investigate how the storage energy level reacts to charging and discharging.

Even though the COP is not necessary to know in RNNs as they learn the weights between the relations automatically, it may nevertheless be beneficial to have real-time information on a changing COP value as a feature, if it is known.

Analysing the ability to deviate and storage properties may be a complex task to do. A suggested method is to carry out step-response tests, where temperature set points are manipulated in order to monitor the response of the electrical consumption. Then, a known change in electrical consumption will provide a measured change in temperature. The heat capacity can be calculated and the assets maximum ability to deviate from its baseline can be mapped and provide P_{min} and P_{max}. Step-down and step-up responses may look different in both time and amplitude. Charging of a cooling storage may probably take longer as opposed to discharging, since compressors often are dimensioned to maintain a temperature and not to boost it.

A simplification
Without practial analysis and knowledge on the asset parameters, some reasonable simplifications must be made. From now on, a constant heat capacity and a constant mass is assumed. Combined, they are referred to as a constant *combined* heat capacity. Different assumptions for P_{max} and P_{min} will be looked at. First, it could be assumed that P_{max} and P_{min} are purely defined out of the baseline and equals the baseline plus or minus a constant respectively. Another assumption that is looked into is that P_{max} and P_{min} are defined from a rolling mean of the baseline. Then, a rolling mean over 3 timesteps is made from the baseline. Adding or subtracting a constant to this yield P_{max} and P_{min} respectively.

Once the asset parameters have been determined, **stage 3** will calcualte and estimate the available flexibility and create a flexplot. Based on this, a flexibility product has to be shaped at **stage 4**. Eventual error margins are considered in the bid before the forecasted baseline and the flexibility bid is entered into the NODES platform.

Chapter 4

Use-case: Flexibility at a grocery warehouse

4.1 Introduction

This entire chapter is about applying the aforementioned theory and methodology on a real-case scenario building. The goal is to investigate the feasibility of the preliminary methodology for assessing short-term flexibility in a flexible asset and to demonstrate the use of the methodology with some real numbers and examples.

The structure of this chapter is as follows. Information about the use-case will be given at first. The main part of the use-case is application of the methodology on a machine room asset for a cooling storage with real consumption data. The dataset is explored for correlations and then preprocessed before forecast models are built. The results from one-step and multi-step RNN forecast models are a major part of the use-case. Then, these baseline forecasts will lay the foundation for the further stages in the implementation of the methodology. Some assumptions has to be done. Each stage will be explained, before a demonstration of a bid event line finishes the whole use-case for the machine room asset. At end, some other short relevant examples for some of the other assets are shown, for the case of diversity. Figure 4.1 shows that assets in this use-case building, along with the internal power flows.

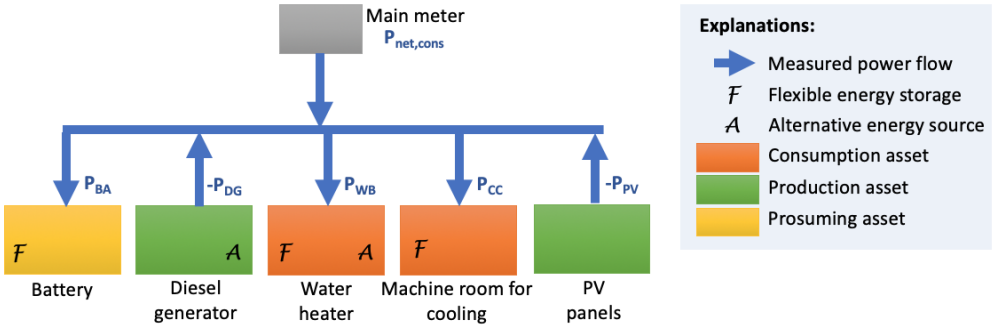


Figure 4.1: The system of ASKOs building and its considered assets. Power flows and explanations are included in the figure. The main meter connects to the grid.

About the use-case

The considered use-case involves a grocery warehouse in Vestby belonging to ASKO ØST AS. Their main task is storage and distribution of groceries in the food transport chain. In other words, groceries - room-tempered, chilled or frozen - arrive here, get temporarily stored until they are sent away. In the era of smart grids and the fact that ASKO ØST is a major power consumer in the local grid, has made them pay attention to demand response management and utilizing their flexibility. This is both of their own economic interest and of interest for the local DSO in order to operate the local grid efficiently. The local DSO is Hafslund AS and operates the distribution grid. They have interest in the digitization age and are exploring new innovative ways to operate the grid, including NODES and flexibility markets. A third party, the smart grid company eSmart, plays the role as the aggregator for ASKO. Their task is to analyse and optimize the use of flexible resources within ASKO and potentially to a future local flexibility market, thought to be NODES.

ASKO already has incentives to utilize their flexibility to make profit by exploiting price variations and the power tariffs. A proposal is to investigate a flexibility market platform with NODES as the operator, where the DSO, Hafslund, is a thought flexibility buyer. ASKO is thought of as a flexibility provider with eSmart in the aggregator role.

All in all, the overarched goal for this use-case is to provide a methodology for eSmart and ASKO to assess short-term flexibility in their flexible assets. The use-case focus mainly on making flexibility bids according to the NODES flexibility market. However, parts of the applied methodology in this use-case may be ad-

vantageous for other scenarios as well. For example, precise RNN load forecasts and asset models are also needed in the process for exploiting price-variations with cost-optimization methods. Good short-term flexibility estimates are needed in the case of direct DRM as well.

Shortly about the electricity contract
The first layer of the electricity price is the area price set by NordPool the day ahead. Power tariff modules comes in addition, and they intend to punish ASKO for extra consumption during the periods where Hafslund wants lower consumption. One power tariff module adds an additional power cost at certain times during the day. Another power tariff does give incentives for ASKO to not exceed a certain monthly peak, where exceeding it involves a penalty power price that applies to all consumption of that month. Shaving of peaks and load shifting is therefore already in ASKOs interest because of the contract.

The NODES flexibility market platform will only add to the incentives for ASKO to use their flexibility. In addition, it will increase the intelligence and preciseness of Hafslunds grid operation. The above-mentioned contractual incentives are still in place, but with a flexibility market in place, ASKO will have more opportunities to revenue from their DFS.

Assumptions
There are some restrictions for fully applying a practical implementation of the methodology to the use-case. In lack of data and practical experiments, many simplifications must be made for the asset parameters and for the dispatch process. For the machine room asset especially, that means that important information about its behaviour, storage parameters, etc. is not available, which call for assumptions. A second effect from the lack of experiments is that the dispatch process itself cannot be measured. The results for the success of delivered flexibility is therefore hypothetical and based on the assumptions. Nevertheless, the application is conducted and will show that the methodology to work.

Building assets and parameters
Figure 4.1 gives a nice overview of the building with its assets, showing their place in the hierarchical structure and the power flows. The main meter measures net building power, and represents the bought and sold power with the grid contractor respectively. The building contains the same assets as the selected assets which were presented in the methodology chapter. The following list will further describe each asset for this specific use-case.

- **Machine room:** a complex cooling system, consisting of compressors, pumps, valves etc. to provide cooling for the frozen and chilled section of the warehouses, where the rest-heat is reused for comfort heating in offices etc. These components are placed in a machine room, whose net consumption is referred to as machine room consumption. It is a power demanding process which could provide 2-way flexibility, if carefully studied. Flexibility potentials, max and minimum power, energy storage capacity and other parameters are currently unknown.
- Electrical **water heater**, whose warm water is used to either defrosting or heating of office space. The asset can replace its electricity consumption by an alternative energy source. It may also consist of a significantly sized flexible thermal energy storage, however its parameters are unknown during the work of this thesis. Both alternative energy source and a flexible thermal storage will both provide negative flexibility, whereas positive flexible power could only be provided with a heat storage.
- **PV panels** on the rooftop, which produces electricity during a sunny day. It produces electricity when the need for cooling is most likely present. The positive flexibility lies in curtailment of production and does therefore depend on solar irradiance forecasts.
- A **diesel generator**, serves as a back-up power source, and need to be test run once each month. It can provide negative flexibility that is limited by a maximum power of 2 MW and an unknown minimum power, for a maximum of unknown hours. Rest time is unknown.
- A high-capacity **battery** bank is still at the planning stage and may be built in order to provide flexibility. For the purpose of this use-case, a Pixii battery is relevant, which has 65kWh of energy capacity and maximum of 30 kW of charging/discharging power. Multiple batteries can be stacked in order to upscale these parameters.

Provided data:
eSmart and ASKO has provided a dataset that contains timeseries of historic consumption for all of the above assets and for the main meter, except the battery. The timeseries has a temporal resolution of at least 15 minutes. The consumption for machine room is used. In addition, inspiration of water heater consumption is taken from the dataseries. Temperature data is gathered from Yr.

4.2 Methodology applied to the machine room asset

About the cooling storage
Understanding the building physics would be helpful for understanding the machine room consumption. A simplified illustration of the cooling storage is in figure 4.2, where the outer gate represents many gates. The gates connects grocery trucks to the cooling hall, with a sealing around to prevent heat losses. The sealing is however not completely sealed and cause heat losses, however there is a thin express gate which closes within 19 seconds of inactivity, which reduce this heat loss. When there is no truck connected, a main well-isolated gate is closed. With reasoning in the equations for heat losses, the machine room consumption depends on outside temperatures and the gate activity.

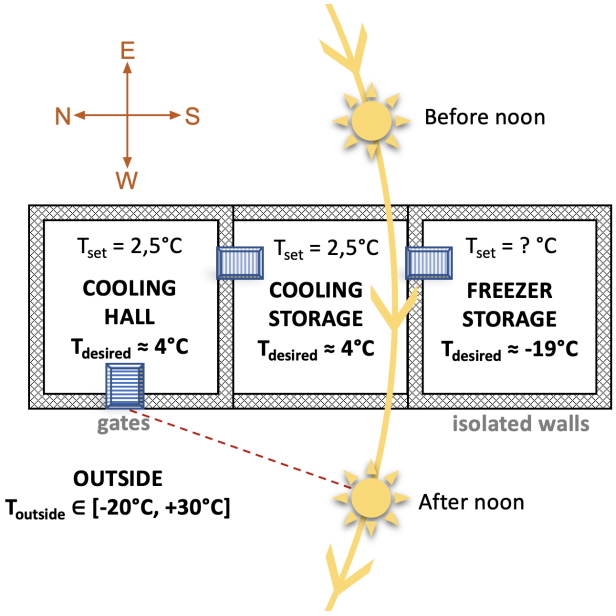


Figure 4.2: Simplified illustration of the cooling storage setup, which is relevant for the machine room asset. The daily sunpath and cardinal directions are indicated as well.

4.2.1 Data investigation, analyses and preprocessing

Historical data reveals that the machine room consumption is a highly volatile time-series, as seen in figure 4.3. The volatile line represents the highest available

resolution; 5 min. Between 12 and 15 o'clock, the minimum consumption is approximate 60 % of the maximum, and this volatility can be observed throughout the day. The smooth line is a downsampled version of the consumption, and shows hourly aggregated consumption, in which a peak can be observed in the day during work hours when there is gate activity.

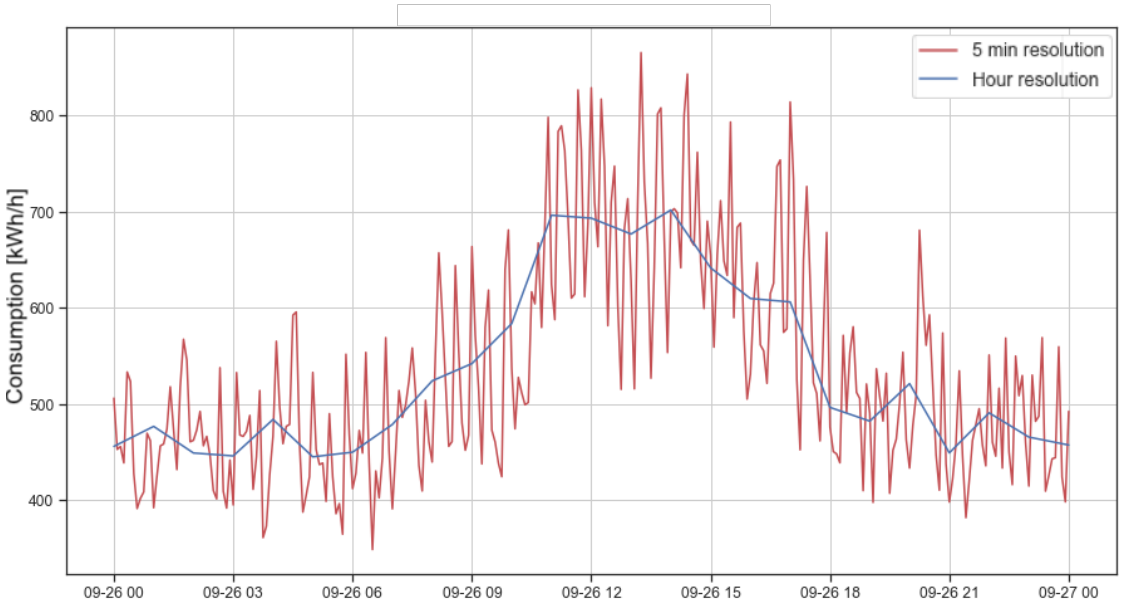
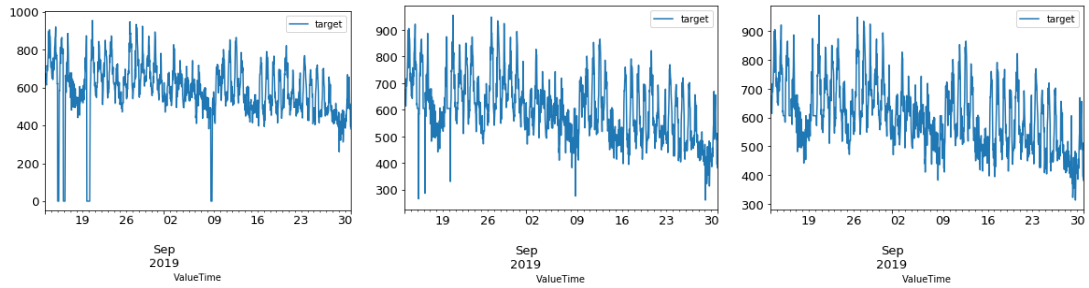


Figure 4.3: Plot showing historical consumption for the machine room asset for Sept 26th, 2019. The volatile red line is the original timeseries with 5-min temporal resolution and the blue averaged line is a downsampled timeseries with 1-hour resolution.

Missing data

The period of Aug 13th to Sept 30th 2019 is chosen as the dataset in this use case. Missing data was found and taken care of with the following steps, illustrated by figure 4.4, showing consumption plots for the whole dataset length. Step one is to find the periods with missing data. They are observed as values equal to zero in figure 4.4(a). Values in the missing periods are set equal to NaN, which we see the result of in figure 4.4(b). Still, some values seem oddly low, which can be due to error during downsampling near to the missing data. The oddly low values have been removed (set to NaN) and all the NaNs are imputed using linear interpolation between the known data points. The end result is seen in 4.4(c).



(a) First step in fixing missing data.

(b) Second step in fixing missing data.

(c) Third and last step in fixing missing data.

Figure 4.4: Plots of machine room consumption for different stages during the process of fixing missing data. Y-axis is consumption in kWh/h, x-axis indicates time in the range from Aug 13th to Sept 30th, 2019.

4.2.2 Correlations

The goal is to find good explanatory variables for a baseline forecast model. Therefore, some test to find correlation between consumption and outside temperature has been carried out.

Outside temperatures and machine room
The temperature outside the warehouse is expected to influence the machine room consumption. Outside temperature data from Ås (NMBU) is collected from *yr.no*, which is the nearest accessible weather station to ASKO. A quick check with Google Maps' measure tool reveals that approx. 8 km sets them apart. As a first correlation test, the machine room consumption is aggregated to each day, and plotted against daily temperature statistics, such as maximum, minimum, mean and normal temperature. The second test checks hour-by-hour correlation on a specific day.

Correlation plots for daily values of temperatures and machine room consumption are presented in figure 4.5. The y axis shows temperature in °C and the x axis power for machine room consumption. 2d, 3d and 4d represent corresponding rolling means of 2, 3 and 4 days of the consumption respectively. The results are supporting the hypothesis, as there seems to be strong positive correlations between machine room consumption and the temperature. Especially note the 'max' temperature, which would be the highest temperature of the day. This temperature most likely represent the middle of the day, when the warehouse has most activity. If all the points were on a straight line, then daily machine room

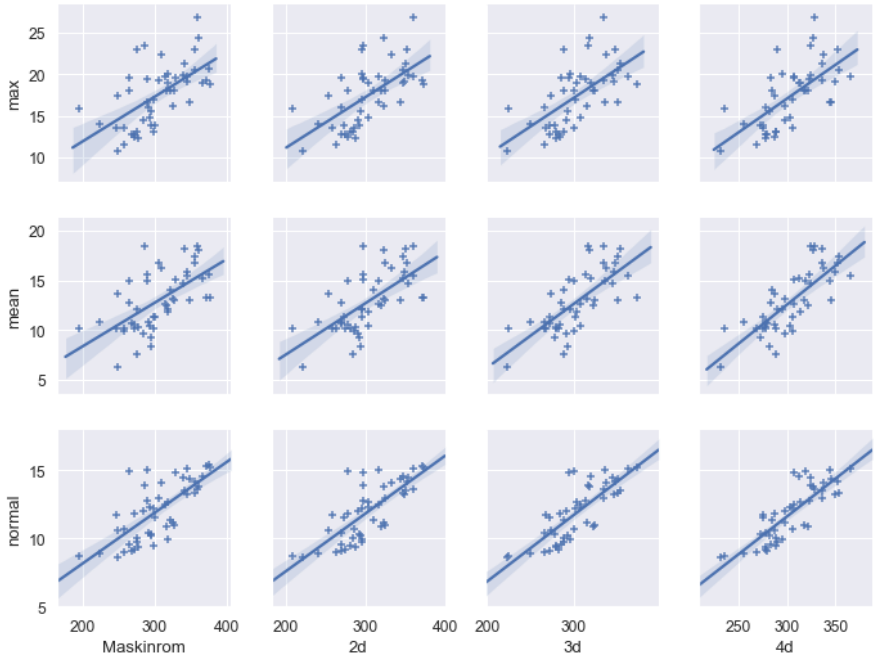


Figure 4.5: Correlation plots between outside temperatures near Vestby and machine room consumption and its rolling means over 2, 3 and 4 days (represented by '2d', '3d', and '4d' respectively). The used timeseries are daily values from Aug 13th to Sep 30th, 2019. Units on x-axis is kWh/day, y-axis is °C.

consumption could be explained purely from temperatures in Ås. However, it is obvious that there are some missing explanatory variables to fully explain machine room consumption or that temperatures in Ås do not represent temperatures outside ASKOs building. Nevertheless, temperature seems to be important, but it is necessary to also consider other parameters, e.g. gate activity or PV production.

Another interesting investigation would be to check the same correlation in higher time resolution. Figure 4.6 shows plot of machine room consumption and temperatures [1] in Ås on the days Sept 22 till Sept 29th with hourly resolution. In addition, a correlation plot between the temperature and consumption is included on the right of the figure. The temperature is upscaled with a factor of 25 in order to align. There seem to be a strong correlation during the week, and not so much during the weekend. Two important findings are found. First one is that the temperature seems to actually lag behind the consumption. The anticipation was

[1] accessed Oct 1st, 2019 from https://www.yr.no/sted/Norge/Akershus/Vestby/Vestby/almanakk.html?dato=2019-09-29

opposite because the causality is that high temperatures lead to high consumption, not opposite. Consumption actually rises each day to a peak independently before the temperature has risen. The second one is that the first and the two last orange spikes, which represent a weekend with less activity, are far off the consumption relatively to the rest.

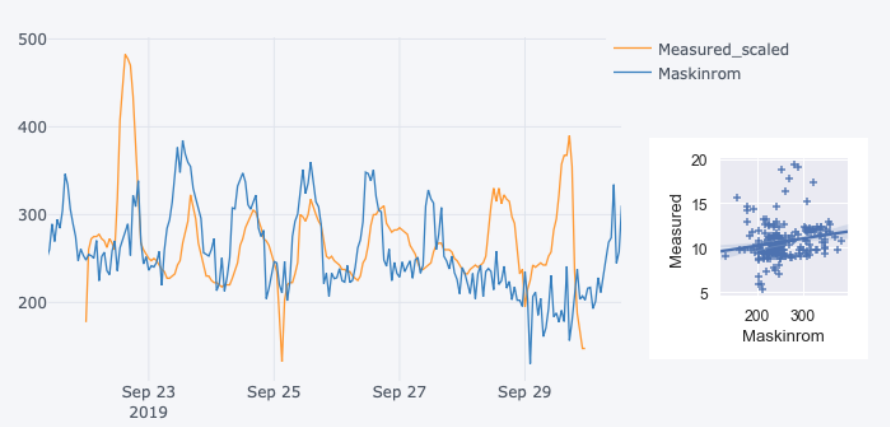


Figure 4.6: Plots of hourly values of machine room consumption and nearby outside temperatures (which is upscaled by factor 25), from Sept 22nd till Sept 29th. A corresponding correlation plot is found on the right.

The conclusion is that temperatures seem to explain some of the consumption, however, there have to be other important variables that are important for explaining the consumption. The expectation is that port activity plays a major role and that convection losses are an important parameter. Pure physically, convection losses are anticipated to be higher than conduction losses through well isolated walls. Nevertheless, the effect of convection losses when the gate is open, are intensified by temperature differences.

Machine room versus inside temperature of cooling storage

Another correlation test is based on the suspicion that the inside temperature of the cooling storage itself would be directly connected to the machine room power. However, a correlation analysis reveals that this is not the case, as seen in figure 4.7. In the figure, machine room consumption is plotted against inside temperature and also against lagged values of temperature values, none of which show any sign of correlation.

Temperatures 8km away may be inaccurate. A recommendation is to use historical and forecasted outside temperatures next to the building. Then, correlation can be investigated.

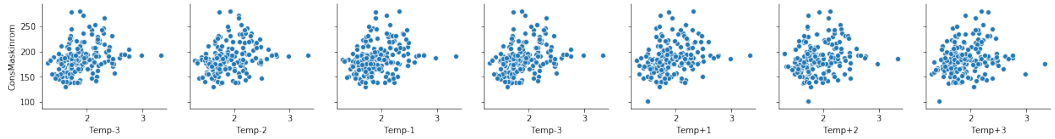


Figure 4.7: Correlation plots between the machine room consumption and inside storage temperatures and its lagged values. The (+) and (-) indicate the forward lead respectively backward lagged values of inside temperature.

Gate activity
The port activity is by hypothesis believed to play a major role to explain real-time machine room consumption. Altering of gate activity data is currently not done during the work of this thesis.

4.2.3 One-step RNN forecasts

The RNN models in this work only use consumption data in addition to created time-features as a features set. They do not include any explanatory variables. A complete list of one-step RNN forecast models with corresponding prediction scores are found in Appendices A.1 and A.2, tables A.1 and A.2. Model parameters and architecture are included as well. Forecast plots for each model is also included in the respective appendices, ref. figures A.1 to A.8. One of these models, the one with name *10292019_1715* is showcased here.

RNN model with GRU, one-step forecast

The RNN forecast model has been trained to forecast machine room consumption one hour ahead, using only lagged values of machine room consumption, time features and lagged values of time series as well. The architecture consists of two layers, first a GRU layer of 512 units, then a Dense layer with a sigmoid activation function to output a single value. It was trained with timeseries having temporal resolution of 1 hour, using batches with batch size of 256 and each sequence with length of 168 (24*7). Extra features were generated, using lagged values of machine room consumption with 1, 2, 3, 4, 5, 6, 24, 48 and 168 hour lag. Dummy binary variables of all categorical features were created, based on week of the year, day of the week and hour of the day. A train size of 90 % were used. A warmup period of 12 timesteps is used to improve overall model performance, which means that the 12 first timesteps of prediction will not count in MSE train validation, indicated

with grey area in the figures below.

Figure 4.8 and 4.8 show the forecast plots, for both train and test set data, respectively. The MSE is printed in the title. The loss value in the test set was **0.0071550351**.

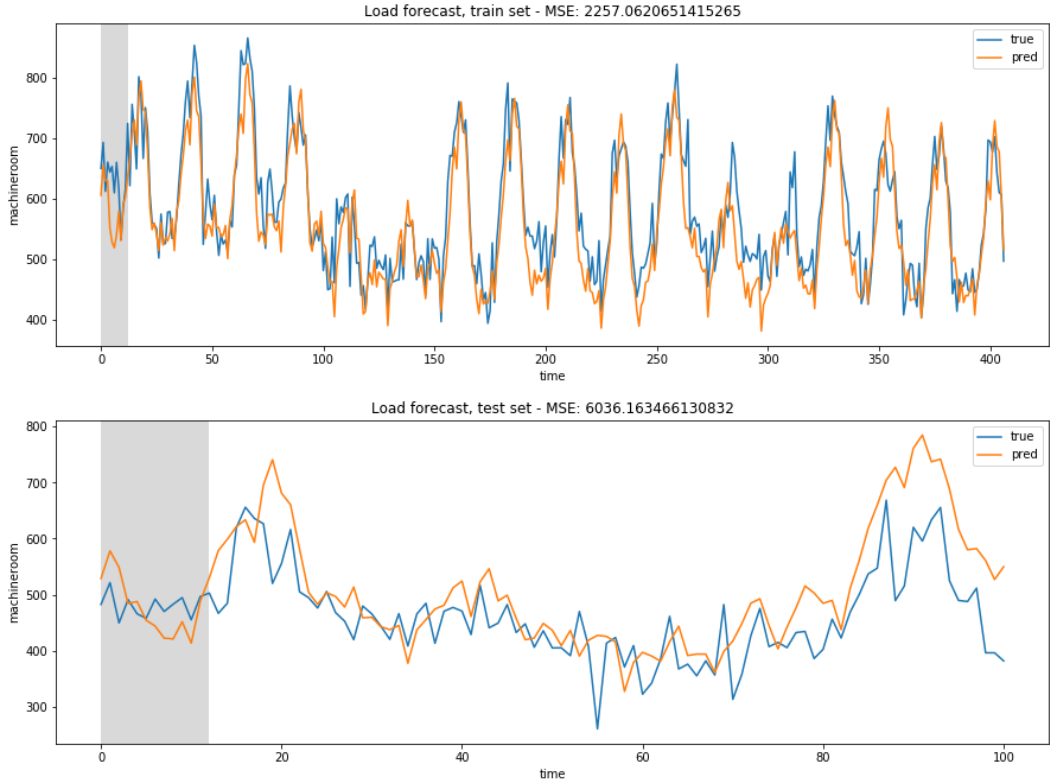


Figure 4.8: Forecast plot for machine room consumption, made with the one-step RNN model *10292019_1715*, on the train (upper) and test set (lower) respectively.

4.2.4 Multi-step RNN forecasts

Table A.3 in Appendix A.3 provides a full score table for 6 multi-step RNN forecast models. Model parameters and architecture are indicated as well. In addition to this table, forecast plots for each of the models in the table are presented in Appendix A.3 as well.

As the table and forecast plots reveal, the results are very different regarding their forecasting success. This is a good illustration that model settings and parameters

must be experimented with in order to get the desired results for the unique problem. *Model D* is now chosen to be showcased. It took a few hours to train. The forecast results from this model are also chosen to constitute the foundation for the rest of the use-case.

RNN model with LSTM, direct multi-step forecast
Model D and its forecast results is chosen as a foundation for the baseline forecasts used in the rest of this use-case. The model parameters are found in table A.3. Forecast plots are found in Appendix A.3 but are also re-shown here in figure 4.9, for train and test data respectively. In the plots, MSE scores are included for each of the forecast steps, h, whereas the average MSE is found in the table.

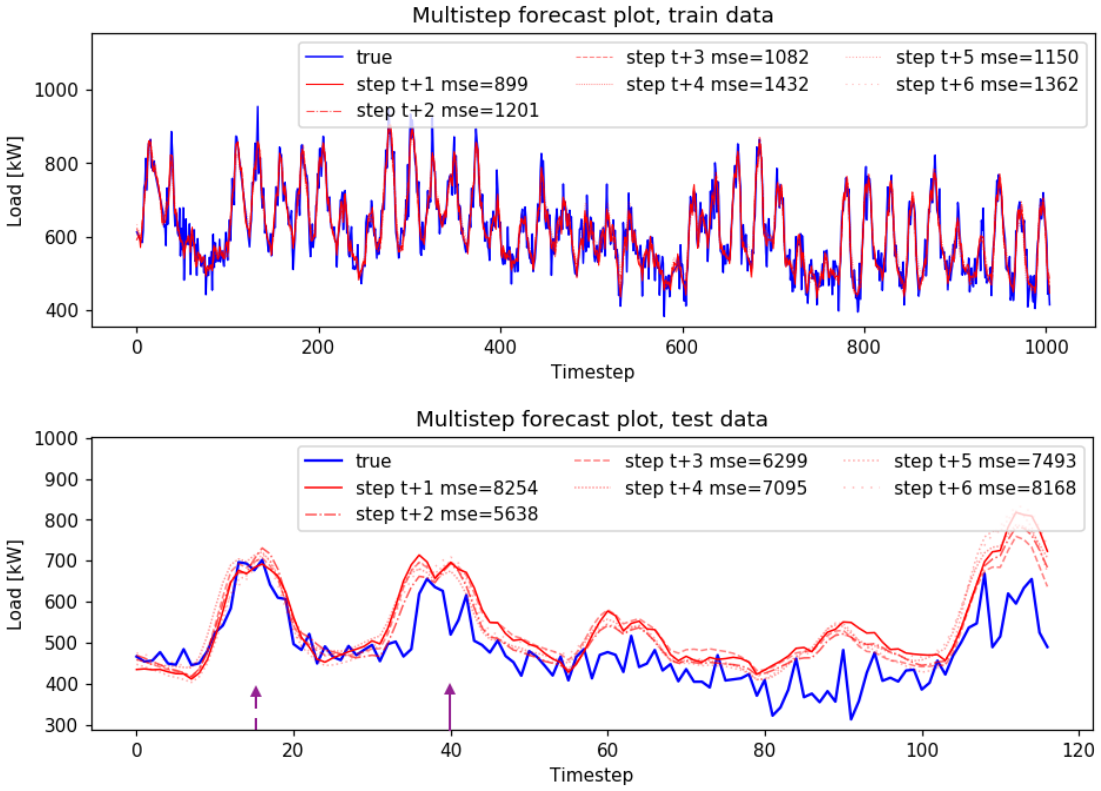


Figure 4.9: Forecast plot for train set and test set respectively, using the multi-step RNN model *Model D*.

Assumptions for the machine room asset
In lack of knowledge for the power and energy storage parameters for the machine room asset, some assumptions must be done in order to succeed with the demonstration. The assumptions are as following

- Temporal resolution is 1 hour, Δt=1
- The baseline equals steady state consumption, $b = P_{cons,SS}$
- The estimated maximum power P_{max} and minimum power P_{min} used for calculating flexibility is +50 kW and -80 kW relative to the forecasted baseline respectively. The estimated flexible power is therefore also +50 kW and -80 kW from the forecasted baseline.
- The actual realized consumption is assumed to be the true baseline plus the activated flexibility (and not the forecasted baseline plus the activated flexibility).
- Initial SoC = 0.6
- ΔE_{range} is arbitrarily picked to be 600 kWh, so that the time frame seems reasonable.
- E_{max} is arbitrarily set to be 8000 kWh.

4.2.5 Example demonstration of a bidding event line

A demonstration is created to follow exactly the design of the conceptual bidding event line demonstration presented in the methodology chapter, figure 3.8. To repeat, the goal is to demonstrate the a line of bidding events with both possible bid outcomes. Referring to figure 3.1, the creation of a bid is made by following stages 1-4 of the methodology and is followed by two possible outcomes - *ignored bid* with pathway A or B or *activated bid*. The demonstration is designed so to show both outcomes. The final figure of bid events is illustrated in figure 4.12. Each sub figure indicate successive events of bidding, where the red arrow indicates current time.

The RNN forecast model that was recently showcased, *model D*, is used to provide forecasted baselines for each bid event. '2019-09-26 07:00' is chosen as the starting current time of the demonstration, and for the first event (a) in figure 4.12. That corresponds to timestep 9 in the test set. The corresponding bidding horizon is then timestep 10 through 15. That represents the first peak seen in fig. **??**.

The reason for choosing this starting point is two folded. Firstly, using predictions close to the train set in time is more realistic approach because the model most likely will be retrained regularly as new observations become available with time, e.g. each morning. Secondly, the predictions at the chosen current timestep seem significantly better than the rest of the test set, which is partly explained by the first reason. Test data that are less distant from the train set will probably relate more with the train set, and therefore conform better with the trends learned by the model. However, whether or not the forecasts in the beginning of a test are consistently better is later investigated in the section *Investigating online training*.

Briefly about the bid event line
Figure 4.12(a) of illustrate the entered bid for event (a). It is for the bidding period 08:00-14:00 and is entered into the flexibility platform at 07:45. This bid is not bought and the *bid is ignored*. We proceed to the next timestep where new flexibility bids are made, as illustrated in figure 4.12(b). The bid is again ignored. For event (c), the bid seen in figure 4.12(c) has been made by following the methodology. This time, a part of the bid is bought and must be activated. In event (d), the dispatch process goes on, the realized dispatch will be monitored and the delivered flexible power will be measured as seen in figure 4.12(d). Prior to the end of activation, new bid estimates could be made, but for this asset, it seems appropriate to have a rebound period in order for the cooling storage to recover its SoC level.

4.2.6 Behind the curtains of the bid event line

A stepwise explanation is now to be given for the creation of each flexibility bid event seen in figure 4.12, including the underlying processes. The figures 4.12, 4.10 and 4.11 are going to referred to frequently. These figures contain real-time forecast plots, flexplots and the flexibility bid for each event. The letters for each sub-figure correspond to each other and to the events with same letter: event (a), (b), (c) and (d). In addition, table 4.1 contains the forecasted baseline for the bidding horizon in each event, along with true values. The table and all the figures are now shown, followed by the walkthrough.

In the table, each row represents the current timestep and contains the true load on that current timestep and forecasted values for each of future timesteps h. To clarify, the cells (step=10,h=2) and (step=11, h=1) are both a prediction for the realized true value at cell (step=12,h=0). The rows at timestep 9, 10, 11 and 15 correspond with each respective event in the demonstration and to the forecast

plots in figure 4.10, respectively

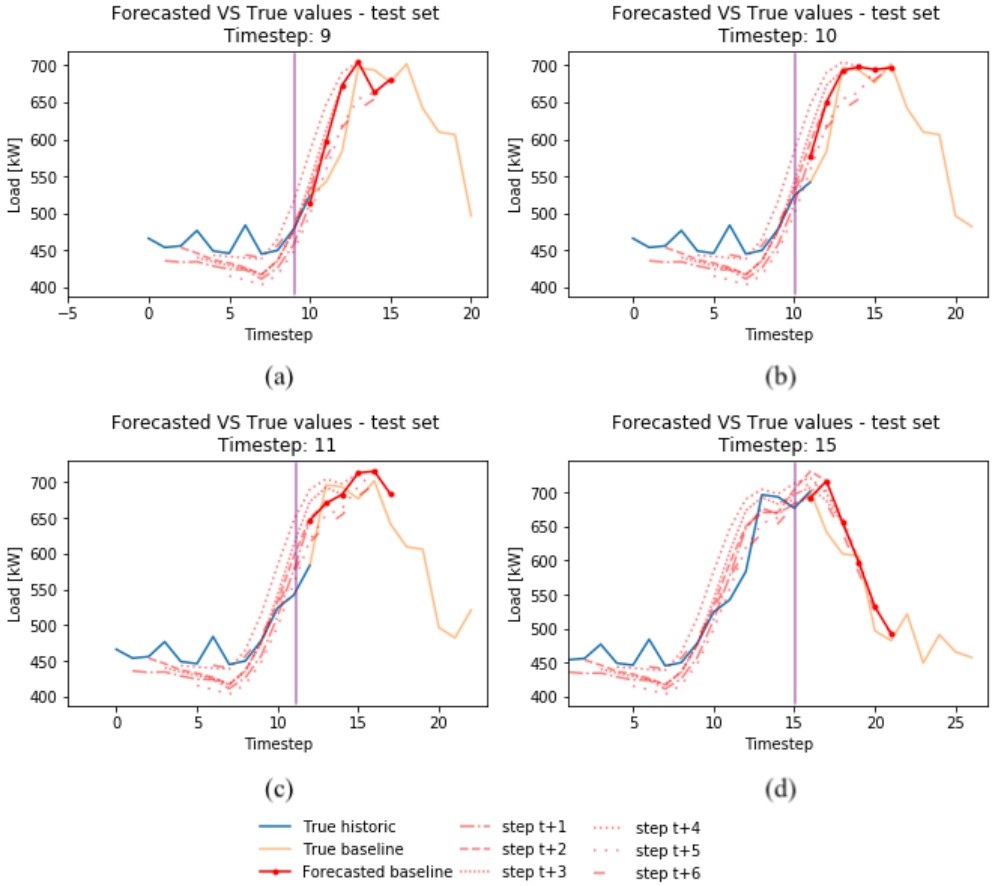


Figure 4.10: Corresponding real-time load forecast plots for events (a) to (d) for the machine room. Each subfigure corresponds to the events of the bidding event line demonstration. Table 4.1 provides corresponding forecast results for each event.

Table 4.1: Table of forecasted baselines and true values for machine room consumption on Sept 23th 2019. The table does provide relevant results corresponding to the events of the bidding event line demonstration. Each row in the table represent one current timestep and contains the true value for that current timeslot, $h = 0$, and forecasted values for the future timeslots in the bidding horizon, $h \in \Big[1, H\Big]$.

	fig 4.12	fig 4.10	True	Forecasted					
event	time	step	h=0	h=1	h=2	h=3	h=4	h=5	h=6
(a)	07:00	9		512.9	596.1	673.2	704.5	663.3	681.0
(b)	08:00	10	524.3	577.5	650.3	692.8	697.9	694.1	696.8
(c)	09:00	11	542.3	**646.9**	**670.6**	**683.0**	**713.2**	**715.5**	**683.5**
	10:00	12	***583.9***	676.7	670.9	697.0	721.4	702.5	641.0
	11:00	13	***696.5***	668.3	704.5	707.2	698.6	664.1	582.3
	12:00	14	***693.5***	683.6	731.0	687.7	657.5	600.0	525.9
(d)	13:00	15	***676.9***	692.1	716.8	655.3	597.5	532.0	493.0
	14:00	16	***701.9***	-	-	-	-	-	-
	15:00	17	***641.8***	-	-	-	-	-	-
	16:00	18	609.9	-	-	-	-	-	-
	17:00	19	606.4	-	-	-	-	-	-
	18:00	20	496.7	-	-	-	-	-	-
	19:00	21	482.4	-	-	-	-	-	-

units: [kWh/h]

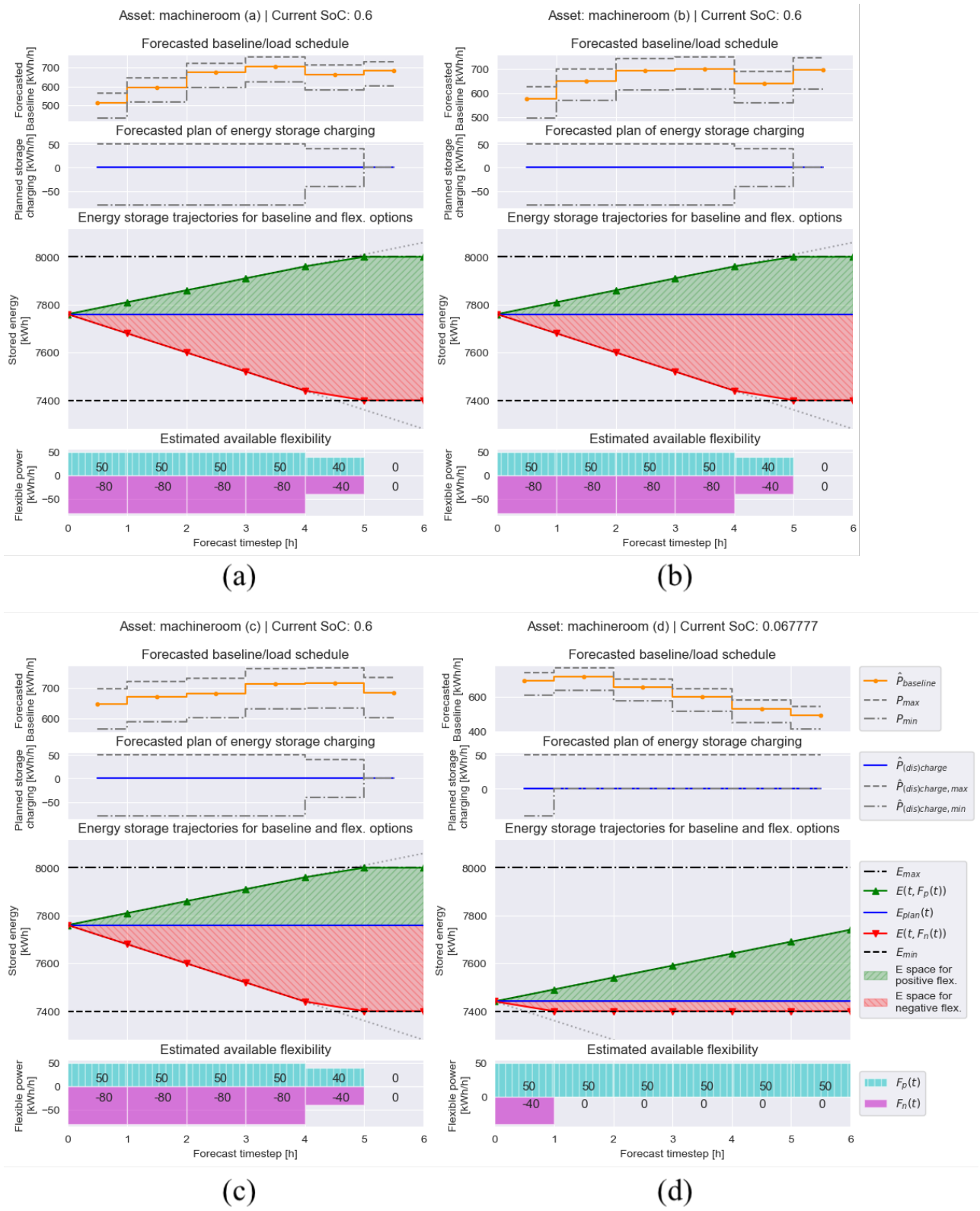


Figure 4.11: Corresponding flexplots for the events (a), (b), (c) and (d).

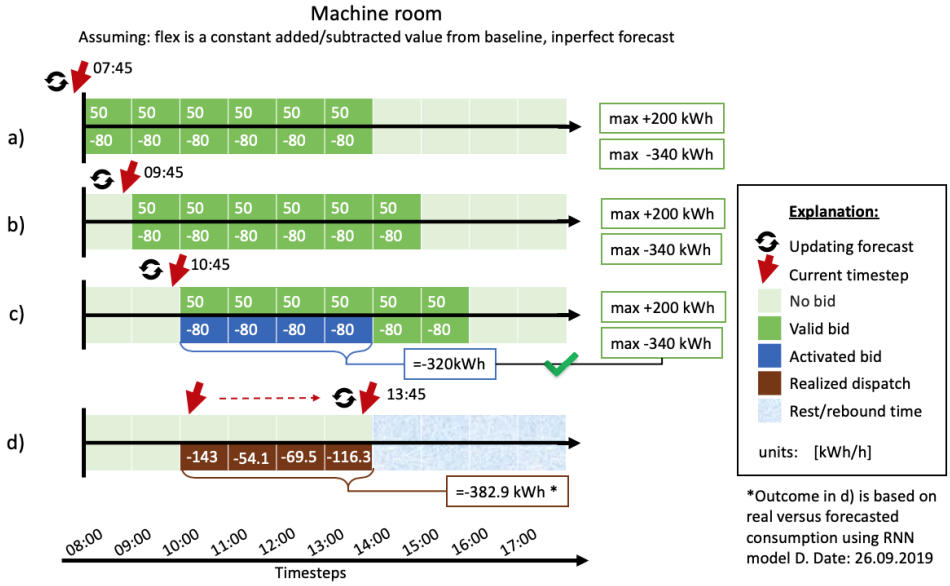


Figure 4.12: Bidding event line plot. Example demonstration of a bid procedure for the machine room asset, on Sept 26th, 2019. Real forecast baselines are used, but assumptions on asset parameters are made. The bid event in each subfigure represent successive bidding horizons where the current time equals (a) 07:00 (b) 08:00 (c) 09:00 and (d) 10:00 $\rightarrow$ 13:00.

Python implementation
A complete script for creating and training the used RNN model, including the forecasts, is found in Appendix B.1. In the bottom of the same script, under *Animation*, is code for creating the real-time forecast plots seen in figure 4.10. The assumed asset parameters and the forecast results are entered into the Python asset modelling framework, event by event. The asset class *Asset* presented in appendix B.2 is used for the purpose. The complete Python script used for simulating each event and for creating the flexplots in fig. 4.11 is found in appendix B.3 and uses the baseline forecast results found table 4.1.

Event (a)
The forecasted baseline (for timestep 10 through 15) is found in table 4.1 where step=9. The corresponding real-time forecast plot is shown in figure 4.10 (a) with red dots. The forecast is input as baseline to the asset model which together with all asset parameters estimate the available flexibility and energy storage trajectories. These estimations are visualized in its flexibility plot, figure 4.11 (a), revealing that the energy storage cannot handle more than 4 hours of full flexible activation. The estimated positive and negative flexibility is formulated as a flexibility bid. The forecasted baseline and the flexibility bid are sent to the flexibility market, as seen in figure 4.12(a). The reader may notice that the +50kWh/h and -80kWh/h is bidded for all slots, in contradiction to the constraints set by the energy storage limits. This is done only to illustrate the freedom in bidding. Suppose that the flexibility buyer does not have to buy the first timeslot, but that they can choose freely. Then, here, while the energy storage still impose constraints on flexibility, is reflected by a maximum and minimum activated energy quota as a part of the bid.

The bid is not activated by anyone at this point. According to the methodology *bid ignored*, time increment as the methodology is reset for new flexibility estimates.

Event (b)
Time has incremented and we are now at step 10, prior to the bidding period for timesteps 11 through 16. The forecasted baseline is the row with step=10 in table 4.1 and plotted in figure 4.10 (b). In a similar fashion to event (a), the flexibility is estimated and a flexplot is made, found in figure 4.11(b). A bid is made and entered into NODES, shown in figure 4.12(b). The bid is ignored this time as well. Flexibility estimates for the next bidding period is prepared.

Event (c) - activation
We are now at timestep 11, and the methodology stages are done exactly as in previous events. In table 4.1, the forecasted baseline is indicated by the numbers

in bold. The true baseline, which we only know because this is a hypothetical demonstration on historical data, is indicated by numbers in bold and italic. Real-time plot of the forecast is shown in figure 4.10(c). The forecast both overestimates and underestimate slightly at different points, but this we do not actually "know" yet in a real-life application. Based on the forecasted baseline, the flexibility is estimated and the energy storage level is simulated, shown in flexplot figure 4.11(c). A bid is created an presented in figure 4.12(c).

As opposed to previous events, the event (c) is subject to a bid activation from a flexibility buyer. They want to buy maximum negative flexibility for the first four timeslots. The corresponding summed energy is -320 kWh. This is all right, because it is below the energy limit of -340 kWh and will not exceed the energy storage limits. The building has now committed to deliver the bought amount of flexible power at each timeslot. It is measured relatively to the *forecasted* baseline, which is important to keep in mind.

According to the methodology, *bid activation* leads to a dispatch of the flexible power and we do now proceed to event (d) in figure 4.12.

Event (d) - dispatching
The process of the dispatch is out of the scope of this thesis, and without further intel, the assumptions made regarding the dispatching ability must be followed. The assumption is a perfect dispatch from the **real** baseline, which is known in this hypothetical backtest. Because of that, the errors in of the flexibility estimates of this demonstration depend purely on the load forecast precision.

The building has now committed to deliver flexibility relative to the forecasted baseline. The delivered flexible power is by assumption based on the true baseline, but the error of deliverance is measured from the forecasted baseline that was entered to NODES. With some math, the resulting delivered flexibility relative to the forecasted baseline, is shown in the event line, figure 4.12(d).

Using equation 3.19, the error of delivered flexible power is
$\big[-63.0, -25.9, +10.5, -36.3\big]$ kWh/h.
Summing up, the integrated delivered flexible power is -382.9 kWh, which is more than what was anticipated and bought by the flexibility buyer. This is most probably not a problem, since the buyer most likely suffer most with too little than to much of what they desire. Hence, the delivered flexibility for timeslots between 11:00 and 13:00 are most likely a more serious crime, because of too low deliverance. Accurate deliverance is nevertheless ideal.

What will be the new state after activation? Although the integrated delivered

flexibility is above the limits, the discharge of the storage is still -80 kWh/h, as this was the main assumption. The total integrated discharged energy of the energy storage equals the bought energy of -320 kWh. This leads to a change in SoC of

$$\Delta SoC = \frac{-320\text{kWh}}{600\text{kWh}} = -0.533$$

leading to a new SoC of $0.600 - 0.533 = 0.067$. Before the activation period ends, the methodology will rerun to make new flexibility estimates. Considering the new SoC, the methodology will be repeated according to the methodology. The new situation with updated flexibility estimates leads to the flexplot shown in figure 4.11(d). The period after the activation period is however denoted as rest/rebound time and is subject for discussion later in the thesis. There is no rest time for the machine room asset, but a rebound effect which results in extra high consumption in order to recover the SoC level is expected.

Another example flexplot for machine room

Now, an example is presented to illustrate how the flexplot may look like with other assumptions for P_{max} and P_{min}. Considerations that are made for the flexplot shown in figure 4.13 is as following:

- $P_{cons} = P_{cons,SS} = \left[\ 696,\ 693 \downarrow 676,\ 701,\ 641,\ 609,\ 606,\ 496,\ 482,\ 521\ \right]$ kWh/h, where the arrow indicates current timestep.
- The maximum power P_{max} and minimum power P_{min} is +100 kW and -200 kW relative to a rolling average (3h) of the baseline consumption.
- SoC is assumed to stay at 0.6, ΔE is arbitrarily picked to be 1000 kWh, so that the time frame seems reasonable.
- E_{max} is arbitrarily set to be 9000 kWh.

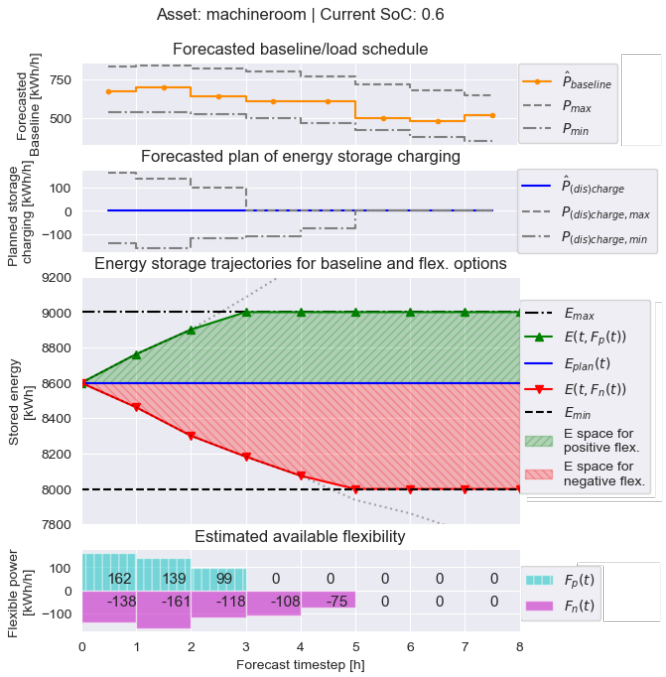


Figure 4.13: Another example flexplot of machine room flexibility where P_{max} and P_{min} are defined from a rolling mean of the baseline.

Code snippets for creating the flexplot (fig. 4.13) is found in the bottom of Appendix B.2, with *EXAMPLE 4* as reference.

In contrary to the other case with the first assumption on P_{max} and P_{min}, this plot looks quite different than the ones in figure 4.11. The estimated flexibility is now different due to a different assumed ability to deviate from the baseline. Note that these estimates are not quite estimates, because the input baseline is historic data and is not forecasted. The goal was to illustrate a different case which might be more realistic. It is not known yet what is more realistic before the asset parameters have been found.

Investigation of online training
The goal of this investigation can be comprehended as:

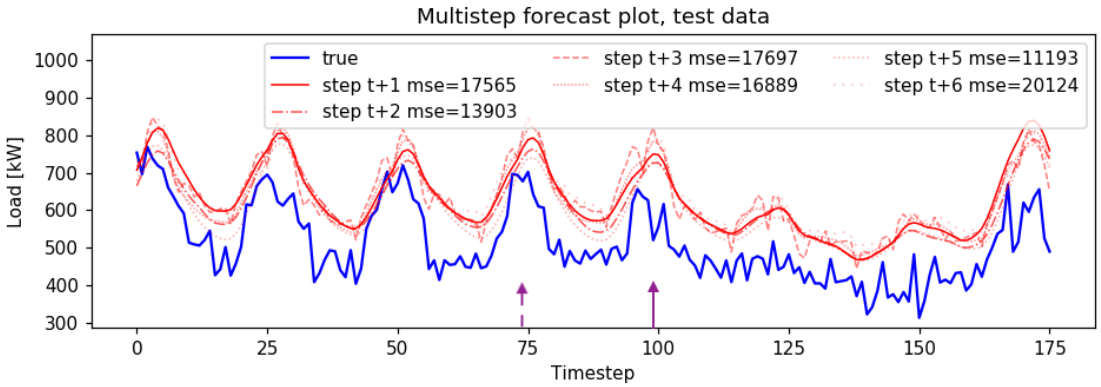
Does online retraining of a forecast model increase predictive performance of short-term forecasts, relative to the exact same model that is not retrained?

As aid for addressing this question, three identical forecast models are created. The models do have a slightly dataset foundation that differs in time. The models are trained on different time periods but all mak forecasts for the same given point in time, in order to check if prediction for the given point is changed. More importantly, do the forecasts become better as the training period moves closer to the given time point?

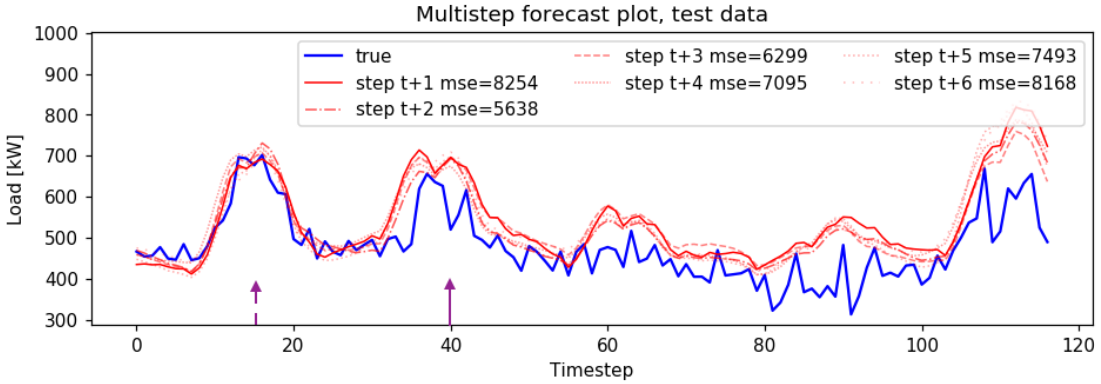
The comparison is best understood by presenting the plot of the forecast results, presented in figure 4.14. Each subfigure presents each model, having a train-test split at slighly different occations. The respective solid and dashed arrows in all the graphs, points at the same point of time and are used for the comparison. Do not get confused by the timestep axis, it can be ignored.

The dashed arrow actually refer to the same peak that was subject for the above bidding event line demonstration. In figure 4.14 (a), the forecasted values do seem to have a clear positive bias, it overestimates. In addition, it seems to lag a bit behind. The results in figure 4.14 (b) us achieved by moving a day ahead and using the observations of that day in the training foundation for new forecasts. The results for the dashed arrow are considerably better. It seems to not have a clear bias and hits very well. The dashed arrow is not included in figure 4.14 (c).

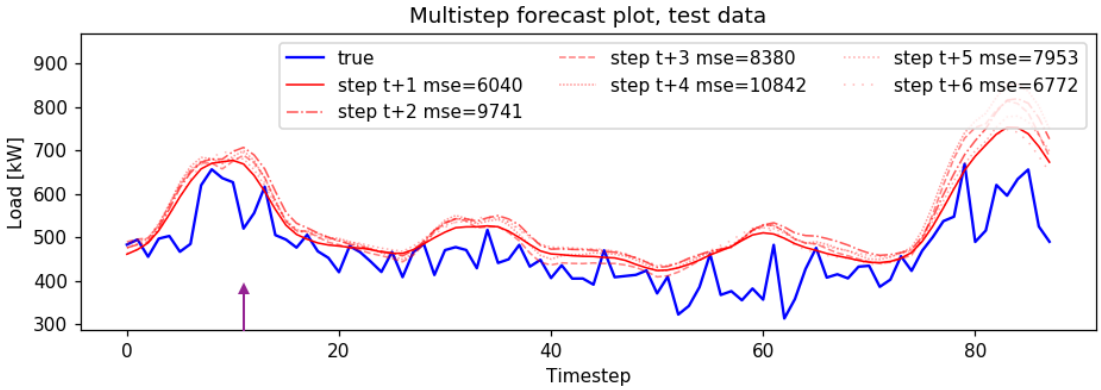
Next comparison is the peak pointed at by the solid arrow. Starting at figure 4.14 (a), the forecast seems to be consequently overestimating, also for the whole test set. In figure 4.14 (b), it seems like the model has learned some important trends, as the forecasts for that peak look better. The interesting thing ti check is whether the model will yield better results for that point with fresher training data. As figure 4.14 (c) shows, there are no visible improvements, except for a slightly smoother forecast. It is hard to conclude anything instantly.



(a) train set size equal to 0.85



(b) train set size equal to 0.90



(c) train set size equal to 0.925

Figure 4.14: Multistep forecast plots from the use of three identical models that have used a slightly different train test split size. The solid and dotted arrows in each figure indicate the same point of time, in order to make a comparison.

4.3 Application for other assets

The following examples are not directly use-cases. The reason for this is they do not directly use data from the use-case, with one exception, and are merely hypothetical demonstrations. However, they are highly relevant, because the considered assets belong to ASKO, although many asset parameters are picked arbitrarily. The data foundation for the water heater consumption is much inspired by data provided by ASKO.

4.3.1 Battery

Flexplot examples for both an active and a passive battery were presented in the *Methodology* chapter. The two following bid event line examples consider a passive battery with the same parameters as the one in the flexplot figure 3.5.

Bidding event line example 1
For this example it assumed that the flexibility buyer *must* activate all of either the positive or negative flexibility if it wants to buy anything. The bid event line example is shown in figure 4.15. The flexibility bid is designed accordingly and is directly restricted by the energy capacity limits. When the negative flexibility bid is bought in (c), the battery will dispatch it during the two hours, followed by a rebound period of two hours to recover the SoC level and bid again.

Bidding event line example 2
Another bidding event line example is found in figure 4.16. The exact same situation as above is assumed, except that the bid formulation now offers a free selection of bid slots. In addition, the flexibility buyer can choose anything in the offered range of flexible power. As seen in (a) and (b), the flexibility bids are always at maximum, not directly limited by any energy storage limits. The important aspect in this case, is overall energy restrictions, here ± 32.5 kWh. As long as total net activated flexible power does not exceed the energy restrictions during any point of the dispatch period, the bid shall be accepted. In (c), some arbitrarily flexibility bid slots are chosen to be activated by a flexibility buyer, and each flexibility slot is not fully activated. The first three bought slots are -10 kWh/h out of the available -30 kWh/h. Thus, -30 kWh of the energy quota is used. The fourth and fifth slot contain +20 kWh/h of bought flexible power each. That sets our energy quota back and up to +10 kWh. The sixth and final slot is full negative activation. In end, the energy quota is -20 kWh. This is checked

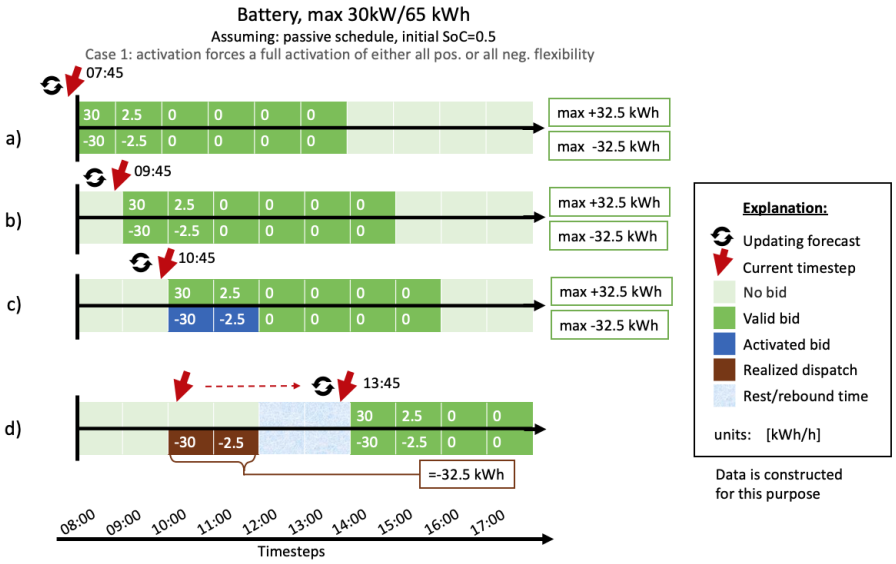


Figure 4.15: Example bidding event line for a battery, example 1.

against the restrictions and confirmed. In (d), dispatching of the bought flexibility shall be completed, followed by a rebound period.

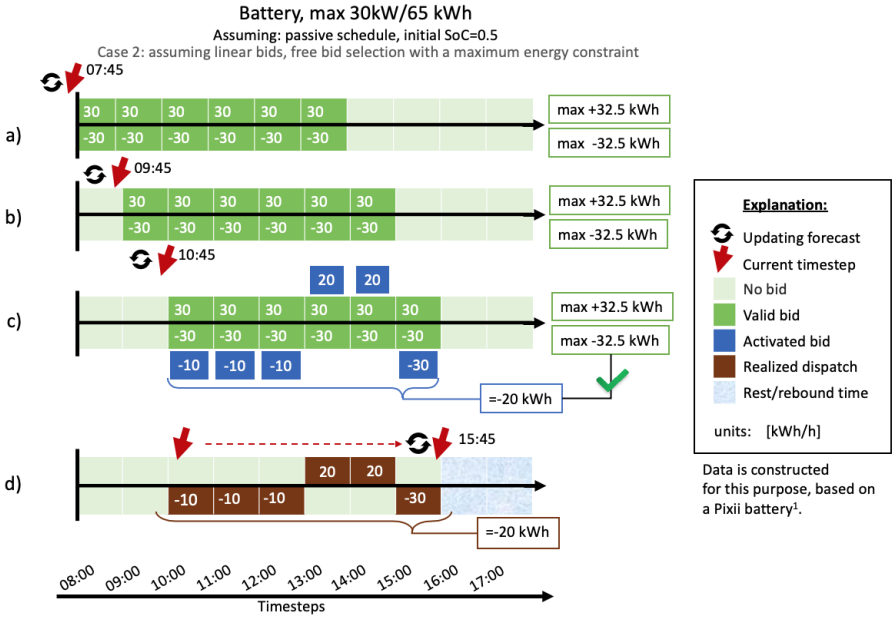


Figure 4.16: Example bidding event line for a battery, example 2.

4.3.2 Water heater w/ alternative energy source

Example bidding event line

Figure 4.17 shows an example bidding event line for a water heater that can replace its consumption with an alternative energy source. As a foundation, real consumption data for the asset has been collected from Sept 4th, 2019. It is assumed that the alternative energy source can run maximum 5 hours and it has a rest time of 2 hours. No forecast is used for this example, or it is assumed perfect forecasts. The flexibility bids for (a) and (b) are ignored, but the first four flexibility bid slots in (c) are bought from a flexibility buyer. 4 hours of activation is within the restriction and the trade is accepted. Since the estimated flexibiltiy was based on the asset to shut off electrical consumption and it will do that perfectly, the delivered flexible power is a success. Any errors that might occur are errors in the forecasted baseline that is entered into the platform, something that should have consequences. Such errors must be detected for *ignored bid* periods whree the true baseline is known. After the dispatch process in (d), there is a rest time of 2 hours and no rebound effects before new flexibility bids are generated.

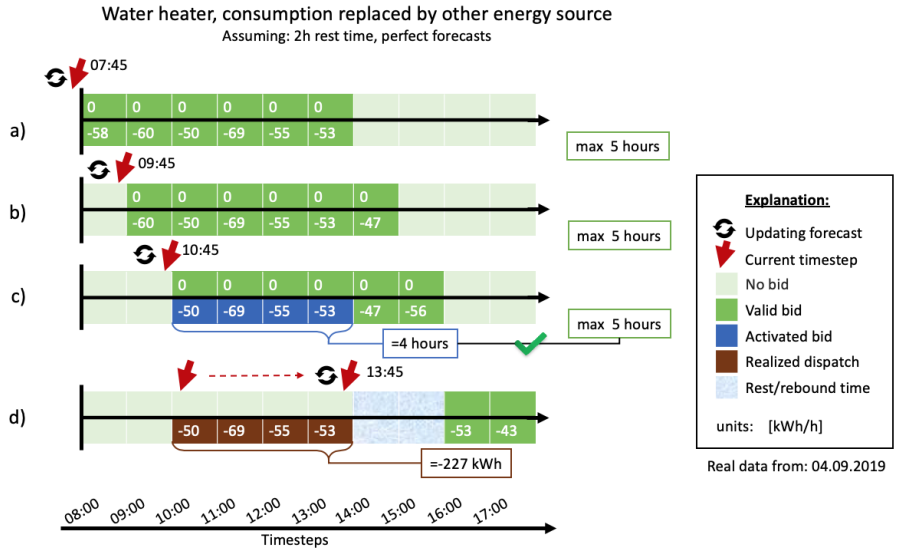


Figure 4.17: Example bidding event line for a water heater that has an alternative energy source.

Example flexplot

The water heater with alternative energy source is now subject to a flexplot demonstration. The available negative flexibility power depends totally on the consumption. In the example, the following assumptions is made:

- Alternative energy source has a max runtime of 4 hours.
- The load forecast of the water heater baseline is assumed to be perfectly known, $\vec{b} = \hat{\vec{b}} = \Big[\downarrow 300, 280, 270, 290, 340, 310\Big]$ kWh/h
- Δt= 0.25
- $P_{min} = 0$ kWh/h, $P_{max} = b$
- $P_{cons,SS} = 0$ kWh/h

These assumptions yield the flexplot presented in figure 4.18. No energy storage is added, but the max runtime is implemented in the Python script. The negative flexibility is clearly equal to the forecasted baseline. Suppose the bid is activated and the consumption is set to zero. What the actual baseline would be if not for the activation is impossible to know. The baseline forecast may have missed.

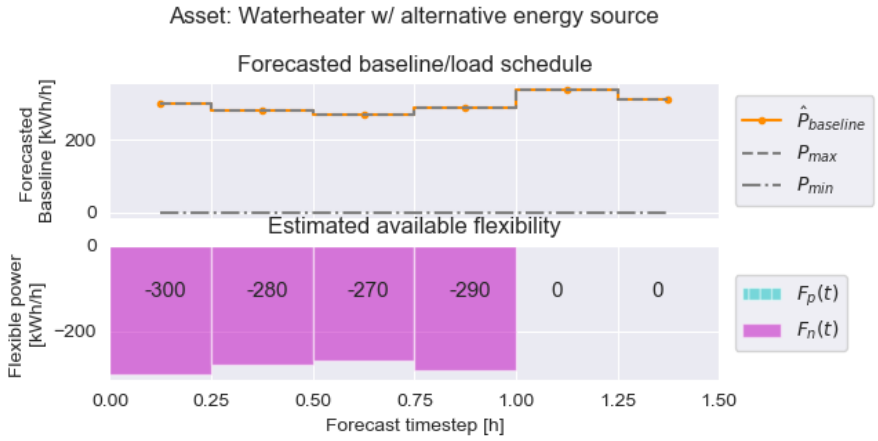


Figure 4.18: An example flexplot for a water heater with an alternative energy source.

Code snippet for creating the flexplot (fig. 4.18) is found in Appendix B.2, with *EXAMPLE 3* as reference.

4.3.3 Water heater w/ flexible heat storage

Example flexplot
A flexplot is now demonstrated for a water heater with a heat storage. In the example, the following assumptions are made:

- $T_{max} = 90$ °C $= 363$ K
- $T_{min} = 60$ °C $= 333$ K
- The load forecast of the water heater baseline is assumed to be the same as for the water heater with alternative energy source,
  $\vec{b} = \vec{\hat{b}} = \Big[\downarrow 300, 280, 270, 290, 340, 310 \Big]$ kWh/h
- The heat capacity of medium is $c_{p,water} = 4.18$ kJ/kgK $= 0.00161$ kWh/kgK
- The mass of the water content is $m = 4000$ kg
- Steady state consumption still equals the baseline.
- Initial SoC is 0.5
- Δt= 0.25
- $P_{min} = 0$ kWh/h, $P_{max} = 400$ kWh/h

Inserting the temperature range, heat capacity and mass into equation 3.26, yields $\Delta E = 193.2$ kWh. E_{min} is set to 0. All in all, these assumptions yield the flexplot presented in figure 4.19.

The negative flexibility is equal to the forecasted baseline, but gets quickly restricted by the energy storage. There are some positive flexibility as well. Increasing the T_{max} might lead to higher heat losses from the storage, but neglected here. To increase the flexibility, one can increase the mass or P_{max}. Suppose the bid is activated and the consumption is set to zero. What the actual baseline would be if not for the activation is impossible to know.

Code snippet for creating the flexplot (fig. 4.18) is found in Appendix B.2, with *EXAMPLE 3* as reference.

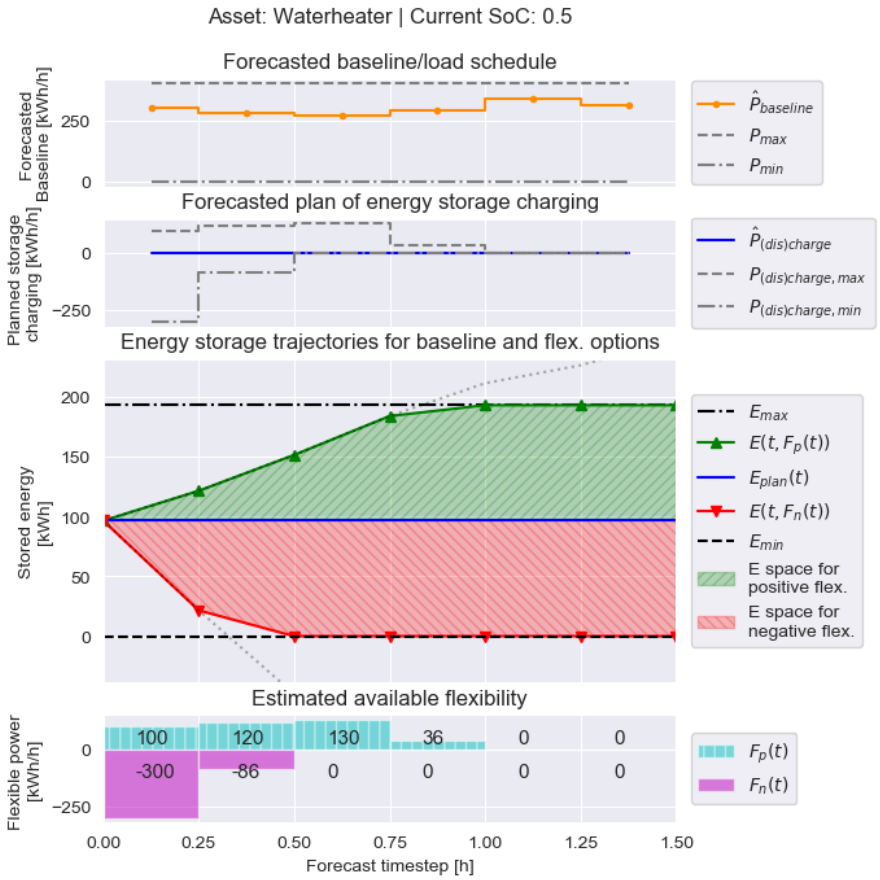


Figure 4.19: Water heater with heat storage used as an example to present an energy and flexibility plot in a step of the conceptual model.

Chapter 5

Discussion

The discussion section is divided into three sections: Discussion of the RNN results, the use-case and the methodology.

5.1 RNN results

Various RNN models have been experimented with in order to address problem statement **(S2)**. An extensive selection of RNN models with their results are found in Appendix A. They will now be discussed.

In general, the models perform very differently regarding train scores, test scores, whether or not they generalize well and how precise they seem to be where it matters. In other words, the choice of model architecture and parameters yield completely different outcomes. What the desired outcome actually is, is also a discussion. A general weakness of these results is that they do not tell us how the models would perform on other train and test sets than the ones used. A model performing good on this chosen data set does not guarantee good performance on future sets (real life forecasting), but will nevertheless give indications of what a good model is.

Test set versus train set MSE scores
The test score versus train score is much about whether or not the model has learned the correct trends between the real explanatory variables and target. If it has, then both test and train MSE could be low, with the test MSE being slightly

worse. A very good train score and a terrible test score strongly indicate overfitting and is unsatisfying. If the test score is good and much better than the train score, there is a good chance there was a good portion of coincidental luck with the given test set.

Consistency between test MSE score and visual precision
Model predictions could seem better in a prediction plot than what the MSE score indicate. An important topic to discuss is whether or not MSE scores provide a meaningful metric for determining if a model does what we want them to do. First of all, what we want the models to do can be different for different purposes. For example, one would perhaps want a model to rather hit the peaks than the rest of the set. Another example is that one wants to predict the first 6 or 24 hours in the test set accurately and the rest is uninteresting. Second of all, the scores do not tell anything about the volatility and stability of the forecasts. Such things can be seen in a forecast plot and perhaps with statistical tools. In some cases, it is desirable with smooth forecasts rather than very volatile ones. On the other side, volatile forecasts could perhaps catch smaller trends, such as peaks, better, but are perhaps more sensitive to noise in the data. A solid and accurate feature set will give a good foundation for explaining every little trend in the target, but a good model will actually be able to learn them properly and ignore any noise.

In this use-case, it is important to have stable forecasts which catches the peaks. It is desirable to have good forecasts for the immideate future. Since Hafslund probably is most interested in negative flexibility, it is better to overestimate the consumption rather than underestimate it in order to not overestimate the available negative flexibility.

Discussion around the results For the results in **One-step forecasts 1**, table A.1, the trend is that all the train scores are better than the test scores. The train scores seem consistently good, but the test scores are very different. The worst test score is 11656 for model *10302019_2000*. It happens to be that this model also has the best train score. That indicates it has overfitted on the training set and weren't able to generalize enough. It may have considered noise from the training set to be important for explaining the test target, when it was actually not. However, when looking at the forecast plots, they reveal that there is not much noise in the forecasts. The forecasts are actually pretty smooth and the predictions for the train set hit very well. Even though the test score was horrible, they seem to have generalized well and catch the trends. It is consistently overestimating, something that could be considered valuable to the specific use-case if there is a lack of accurate models. In addition, if it was not for the bad forecast on the last peak, the test MSE would have dropped considerably. This is a good example that

the MSE is not representable for where we want good forecasts.

The two best models regarding test score are *10292019_1915* and *10302019_1110*. They have test scores of around 4800. The first has the worst train score of them all, whereas the latter actually almost has the best train score of them all. For the first, the forecast plot reveals that many of the peaks were not reached in the train set, but they were in the test set. The second model (*10302019_1110*) does not seem to have the same problem. It hits the peaks in the train set and performs well on the test set. It has a relatively simple structure of two LSTM layers with 48 units each, as opposed to one single LSTM layer with 400 units. The models with a Masking layer seem not to work well and only worsen all the test scores. That could be explained by misuse and wrong implementation from my side. To this point, it seems like a small model architecture of two LSTM layers with a small amount of units, performs the best. Larger networks with techniques to avoid overfitting are discussed for the multi-step forecasts.

For the results in **One-step forecasts 2**, table A.2, all models have much of the same architecture: two LSTM layers with 400 units each, except for the last one which has 200 units each. Other parameters have been varied. The test scores are consistently worse than the train scores, except for one rare exception. Model *12112019_1327* has the best test MSE and it is better than its train MSE. This model has one distinct difference to the others, namely that it has a linear activation function and not a *sigmoid* activation function. The reason of the good test score and why it is better than train score is probably because of what we see in the forecast plots, fig A.6. None of the peaks and valleys in the train set are caught, as opposed to the test set. This fact tells more about the chosen test set than about the model. First, the model is very weak, from what is seen in the train set forecast plot. The test set seem to be a portion of the data where the peaks and valleys are small. On this basis, it can be concluded that the choice of the test set highly influence the scores. If a model is overall good or not, cannot be concluded before it has been tested on various test sets. We can however conclude that a premise for a good model is that the predictive performance for the train set and the test set should be similar, and that the main trends both in the train set and test set, such as peaks and valleys, are caught by the forecasts.

For the results in **Multi-step forecasts**, table A.3, the trend is good train scores and bad test set scores. The MSE scores in the table is an average score of the separate MSE scores for each forecast step, found in the forecast plots. In general, the results look similar to the previous ones and the first two peaks in the test set, a Thursday and Friday respectively, has considerably good forecasts. The last peak, a Monday, does not. Relatively big architectures with overfitting

techniques, such as Dropout, do not seem to drastically improve results relative to the previous and simpler ones in the one-step forecast section. On the other side, the forecasts for the train sets seem to be very precise and catch the peaks and valleys well. That indicates that the trends of the target in the train set has been learned. Thus, there might be some benefits in large architectures with overfitting techniques. The test set MSE scores are not good, but from what the plots show, they perform considerably well on the first two peaks. The predictions are however unstable because of bad performance on the last peak. As later discussed, the cause to this might be factors other than model architecture.

Model D was arbitrarily chosen from the fact that its forecasts were good for the train set and it performed well on the beginning of the test set, thus giving a good foundation for the use-case. For direct multi-step models, perhaps it could be beneficial to optimize each model for each forecast step. Also, it is thought that a recursive-direct strategy would overcome some of the limits of a direct strategy.

For the many models having bad forecasts for the last peak of the test set, it should be mentioned that this last peak is a Monday of a new week. This day has a new categorical week feature, which has not been part of the training data. These forecasts therefore lack a part of its feature foundation for prediction. Skipping the Monday during evaluation will provide more representable MSE scores. It is also thought that replacing the categorical 'week of the year' feature with an alternative continuous and seasonal feature will provide more stable and better forecasts.

Sources of error
The real world contains uncertainties and can never be modelled perfectly. Nevertheless, imperfect models can be important tools that are sufficient for the purpose. Some errors for RNN models can be identified and minimized. A source of error is wrong measurements in the dataset. The monitoring equipment itself can have an error which means that it does not measure the values precisely. It could be systematic errors (bias) or random errors. Such errors cause noise for the forecast model.

In the use-case, it was found an error in the measured consumption of the machine room components, which constituted the target. This was resolved by finding a way to calculate the correct measurement based on the other assets which seemed to be correct. However, the other measurements can still contain errors, and if they do, there will be an error in the machine room consumption data as well.

Weaknesses

What does the scores not tell us?
It does not tell us how the models would perform on different test set at other data and periods. Therefore, as a further work, it is necessary to test a model on many different train and test sets throughout time.

Length of input data
One important topic regarding predictive performance is the length of the training data. The RNN models in this use case were provided one and a half months of data. According to the literature, neural networks benefit from long and good structured datasets, which may not have been the case here. It is discussed that it could affect predictive performance negatively, however it may not be purposeful to provide longer datasets.

Firstly, these models have been specialized in the trends of August and September. They have most probably not been able learn the various trends throughout the year. When the winter comes, new trends will occur and unless the models are retrained with winter months, they will probably lose their predictive strength. An exception, of course, is if the RNN model has been provided all explanatory parameters (features) that completely explain the target, both through seasonal and systematic changes.

As for the use-case, time features were made for 'hour of the day', 'day of the week' and 'week of the year'. The two first ones will repeat themselves frequently throughout the year and catch daily and weekly trends well. The training foundation for 'hour of the day' is good, there are around 50 observations for each category (01:00, 02:00 ... 23:00). The 'day of week' has a weaker training foundation, but there are at least 7 weeks of observations for each day (Monday, Tuesday,...). The latter, 'week of the year', has a lacking training foundation, because the training set needs to be of at least one year, so that each week is iterated through. In the use-case, the models were trained on a few weeks in the train set, but this learning is not brought into the prediction of unseen weeks in the test set. Either, the model must be trained for a whole year of systematic and structured data, or another approach for implementing seasonal structure must be used. One alternative, which does not demand a whole year of training to catch seasonal trends, is to create a continuous sinus-cosines feature, as proposed by Gábor [37]. Another alternative, as also concluded by the last paragraph, is to instead just have an explanatory feature set which fully explains the target and all its seasonal trends, such as port activity/work schedules and temperatures.

It might not be purposeful to make a complete model for the whole year, because the system can change.

Systematic changes
An RNN model based on historical data without explanatory variables performs good when there is a clear structure in the data. What happens if there is a structural change in the data foundation, such as changing the work plans or changing the properties of the asset, e.g. expanding the cooling storage? An RNN model without explanatory features would suffer from such systematic changes, because previous trends get broken. A model that is trained on previous trends does not work on a new situation where the correlations between the historical data and the target have changed and a new model must be made.

On the other side, if the systematic change is caused by a change of the work plans, then a model that uses work plans as a feature will be able to handle that. Also, if the systematic change is caused by a change in storage size etc., again having a model with such parameters as a part of the feature set will be able to handle these changes. The same accounts for temperature trends.

Finding a good model is time consuming
Another setback for using RNN models, as with many other models, is that it is time consuming, demands knowledge on the field and finding the best predictive model is requiring. On the other side, this is a comprehensive work that is usually required once in the initial process. Finding the optimal model can be done by means of a grid search with cross-validation, which involve training and testing of a vast amount of model parameter combinations and various test sets. It can be run once. It requires a lot of time, memory and computational resources. Grid search was however not done in this work, because optimizing model parameters was not in the scope of this thesis, because it would be too time consuming and demanding to perform with the available resources. A grid search will also reveal which parameters have the most influence and also which parameter combinations yield the best results.

Retraining of a model with new observations will also take some time. Retraining is done because we think that longer training data might improve predictive performance and also that the model will learn new trends that are found in the new observations. Computational time of retraining sets a restriction on how often models like this can be retrained. The computation complexity can be reduced by choosing a simpler model structure than those used. Simpler models may very well perform as good or better. On the other side, it is perhaps not necessary to retrain a model that often. 1 week could work very well as each day. Retraining

every month could even work, but then it is probably important to have many good explanatory parameters in the feature set, as already mentioned many times.

Black box
Another setback of using RNN model is that is hard to interpret the learned trends and rules behind its results, as opposed to other simpler ML methods and traditional statistical methods.

Missing explanatory features
The RNN models in this thesis did not use the explanatory variables that were thought valuable for model performance. It is thought that including features for port activity/work plan and outside temperatures will improve the stability and performance of the forecast model significantly. Then, the model is thought to also cope with structural and seasonal changes. This hypothesis has roots in the physics of heat loss mechanisms, as presented in the theory and methodology chapters and it must be further investigated and tested.

Strengths

Retraining catch new trends
It seems that the RNN models can catch trends and perform very well even on short datasets without explanatory variables. Using RNNs as a load forecast models seem promising, but further work is to use explanatory features and to find optimized model parameters.

Making forecasts is quick
Another pro of RNN models is that they are really quick at making new forecasts. Once the training process is finished, making real-time predictions with new observations only take seconds.

Overcome input data errors
An RNN model might overcome sources of error in input data, because it will still learn the trends between the dataset, although it contains errors, and the target in order to explain the target. However, if the *target* contains errors, the model will learn to output a target with similar errors, which should be avoided.

Investigation of online training

An investigation was done to check whether or not *model D* had consistently better forecasts for a point in time when it got retrained with new observations.

Weaknesses with the experiment
There are some clear weaknesses with this experiment. First of all, the test is not very extensive and the answers that are possibly found, are not general at all. The answers only tell us the case with this specific situation with these specific choices of test sets.

Strengths with the experiment
There are however some strengths with the experiment. Figure 4.14 is now referred to. The results clearly indicate that some new trends have been learned from figure 4.14 (a) to figure 4.14 (b) and the forecasts improve significantly for the whole test set. How can this be explained? One explanation, that is slightly discussed above, is the time feature 'weak of the year'. Further investigation reveals that the arrows point to the peaks of a Thursday and a Friday. The test set in figure 4.14 (a) starts with a Monday and indicates the beginning of a new week, which is unseen for the model. Therefore, the model lacks some seasonal information for this week. Moving on to figure 4.14 (b), the forecasts suddenly improve. This could be explained by that fact that the model now has been trained on Monday and Tuesday of that week. The model thus contains trained weights for that week, which can be used for the forecasts in the rest of that week.

Although it seems like retraining with a new batch improves performance, it is hard to conclude that online retraining works, before the categorical feature 'week of the year' is excluded as a probable cause. Using better explanatory features is always the best choice. Alternative ways to implement seasonal implementation should also be considered. Further work for investigating the effect of online training is a more thorough analysis.

Implications for the methodology

RNN models do very well in some of the cases and good forecasts are essential for good flexibility estimates. Bad forecasts will implicate high errors in the flexibility estimates. The upcoming discussion is about the use-case results, followed by a discussion on the methodology itself.

5.2 Use-case results

Weaknesses of the results

Many assumptions
One of the main weaknesses for the use-case is that the results are merely based on assumptions, especially made for asset parameters at stage 2. The assumptions made for the parameters are crucial for the estimation of available flexibility. Therefore, the weakness of the use-case lays in the choice of the assumptions and whether or not they represent a real case well. It is hard to know which assumptions actually represent a real scenario in a good way, before practical tests are done. Therefore, it is also hard to conclude what the results in real life would be. It was more difficult to find good assumptions for some assets than others. The machine room asset was perhaps the most complex one. Therefore, two cases for the machine room were presented. Based on different assumptions, they did show completely different results for the estimated flexibility and the delivered flexibility as well. For the other assets, the task was simpler, and many of the assumptions were logic.

The following discussion will be about the assumptions that were made for the machine room use-case. Questions that are tried to be answered, are:

Were the assumptions realistic? What were the consequences of the assumptions? Would different assumptions yield completely different results? Were the assumptions crucial for the workflow of the methodology itself or just for the results? Do assumptions make up the pillars of the methodology?

The assumptions on steady state consumption for the thermal storages seemed to be reasonably realistic. There is of course not a steady state situation all the time, at a continuous level, but it seems to be a reasonable assumption over the period of 15 minutes and especially 1 hour. The machine room use-case assumed a constant combined heat capacity (constant grocery volume). In real life, there would be groceries that are transported in and out, having different temperatures, which will change the grocery volume and the energy of the storage. In real life, the total flexible energy storage range ΔE_{range} would actually vary with a variable grocery volume, referring to m in equation 3.26. Estimations on a varying E_{max} and E_{min} could be implemented in a further work. In addition, if incoming or outgoing groceries for example have a lower temperature than the storage, it will add to or subtract to P_{losses} respectively, however this aspect should be handled by a sufficiently good load forecast model. A perfect load forecast model will give

perfect baseline forecasts, which in turn give perfect estimates on the steady state consumption, given a steady state situation.

The assumption on a steady state situation seems to be an important foundation for the rest of the assumptions. If the steady state assumption does not hold in real life, it would throw away the validity of the use-case. Then, another approach to find the steady state consumption must be found, e.g. an extensive physical model.

For the results, assumptions made for the energy storage parameters are not as important as the assumption made for the power parameters. If the energy storage parameter would change, then this would only scale up or down the durability of flexibility.

The two other main assumptions are for P_{max} and P_{min}. The estimated flexibility depends directly on these parameters, in addition to the forecasted baseline which already has been discussed. Look at figure 4.11 (a) versus figure 4.13. The important difference in the estimated flexibility between these flexplots, lays in the assumption for max and min power. As illustrated, the choice of max and min power yielded very different results for estimated flexibility.

Determining P_{max} and P_{min} in practice could be done with step-responses. It is merely about mapping how much power the asset can deviate under many different circumstances and potentially creating a function for P_{max} and P_{min}.

Look at figure 4.12 (d), showing the bid event line for the main machine room use-case. It was assumed that the dispatching ability was perfect. Delivered flexible power was assumed to be the true baseline in addition to the bought flexibility. That means that the only source of a failed delivered flexibility is due to errors in the forecasted baseline. All of these assumptions are arbitrarily and is chosen just for demonstration. In real life, the dispatch abilities are not perfect and neither will the true realized baseline be available. Whether or not the assumptions were realistic or not is hard to conclude. The figure shows that the delivered flexibility is very far from the committed, but this does not mean that the methodology is a flop. The result is completely dependent on the choice of correct the asset parameters. Another aspect of it all is that the dispatch system may have the intelligence and ability to compensate and correct the dispatching according to match the committed flexibility. The wrong numbers in the results of delivered flexibility do not mean that the methodology is wrong, but merely that one needs better inputs to the stages and conduct practical experiments to get correct parameter.

The discussion shows that the methodology itself is not weak, but that the results

are very sensitive to the choices and assumptions for the asset parameters. Good results relies on reliable parameters. Many of the choices do have a significant impact on the resulting flexibility estimates. The dispatch process itself depends on the control system. Good asset parameters will incorporate the abilities of this control system. It is time to move onto discussing the strengths of the use-case.

Strengths of the result
The weaknesses reveal that that results are very fragile to the choices and assumptions made in the various stages. Baseline forecast precision affect the ultimate results as well. The good thing is that they do not disprove anything about the concept. The methodology itself is very certain and stable where its stages is very flexible and can be improved to improve the end results. For example, RNN models could be replaced by other forecasting models in order to yield more precise forecasts. The methodology does not really depend on any of the assumptions and chosen parameters, although its success relies on them. Even if there is a lot of uncertainty in the parameters, they do not affect the workflow. All of the assumptions and parameters can be adjusted and improved in order to yield the desired flexibility estimate accuracy.

The results that are seen in fig 4.12d) are highly dependent on the assumptions and choices made, and the failed delivered flexibility is specifically due to unprecise baseline forecast. The uncertainty lies in the choices of stage approaches, assumptions, parameters, the control system and etc. It can be concluded that the methodology itself has a minor uncertainty and do not depend on any assumptions. That is a major advantage of the methodology. Another pro, in case the results are not satisfactory, then the stages or parameters can be changed for the better.

5.3 Discussion of the methodology

In general, the weaknesses of the methodology are many of the ones discussed for the use-case above. The methodology contains sources of errors for both flexibility estimates and the dispatch process.

In general, the major sources that cause errors in the flexibility estimates have been identified as:

1. Unprecise baseline forecasts

2. Wrong estimates on the ability for an asset to deviate, regarding power
3. Wrong storage parameters

The first error source to flexibility estimates is unprecise baseline forecasts. Precise load forecast provide valuable information on how the asset is going to behave and do directly affect the flexibility estimates. The RNN forecast models are already discussed above and will not be repeated.

The second error source is the power parameters of the asset. If the parameters P_{min}, P_{max} and P_{SS} does not represent the reality, the ultimate flexibility estimate will neither. In general, if there are many external uncertainties, the parameters may be impossible to determine perfectly. Practical experiments that can yield sufficiently accurate estimates, will nevertheless be valuable for assessing the flexibility. It is more difficult to find these parameters for some assets, especially the machine room asset, than others.

For the machine room asset, the ability to dispatch flexible power must be measured reliably in order to provide representable parameters. For all the other assets, the dispatch abilities, and thus the power parameters, seem to be easier to determine.

The third source of error is wrong energy storage parameters. If they are wrong, we will have wrong information about the amount of flexible energy we can tweak around with and for how long. Errors could lead to overestimating the durability of a flexibility bid. First counteraction is to remove the primary error sources and to actually yield good estimates of the energy storage level. For a thermal energy storage, precise temperature measurements and exact knowledge on the combined heat capacity (incl. masses) are essential. Nonetheless, considering that there are some errors in the energy storage parameters, the methodology can add some mechanisms to avoid overestimating the duration of flexibility. First is to reduce the energy range with some safety margin. Second is to subtract a safety margin from the estimated flexible power.

Another source of error to the methodology is uncertainties in the dispatch abilities itself, which depend on the control system, or dispatch system. Errors in unprecise dispatch systems are not errors of the methodology itself. They will however lead to errors in the delivered flexibility, thus making the flexibility estimates "wrong". Uncertainties in the control system is an external uncertainty and is not an uncertainty of the methodology itself and is out of the scope of this thesis.

When that has been mentioned, the methodology for assessing flexibility and the

control system for dispatchment are two parallel systems that depend on each other for the success of assessing available flexibility. They must interchangeably exchange information in order to improve each other. This process is thought to be iterative. For example, a wrong flexibility estimate may have wrongly estimated the assets ability to deviate from the baseline, which ultimately leads to a failed dispatch of the committed flexible power. Learnings from this error must be fed back to the methodology by updating the parameters, which in turn yield a new flexibility estimate and a new dispatch. The ball is thrown back and forth and it may take time before the process achieve a balance.

Strengths with the methodology

In general, the methodology has been showed to be stable and uncertain itself. It is designed to work on many various types of assets and can be used to calculate both positive and negative flexible power. The success of the methodology is only as good as each stage. The workflow of the methodology may provide accurate flexibility estimates, if the sources of error are being dealt with. With the discussions provided here, it is hoped hope to provide sufficient insights on how the errors can be dealt with.

Various discussion topics

Discussion on the bid format

Now, the bid formulation must be discussed. The available flexibility that is estimated using the methodology does not have to make up the bid. ASKO could for example hold back parts of the bid. They may do so for several reasons. They might for example believe that the flexibility bid has a higher value 3 hours later. More on that is discussed under "Where is the values of DFS?". Another aspect about the bid is the slots, and what slots the DSO can buy. Do they have to buy the first slot and successive slots? Or can they choose to buy the second and forth slot only? If they can only buy the first and any successive slot, and they have to buy the maximum flexible power in each slot, that is what yields the $E(t, F_{p/n}(t))$ pathways in the flex plots and is what impose restrictions on the flexibility estimate. However, if the DSO have the ability to freely choose timeslots and also the ability to choose among a range of flexible power within each slot, this breaks down. In that case, none of the flexibility bid slot must be restricted by potential energy limits. Instead, the bid should come with energy restrictions that puts limits for the maximum activated energy quota. However, it may not be

advantageous for neither the flexibility provider nor buyer to allow free choice of slots, because the estimates may actually become more precise the closer they are to operational time (which was the case here).

Aggregation
Aggregation of flexibility has been left out of the scope of this thesis. Some thoughts on how to successfully aggregate bids, are that it is important to re-assure that the flexibility estimates are not consistently biased all in one direction. They should all even out. All estimates have errors, but it is important to make sure these errors zero out so that they do not add up to one major error in the aggregated estimate. The aggregated error can be calculated as a tool to identify consistent bias and prevent it.

Rebound effects/rest time implementation
Rebound effects and rest times have been briefly introduced, but not properly discussed. That is because there are many ways to solve it. One must remember the overarch goal of our quest, and that is to help with grid balance, relief the grid and to solve local congestions. Therefore, it is important that activation of one flexible source for solving a local congestion does not lead to another local congestion. Rebound effects might lead to that, and this must be considered. Some suggestions are as following. To include estimates of the rebound effect as a part of the bid, so that this information is available for the DSO when they activate a bid. When they know the effects of a bid activation, they can either seek alternatives, activate it and later buy more flexibility to avoid it causing new problems or activate it and just confirm that it does not cause new trouble elsewhere. Another solution is that the rebound effect is estimated and included as a part of the bought bid itself. Another aspect to mention is that NODES and some other flexibility platforms want to include an automatic rebalancing feature. The idea is that a flexibility provider which is left out of balance with rebound effects after activation, could rebalance their portfolio by trading from other markets (ID/DA/reserve markets/other flexibility markets) that are connected to NODES.

Problems with weak forecasts
Do weak baseline forecasts have any other negative effects than those already mentioned? On aspect to consider is that the forecasted baseline is input to the flexibility market platform and available for the DSO to see. It is very probable that the DSO base their grid simulations on entered baseline forecasts. Should there be any consequences on entered baseline forecasts that are very wrong? There probably should, because the DSO might perform actions on a false basis. Although uncertainties in RNN forecasts are left out of the scope of this thesis, it should be included in the further work. Then, if a forecast is known to have

a large amount of uncertainty, the uncertainty could be included as a part of the entered baseline for the DSO to know.

Problems with weak dispatch systems
What are the consequences of weak and unprecise dispatch systems? It would essentially lead to a failed deliverance of flexibility. However, a very good dispatch system could potentially compensate for errors in the flexibility estimates in order to yield a successful delivered flexibility.

Transfer value to other cases
The methodology itself and the RNN forecast model does have transfer value to other scenarios and use-cases. It is thought that the methodology can have transfer value to other industry buildings and even neighbourhoods. The methodology is flexible for other buildings and assets. as the developed methodology is generic. However, one must figure out how to implement it for new assets and find methods to determine the asset parameters. The developed simplistic physical asset model is generic, but it could also be replaced with a more advanced one. For neighbourhoods, aggregation techniques must be further looked into. Assessing short-term flexibility, and especially a precise forecast model, is also thought to be beneficial for load optimization models. The methodology is also thought to have transfer value for other markets than NODES, such as GOPACS and other platforms that need a methodology to assess short-term demand-side flexible power. Short-term flexibility estimates are also valuable for market players who participates in a DRM direct control program. Lastly, as briefly mentioned, precise load forecast will probably add value to grid simulations, which will help to predict local congestion situations.

Expanded consequence thinking
Now comes some general high-level thoughts on what some broader consequences of this methodology might be, for the building and for the DSO.

If the methodology is implemented in an unprecise manner, then both the building and the DSO might get trouble to succeed using flexibility markets to reach the overarched goal on local congestion management. That is because they wrongly estimate available flexibility which may only worsen the congestion management and not provide a reliable grid operation tool. However, if the methodology is precise and provide accurate flexibility estimates coherent with the dispatch system, then it is an essential tool for entering flexibility markets. The building will get added value for their flexible resources. The DSO will get new grid operation methods, with access to the building's demand-side flexibility. However, it is given that the building chooses to allocate their flexibility into the market, when the

DSO actually need it. The next challenge for the building and the aggregator is to decide a cost for their flexibility, and to find out where their flexibility is worth the most – by internal use or in flexibility markets? The DSO has to make sure that the building gets intel on when they desire their DFS, e.g. give signals on their willingness to pay, so that they actually have access to the building's flexibility when needed.

What happens if all the end-users in a distribution grid make use of their DFS with a methodology that is accurate, and enters the local flexibility market? Of course, that would be of the DSO's greatest interest. They will then be able to choose the cheapest and best alternative for grid operation and even have backups. On the other side, the flexibility providers will have higher competition, and overall competition lead to overall reduction in flexibility costs and revenue. That might make some flexibility providers to leave the game or to use their flexibility internally. It is fine trade-off, but many buildings do possess easily accessible flexibility they did not utilize yet. In many cases, utilizing their flexibility only result in revenues, so they might as well offer their flexibility if the cost for assessing it with a methodology and enter flexibility markets is low. Therefore, aggregators play an important role in bringing DFS into the markets, and as they gain more experience and knowledge, their services will become cheaper. All in all, the grid could be more flexible at demand-side, however there us a right time for everything. The grid and participants must be mature for such an implementation and the grid must perhaps firstly have a high share of VRES and high demand-side complexity. That is anticipated to happen many places. Importantly, it needs to be valuable for everyone who is participating.

Where is DFS most valuable for ASKO and for the grid?
The following discussion is on the problematic regarding where ASKOs flexible power is worth the most. Do they revenue the most for using it internally, e.g. exploit price variations and optimize against power tariffs, or when offering it at the flexibility market? And to whom on the market is it worth the most? NODES says that the value of flexibility could be different for different flexibility buyers, e.g. DSO versus TSO. They want flexibility to have a higher value where it is more needed for grid congestion. Deciding when and where their flexibility is most worth is something the building and aggregator must be able to foresee, in order to maximize their revenues.

Take the following example of a building which optimize the use of their flexibility in order to minimize their costs. The optimized load schedule could be set day before, or even again in the middle of the day. Even if the building has a load schedule with active use of their own flexible assets, they can still estimate and

offer the remaining flexibility using the methodology in this master thesis. But that might lead to not being able to offer it on the market at a time when the market is in need of it. Then the building will miss revenues, and the DSO will also suffer from the lack of available flexibility. Hence, this must be planned, so that the building maximizes their revenue, at the same time as the DSO have flexibility available when they need it. NODES offer an availability contract, which force a flexibility provider to have flexibility at idle.

For a building, there are costs involved when deviating from a cost-optimized load schedule, because that is how the optimized load schedule is defined [14]. Knowing that deviating from an optimal schedule could lead to loss of profit, ASKO and eSmart must know that there is a higher pay-off when offering the flexibility on the market instead. Therefore, they need to be able to predict what the remuneration might be for their flexibility in the flexibility market at different times. If eSmart and ASKO believe that there is a higher pay-off in the market at a point, they will plan ahead and for example make sure their assets are fully charged, or even hold back on their flexibility bidding if they are afraid to miss out a greater pay-off shortly after. In the GOPACS platform in the Netherlands, the DSO sends a message into the market platform with information on when and how much flexibility they need. That gives all flexibility providers the ability to respond to this message and plan accordingly. Communication and collaboration between flexibility providers and the buyers are very important.

Chapter 6

Conclusion and further work

6.1 Conclusion

To conclude, the main goal **(M)** has been addressed and has led to the development of the core result of this thesis: A methodology that assesses short-term flexibility for various assets for the making of flexibility bids into local markets. The flexibility in a generic asset has been identified and involves the use of baseline forecasts and asset parameters to model and estimate the flexibility for multiple timesteps ahead. It takes into consideration the restrictions of a potential flexible energy storage. Making bid formulations could be relatively straightforward and the ultimate bid consists of many timeslots, each containing a positive and negative flexibility bid. A demonstration of a bidding procedure has been developed and illustrates exactly the methodology is used in real-time flexibility estimation.

Many RNN models has been developed to forecast the baseline consumption for the machine room asset at ASKO and has proven to give a working foundation for the methodology implementation. The models have used time-dependent features and historical consumption with a relative short train set length, and the results seem promising. It did learn the trends in the training data and to some degree in the chosen test set. However, for these RNN models to become more accurate and reliable, they have to be optimized, tested on various test sets, include explanatory variables such as outside temperature and work schedules, and include alternative time features. Accurate baseline forecasts will remove one source of error in the flexibility estimates.

The asset model that has been used is simple, yet effective for the purpose. It can be applied for many different assets and constitutes an essential stage of the methodology. It does however require correct inputs which represent the correct properties and constraints of the respective asset. If not, it will lay a false foundation regarding an assets ability to be flexible, and propagate errors to the flexibility estimates. The task of determining correct parameters seem complex for a machine room asset, but easier for the other assets. With correct model inputs, the second source of error to the flexibility estimates can be terminated.

In addition to a generic conceptual description of the methodology, its conceptual implementation has also been thoroughly explained for five specific assets. Especially complex is the machine room asset. The use-case have presented some results for a practical application to a real machine room asset with a cooling storage. This use-case demonstration have successfully proved the methodology to work, although its flexibility estimates were hypothetical because of the assumptions that had to be made for its asset parameters. Its results, being the flexibility estimates, were sensitive to different choices of assumed parameters, thus emphasizing the need for the model to have accurate and representative inputs.

The methodology itself has proven its stability and certainty. It is flexible, with application to many assets either they are consuming or producing, or both. It provides estimates for both positive and negative power. Its results, namely the flexibility estimates, are only as good as the implementation of its constitutional steps. Therefore, provision of accurate load forecasts and asset parameters are crucial for the success of providing accurate flexibility estimates. Nevertheless, if some of its stages are not satisfying, they can be improved for the better, for example changing the forecast model or changing the asset parameters or assumptions. Another crucial aspect is to have a proper dispatch system that have the ability to dispatch the estimated flexible power accurately, preferably also compensate for flexibility estimate errors.

All in all, the methodology seems to work well on paper, and the flexibility estimates are expected to be accurate if accurate forecasts and asset parameters are provided. It is yet to be tested if it will bring success in a real life application, hence the word "Preliminary...".

6.2 Further work, summarized

The feasibility of the methodology was not shown the way it was thought conceptually, mostly due to lack of practical experiments and the many assumptions that was needed to be done. Therefore, the first important work that must be done, is to fully map the correct parameters of the assets and to test out the methodology in practice. During practical tests, one also needs to measure the accuracy of the methodology and address the sources of error, be it wrong asset parameters, the dispatch system or the RNN baseline forecasts. For the RNN forecast model for the machine room, further work is to implement the features for port activity/-work schedule and outside temperatures as it was conceptually thought. That is thought to improve the model performance and its stability during the year and changes ranging from daily to seasonal patterns. Additionally, the model must be optimized, e.g. grid search, in order to find the optimal forecast model. It would be beneficial having a way to implement uncertainties in the future methodology. That could be done either by having a compensating dispatch system or adjusting the model parameters e.g. add safety margins.

Determining the price of a flexibility bid was left out of the scope of this thesis, but is of uttermost importance. The price setting must be done by the building or smart grid company (ASKO or eSmart). Further work is for them to determine a cost of their flexibility bids. They also need to develop methods to determine when they should offer the flexibility on the market, in order to maximize revenue. Hafslund needs to join the conversation and give indications on their willingness to pay and join the discussion on temporal resolution, aggregation levels and etc. Investigating aggregation methods is also a further work. In the future, the ultimate goal is to automatize the whole process of estimation, bidding, activation and dispatching.

Bibliography

[1] Enerdata Yearbook. (2019). World energy primary production — energy production — enerdata, [Online]. Available: https://yearbook.enerdata.net/total-energy/world-energy-production.html (visited on 19/11/2019).

[2] BloombergNEF, 'New energy outlook 2019', 2019. [Online]. Available: https://about.newenergyfinance.com/new-energy-outlook/ (visited on 20/11/2019).

[3] European Comission, 'Communication from the commission to the european parliament, the council, the european economic and social committee and the committee of the regions energy roadmap 2050 (com/2011/0885 final)', 2014. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:52011DC0885 (visited on 19/11/2019).

[4] R. Bacher, E. Peirano and M. de Nigris, 'Vision 2050 integrating smart networks for the energy transition', *ETIP SNET*, 2019. [Online]. Available: https://www.etip-snet.eu/etip-snet-vision-2050/ (visited on 23/11/2019).

[5] Statnett SF, 'Fast frequency reserves 2018', 2018. [Online]. Available: https://www.statnett.no/om-statnett/nyheter-og-pressemeldinger/Nyhetsarkiv-2018/fleksibelt-forbruk-bidrar-til-stabilitet-og-verdiskapning-i-det-nordiske-kraftsystemet/.

[6] Agora Energiwende, 'A word on grids - how electricity grids can help integrate variable renewable energy', 2019. [Online]. Available: https://www.agora-energiewende.de/en/publications/a-word-on-grids/ (visited on 20/11/2019).

[7] CINELDI, Centre for intelligent electricity distribution, 'Annual report 2018', 2019. [Online]. Available: https://www.sintef.no/globalassets/sintef-energi/cineldi/annual-reports/cineldi_annual_report_2018_hr.pdf (visited on 19/11/2019).

[8] Z. Xu, 'The electricity market design for decentralized flexibility sources', *OIES PAPER: EL 36*, 2019. DOI: https://doi.org/10.26889/9781784671433.

[9] J. Radecke, J. Hefele and L. Hirth, 'Markets for local flexibility in distribution networks', eng, Kiel, Hamburg, Tech. Rep., 2019. [Online]. Available: http://hdl.handle.net/10419/204559 (visited on 15/12/2019).

[10] NODES market. (2019). Nodes web page, [Online]. Available: https://nodesmarket.com (visited on 13/11/2019).

[11] GOPACS. (). Gopacs web page, [Online]. Available: https://en.gopacs.eu (visited on 13/11/2019).

[12] NODES Market, 'Nodes white paper - a fully integrated marketplace for flexibility', 2018. [Online]. Available: https://nodesmarket.com/market-design/ (visited on 15/12/2019).

[13] J. Cochran, M. Miller, O. Zinaman, M. Milligan, D. Arent, B. Palmintier, M. O'Malley, S. Mueller, E. Lannoye, A. Tuohy, B. Kujala, M. Sommer, H. Holttinen, J. Kiviluoma and S. K. Soonee, 'Flexibility in 21st century power systems', Thu May 01 00:00:00 EDT 2014. DOI: 10.2172/1130630.

[14] R. De Coninck and L. Helsen, 'Quantification of flexibility in buildings by cost curves–methodology and application', *Applied Energy*, vol. 162, pp. 653–665, 2016. DOI: https://doi.org/10.1016/j.apenergy.2015.10.114.

[15] L. Barth, N. Ludwig, E. Mengelkamp and P. Staudt, 'A comprehensive modelling framework for demand side flexibility in smart grids', *Computer Science-Research and Development*, vol. 33, no. 1-2, pp. 13–23, 2018.

[16] S. Ottesen, 'Techno-economic models in smart grids, demand side flexibility optimization for bidding and scheduling problems', 2017.

[17] Norges vassdrag- og energidirektorat. (2019). Smarte strømmålere (ams), [Online]. Available: https://www.nve.no/stromkunde/smarte-strommalere-ams/.

[18] Statnett SF. (2019). Fast frequency reserves 2018 - pilot for raske frekvensreserver, [Online]. Available: https://www.statnett.no/contentassets/250c2da4dd564f269ac0679424fdfcfc/evaluering-av-raske-frekvensreserver.pdf.

[19] A. Ulbig and G. Andersson, 'Analyzing operational flexibility of electric power systems', *International Journal of Electrical Power & Energy Systems*, vol. 72, pp. 155–164, 2015. DOI: https://doi.org/10.1016/j.ijepes.2015.02.028.

[20] (). Viking link, [Online]. Available: http://viking-link.com (visited on 18/10/2019).

[21] (). Statnett sf, nord-link, [Online]. Available: https://www.statnett.no/en/our-projects/interconnectors/nordlink/ (visited on 18/10/2019).

[22] M. K. Petersen, K. Edlund, L. H. Hansen, J. Bendtsen and J. Stoustrup, 'A taxonomy for modeling flexibility and a computationally efficient algorithm for dispatch in smart grids', in *2013 American Control Conference*, 2013, pp. 1150–1156. DOI: 10.1109/ACC.2013.6579991.

[23] R. Pinto, R. J. Bessa and M. A. Matos, 'Multi-period flexibility forecast for low voltage prosumers', *Energy*, vol. 141, pp. 2251 –2263, 2017, ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2017.11.142. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360544217320005 (visited on 03/09/2019).

[24] Norges vassdrags- og energidirektorat. (2019). Markedssegmentene, [Online]. Available: https://www.nve.no/reguleringsmyndigheten/engrosmarkedet/markedssegmentene/ (visited on 25/11/2019).

[25] Nord Pool AS. (2019). Price calculation, [Online]. Available: https://www.nordpoolgroup.com/trading/Day-ahead-trading/Price-calculation/ (visited on 25/11/2019).

[26] Statnett SF. (2019). Primærreserver - fcr, [Online]. Available: https://www.statnett.no/for-aktorer-i-kraftbransjen/systemansvaret/kraftmarkedet/reservemarkeder/primarreserver/ (visited on 25/11/2019).

[27] D. Engelbrecht, A Schweer, R. Gehrcke, E. Lauen, B. Deuchert, J. Wilczek, H. Schuster and J. Büchner, 'Demonstration of a market-based congestion management using a flexibility market in distribution networks', *Internationaler ETG-Kongress*, pp. 306–311, 2019.

[28] H. Shi, M. Xu and R. Li, 'Deep learning for household load forecasting—a novel pooling deep rnn', *IEEE Transactions on Smart Grid*, vol. 9, no. 5, pp. 5271–5280, 2017.

[29] *Python Machine Learning (Second Edition)*, eng. Packt Publishing, 2017.

[30] *Deep Learning with Python*, eng. Manning Publications, 2017, ISBN: 9781617294433.

[31] S. Ruder, *An overview of gradient descent optimization algorithms*, 2016. arXiv: 1609.04747 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1609.04747 (visited on 15/12/2019).

[32] G. Bontempi, S. Ben Taieb and Y.-A. Le Borgne, 'Machine learning strategies for time series forecasting', in *Business Intelligence: Second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures*, M.-A. Aufaure and E. Zimányi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 62–77, ISBN: 978-3-642-36318-4. DOI: 10.1007/978-3-642-36318-4_3.

[33] N. Srivastava, H. Geofrrey, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, 'Dropout: A simple way to prevent neural networks from overfitting', *Journal of Machine Learning Research 15*, 2014. [Online]. Available: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf.

[34] Pixii. (). Powershaper 30 kw / 65 kwh, datasheet 11878, [Online]. Available: http://www.pixii.com/wp-content/uploads/2019/04/2019-06-27_pixii-sheet.pdf. (visited on 18/10/2019).

[35] R. E. Sonntag and C. Borgnakke, *Introduction to engineering thermodynamics*, eng, Hoboken, N.J, 2007.

[36] *Solar Energy: The Physics and Engineering of Photovoltaic Conversion, Technologies and Systems*, eng. UIT Cambridge Ltd, 2016, ISBN: 9781906860325.

[37] G. Petneházi, 'Recurrent neural networks for time series forecasting', 2019.

Appendix A

An extensive selection of RNN model forecast results

This appendix contains a full overview of various tested RNN forecast models, both one-step and multi-step. Architecture, parameters and prediction scores are given in the tables. Corresponding forecast plots to each model are also attached.

A.1 One-step forecasts 1

These one-step RNN forecast models were developed in the early phase of the work in this master thesis. Therefore, in addition to creating a lagged timesteps dimension dataset for Keras RNNs, the datasets used here also contain lagged values of the time features as extra added features.

A.1.1 Table of scores

Table A.1: Model architecture, parameters and forecast scores for various RNN one-step models, where lagged values of time features is created explicitly as new features.

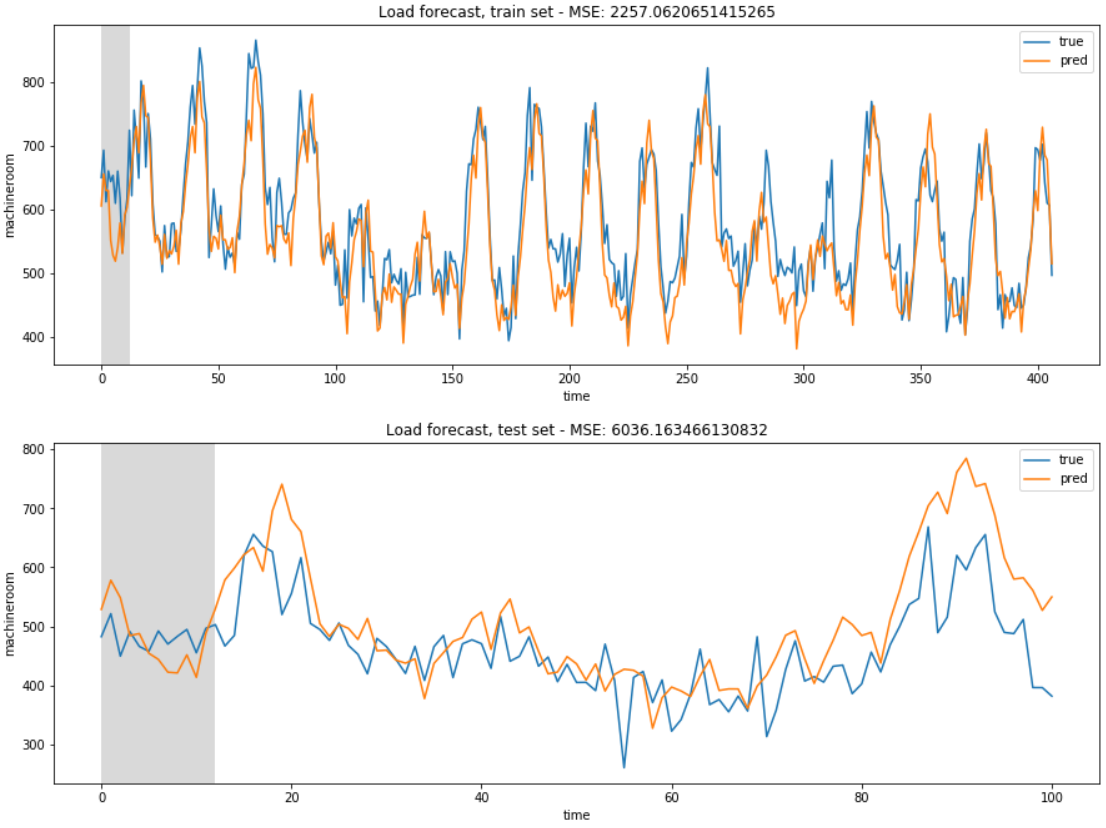
NAME	Model parameters.	mse of train, test
10292019_1715	Layers: (GRU 512), lb=24*7, bs=256	2257, 6036
10292019_1851	Layers: (LSTM 100), lb=24*7, bs=256	1555, 8198
10292019_1915	Layers: (LSTM 400), lb=24*7, bs=256	2275, 4874
10302019_1110	Layers: (LSTM 48, LSTM 48), lb=24*8, bs=256	1471, 4884
10302019_2000	Layers: (Masking, LSTM 48, LSTM 48), lb=24*9, bs=256	1410, 11656
10312019_0900	Layers: (Masking, GRU 256, GRU 128), lb=24*10, bs=256	1629, 7724
10312019_1000	Layers: (Masking, LSTM 256, LSTM 256), lb=24*11, bs=256	1551, 7076

for all: Lagged values as features. act.fct.=sigmoid, loss-fct.=mse on train data, optimizer=RMSprop(lr=1e-3), train_size=0.9, epochs=20, s/e=100, es_pat=5

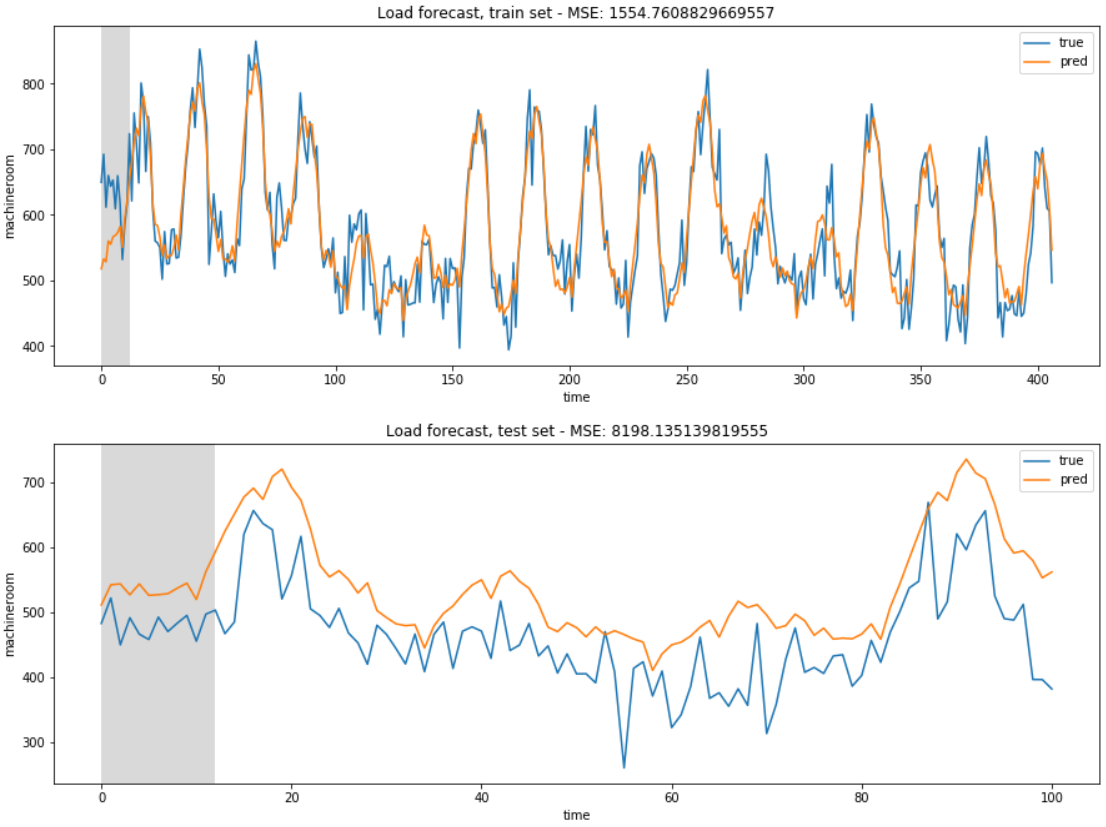
lb = N_LOOKBACK, bs= batch_size, s/e=steps per epoch, es_pat = early stop patience

A.1.2 Forecast plots

Forecast plots corresponding to the models in table A.1 follows on the next pages.



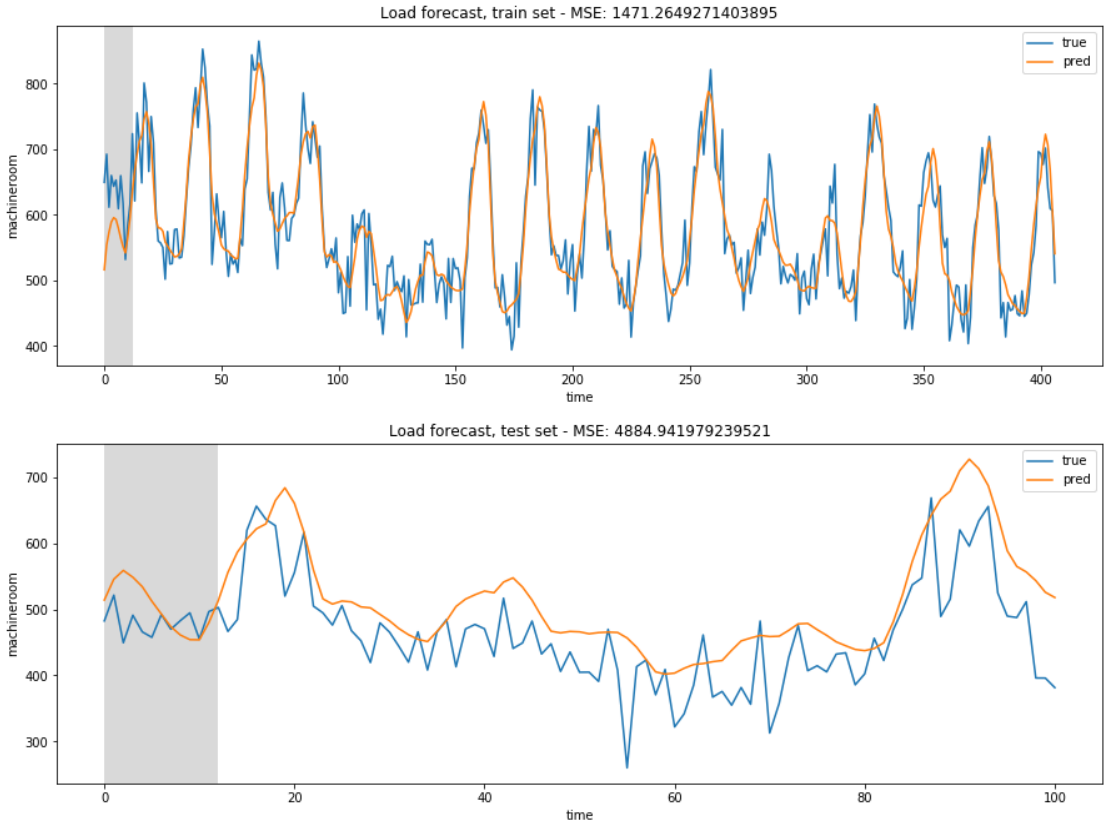
(a) Model *10292019_1715*. Upper and lower plot is for train and test respectively.



(b) Model *10292019_1851*. Upper and lower plot is for train and test respectively.

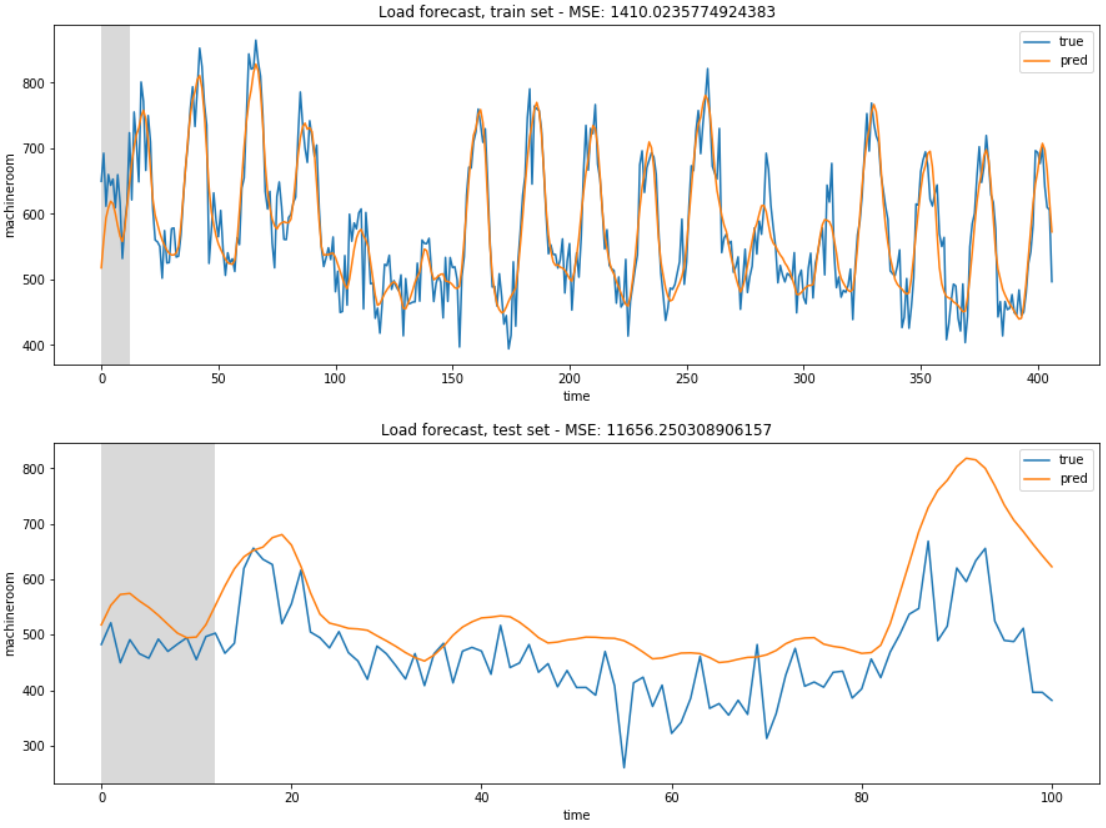
Figure A.1: Multi-step RNN forecast plots of model *10292019_1715* and model *10292019_1851*

(a) Model *10292019_1915*. Upper and lower plot is for train and test respectively.

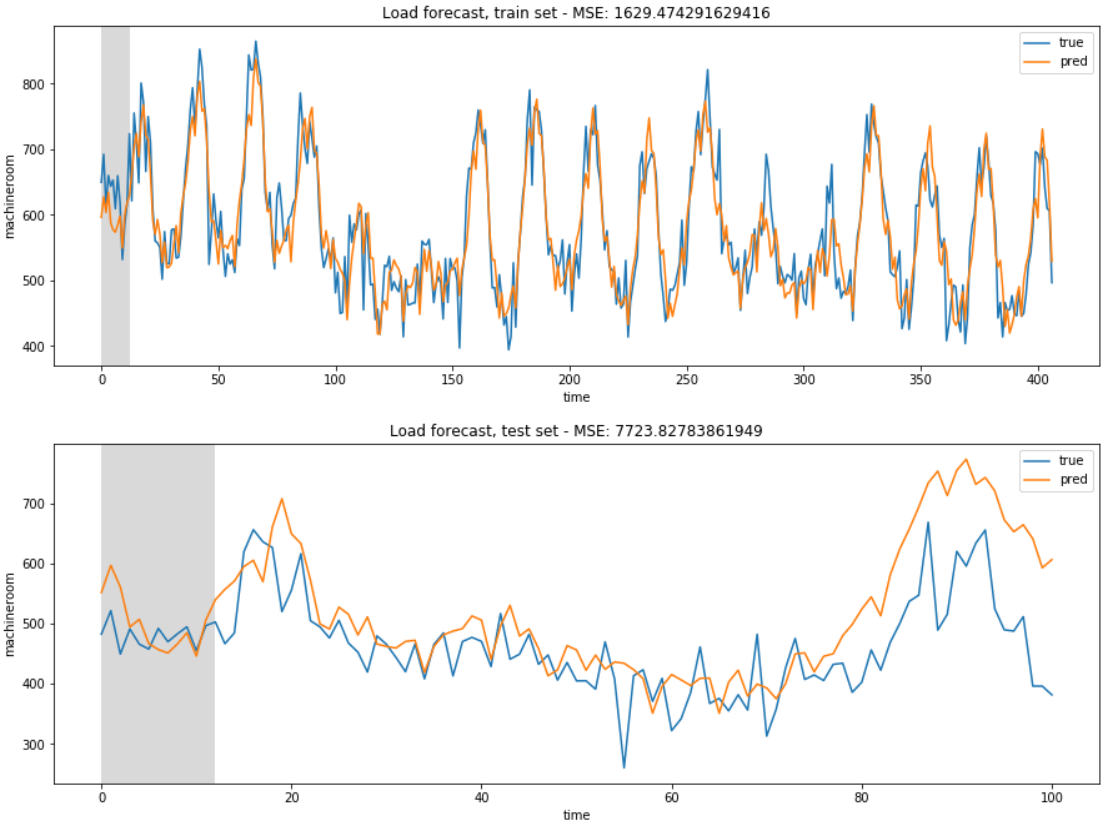


(b) Model *10302019_1110*. Upper and lower plot is for train and test respectively.

Figure A.2: Multi-step RNN forecast plots of model *10292019_1915* and model *10302019_1110*



(a) Model *10302019_2000*. Upper and lower plot is for train and test respectively.



(b) Model *10312019_0900*. Upper and lower plot is for train and test respectively.

Figure A.3: Multi-step RNN forecast plots of model *10302019_2000* and model *10312019_0900*

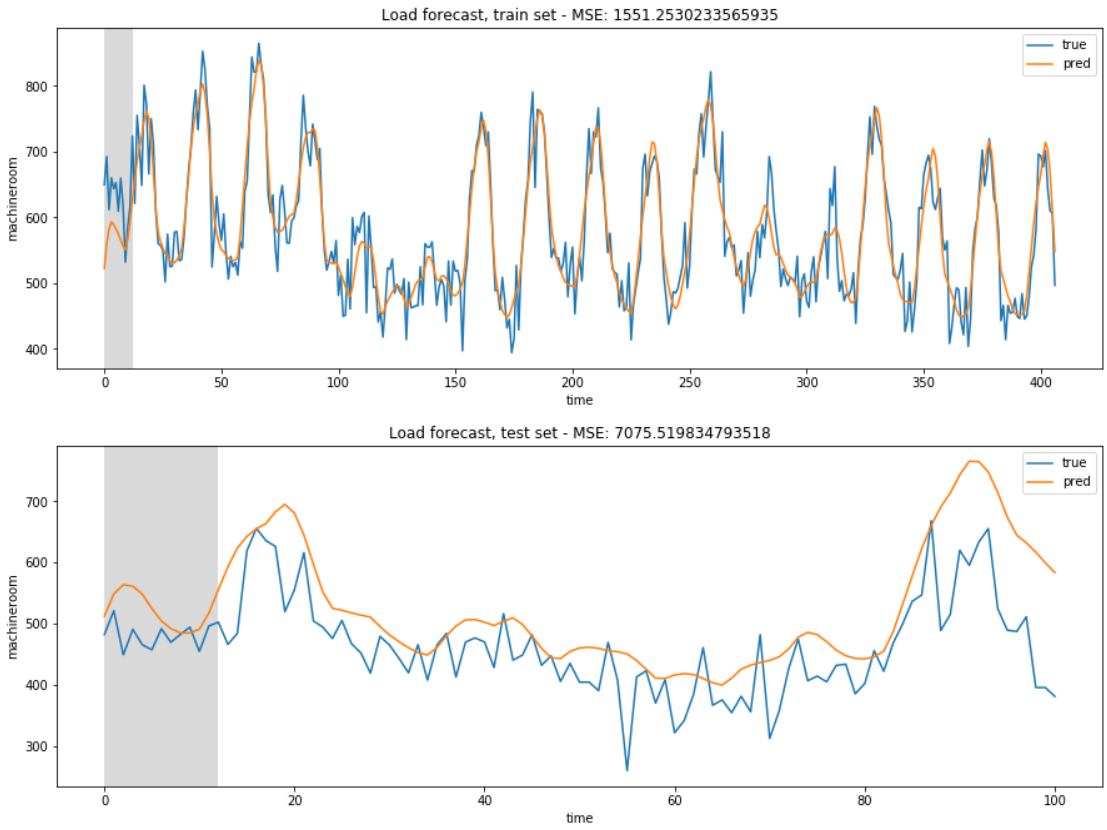


Figure A.4: Multi-step RNN forecast plot of model *10312019_1000*. Upper and lower plot is for train and test respectively.

A.2 One-step forecasts 2

These one-step RNN forecast models were developed in the mid-phase of the work in this master thesis. For these models, the extra added lagged time features are not present in the feature set as opposed to the previous models. However, a timestep dimension according to input shape of Keras RNNs is used to provide time memory to the models. The size of this *memory* of timesteps is determined by N_LOOKBACK.

A.2.1 Table of scores

Table A.2: Model architecture, parameters and forecast scores for various RNN one-step models, where no features is made from algged time features, but instead, the lag memory is represented as the timestep dimension that Keras in Python want for RNNs.

NAME	Model parameters.	mse of train, test
No lagged features*, act.fct.=sigmoid, loss-fct.=mse on test data, optimizer=adam, train_size=0.9, steps per epoch=adapted to bs		
11112019_1710	Layers: (LSTM 400, LSTM 400), lb=24*3, act.fct.=linear, missing data not fixed, bs=64, ep=50, no es, ts=0.9	1785, 7360
12112019_1327	Layers: (LSTM 400, LSTM 400), lb=24*3, act.fct.=linear, bs=64, ep=50, es_pat=5, ts=0.9	5343, 4443
12112019_1344	Layers: (LSTM 400, LSTM 400), lb=24*3, act.fct.=sigmoid, bs=64, ep=50, es_pat=8, ts=0.9	3007, 8119
12112019_1410	Layers: (LSTM 200, LSTM 200), lb=24*3, act.fct.=sigmoid, bs=128, ep=40, es_pat=8, ts=0.9	2473, 8477

lb = N_LOOKBACK, bs= batch_size, s/e=steps per epoch

es (_pat)= early stop (patience)

A.2.2 Forecast plots

Forecast plots corresponding to the models in table A.2 follows on the next pages.

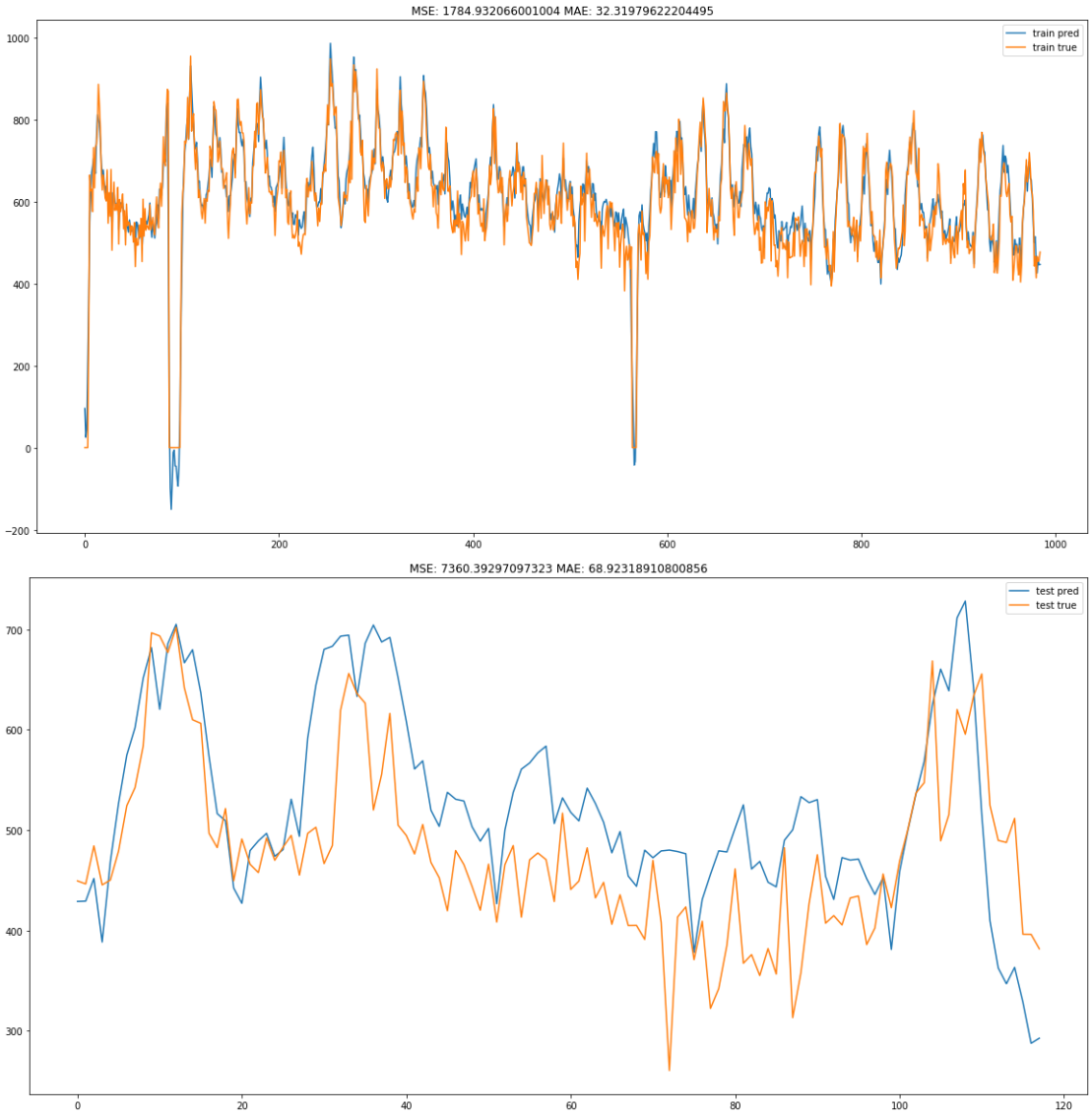


Figure A.5: Multistep forecast plots from model *11112019_1710*. Upper and lower plot is for train and test respectively. As opposed to the coming models, this modesl used input data that was not preprocessed for missing data points, as the dips to zero indicate.

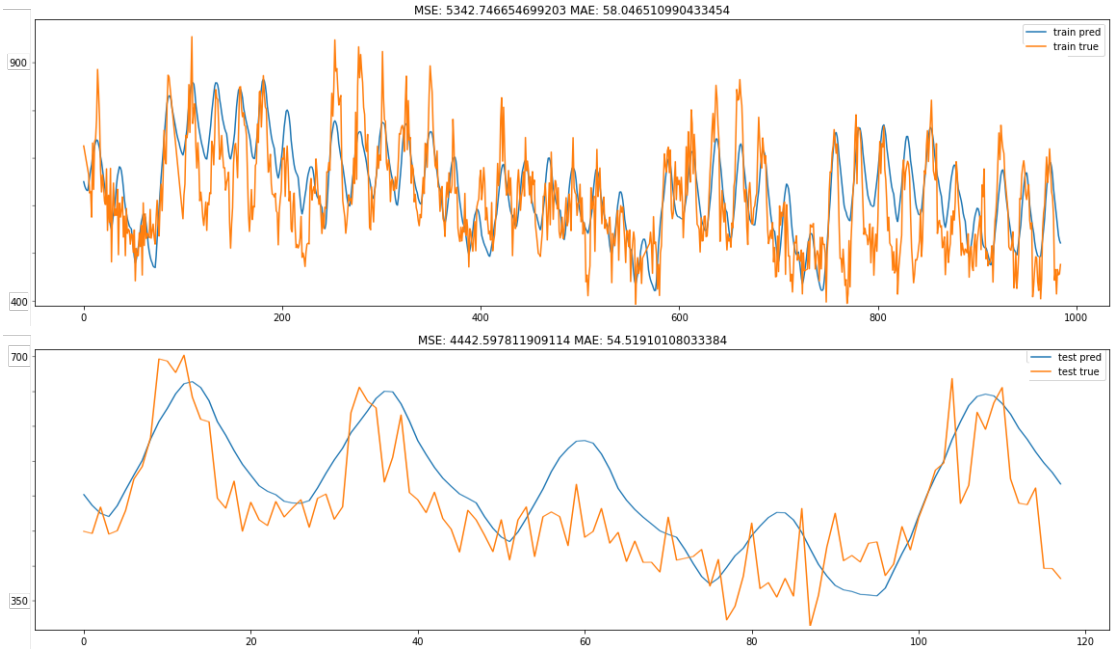


Figure A.6: Multistep forecast plots from model *12112019_1327*. Upper and lower plot is for train and test respectively.

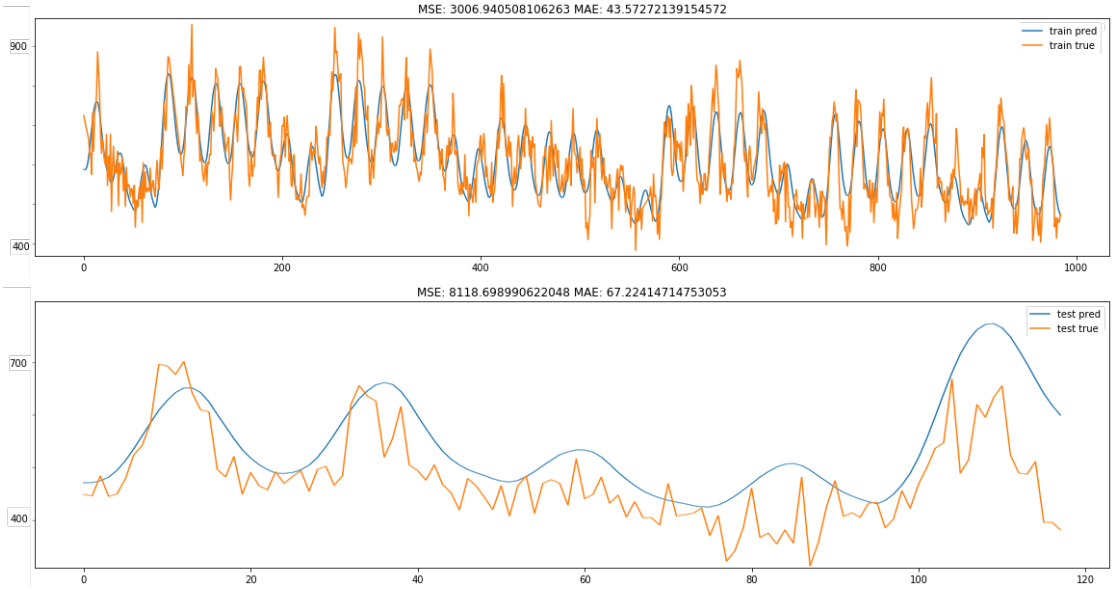


Figure A.7: Multistep forecast plots from model *12112019_1344*. Upper and lower plot is for train and test respectively.

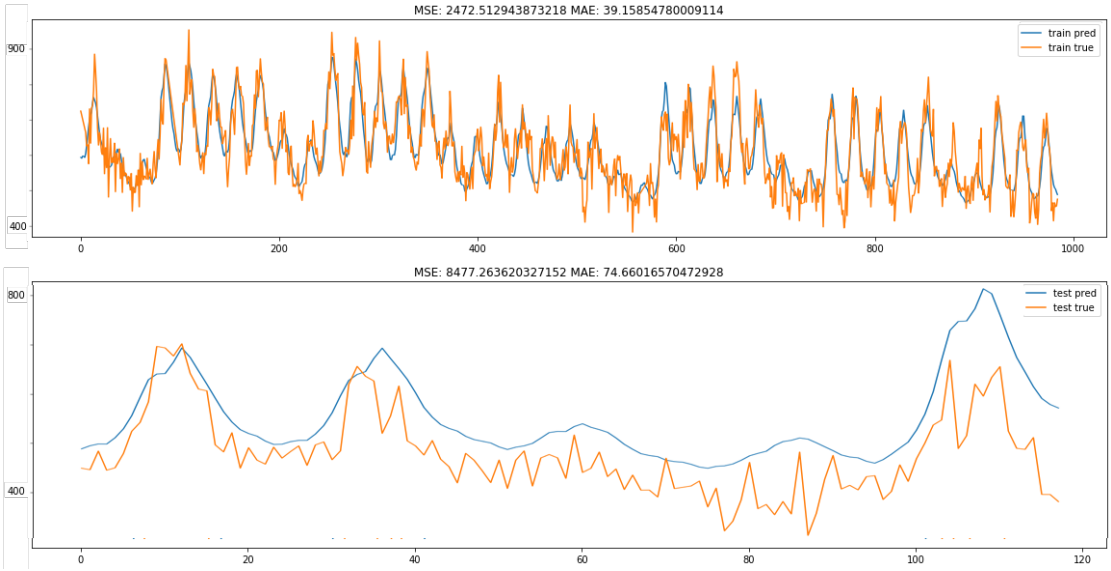


Figure A.8: Multistep forecast plots from model *12112019_1410*. Upper and lower plot is for train and test respectively.

A.3 Multi-step forecasts

These models are direct multi-step models and was developed in the last phase of the work in this master thesis. Models experimented with in the mid-phase are extended into use in a direct strategy. In total, 6 models are presented, each with various model architecture and parameters.

A.3.1 Table of scores

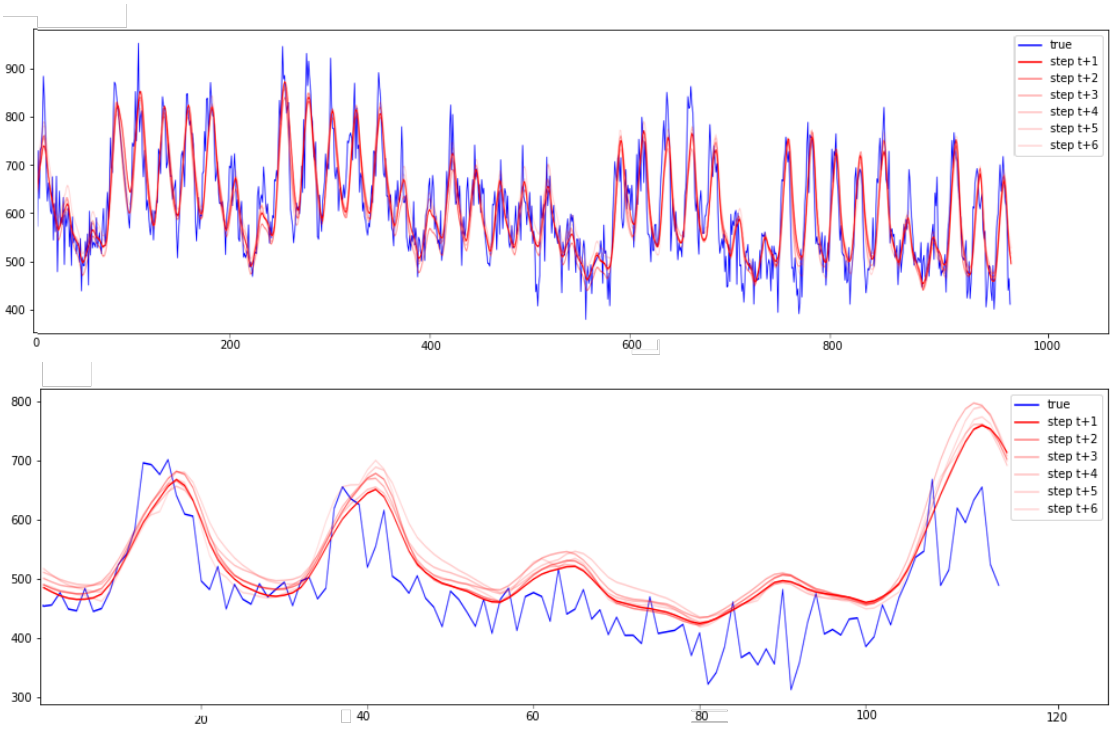
Table A.3: Model architecture, parameters and forecast scores for various RNN multi-step models.

NAME	Model parameters.	mse of train, test
for all: s/e=100, act.fct.=sigmoid, loss fct.=mse on validation data		
12112019_2252 or A	Layers: (LSTM 400, LSTM 400), lb=24*3, bs=100, ep=50, es_pat=8, ts=0.9	-, -
14112019_2020 or B	Layers: (LSTM 200, LSTM 200), lb=24*2, bs=128, ep=50, es_pat=8, ts=0.9	2015.8, 5936.67
C	Layers: (LSTM 400, LSTM 400, Dropout 0.1), lb=24*4, bs=32, ep=70, es_pat=10, ts=0.9	2330.3, 7693
D	Layers: (LSTM 100, LSTM 100, Dropout 0.1), lb=24*2, bs=64, ep=70, es_pat=20, ts=0.9	1187.67, 7157.8
E	same as D, except for ts=0.925	1427.5, 8288
F	same as D, except for ts=0.85	1531.3, 16228.5

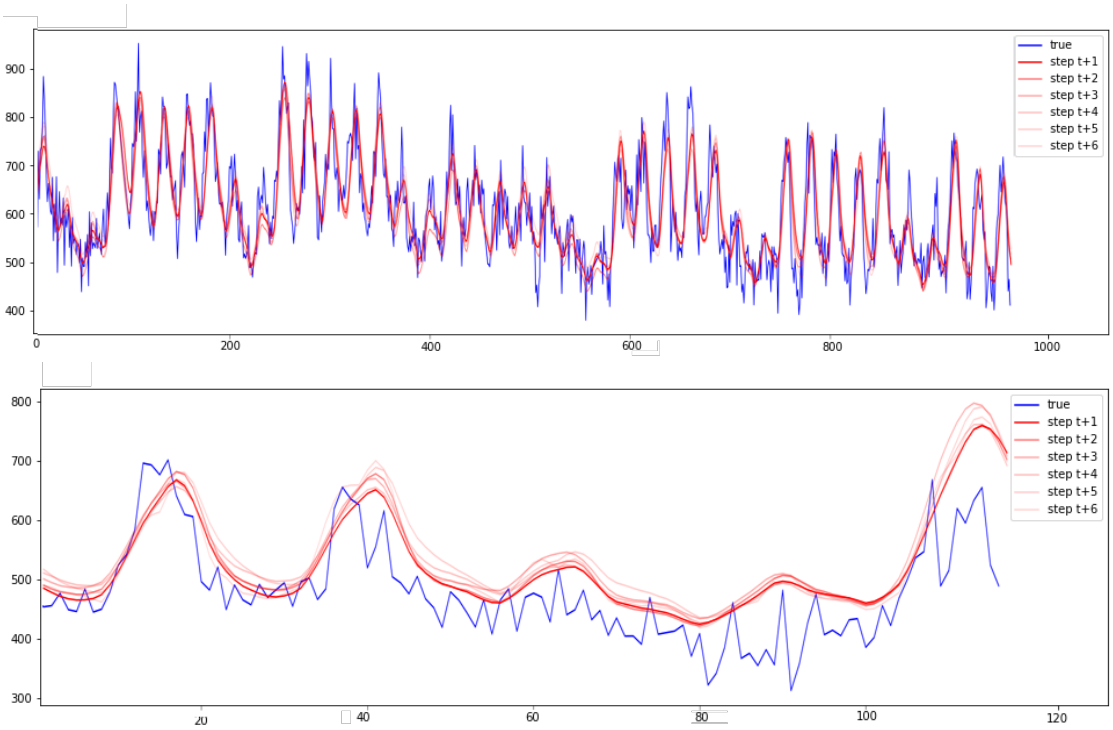
lb = N_LOOKBACK, bs= batch_size, s/e=steps per epoch, ep=epochs,
es (_pat)= early stop (patience), ts=train size

A.3.2 Forecast plots

Forecast plots corresponding to the models in table A.3 follows on the next pages.

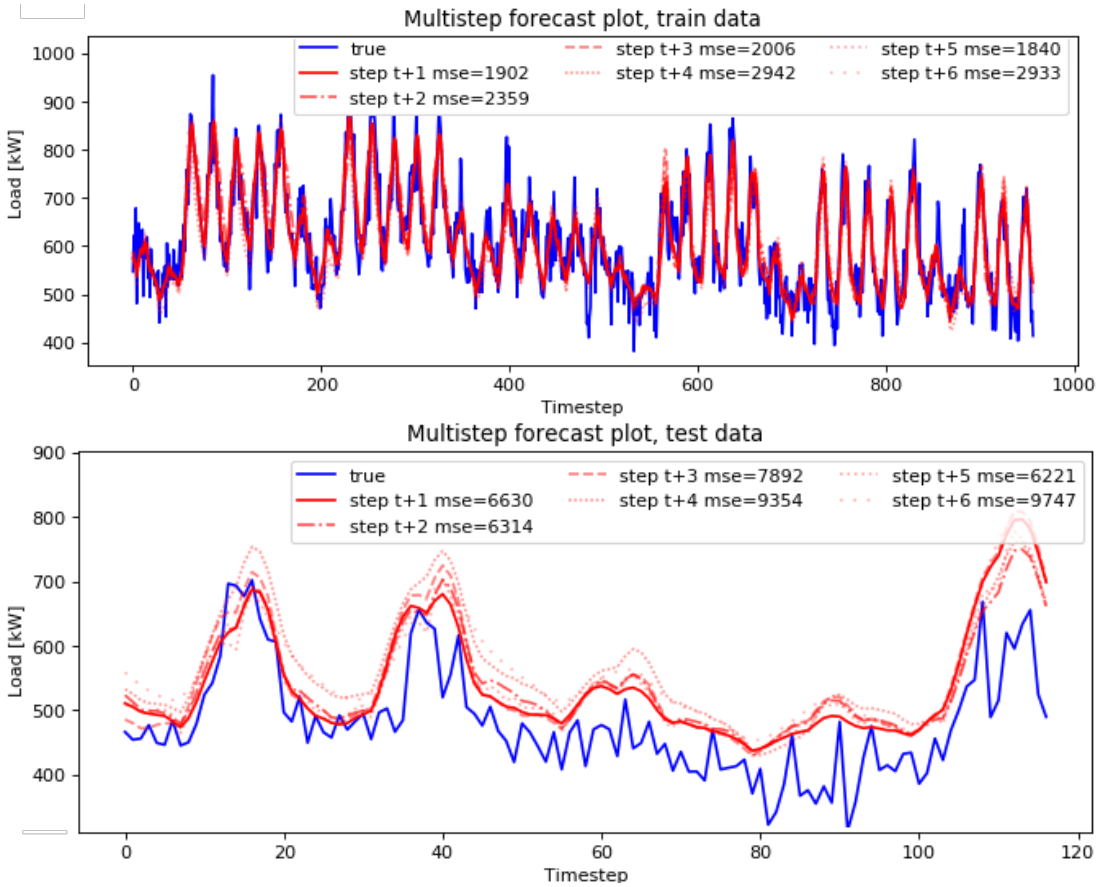


(a) Model *A*. Upper and lower plot is for train and test respectively.

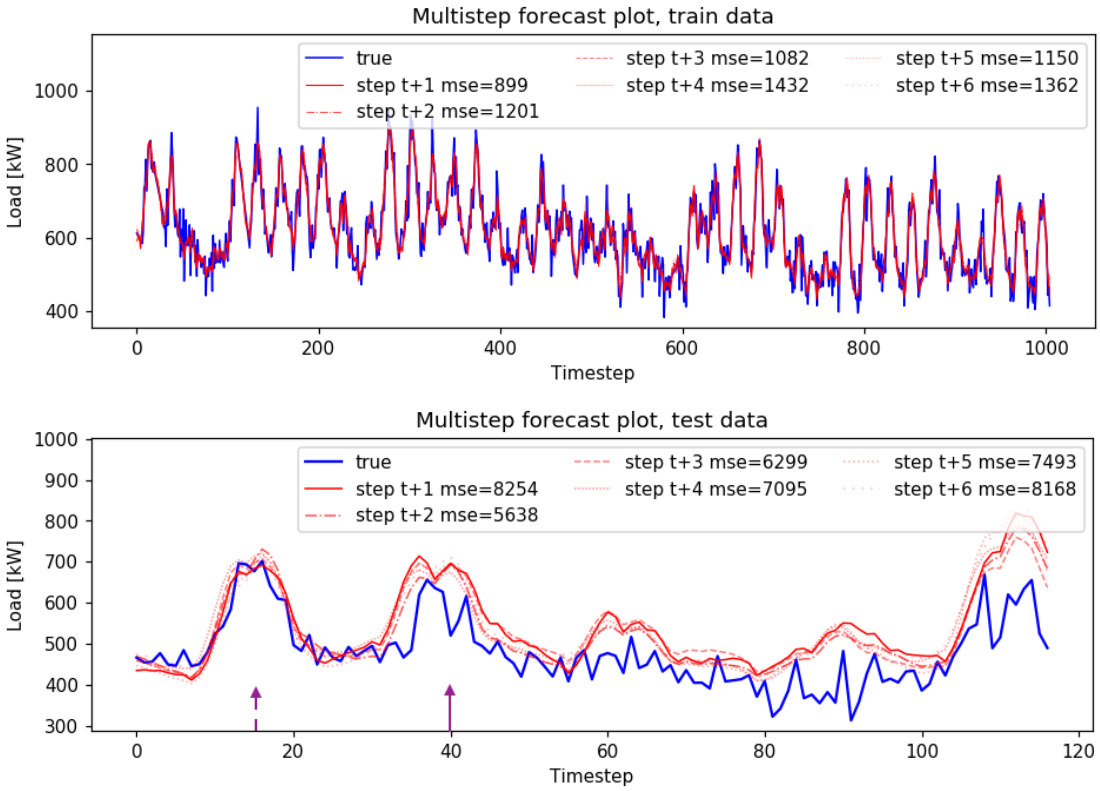


(b) Model *B*. Upper and lower plot is for train and test respectively.

Figure A.9: Multistep forecast plots for *model A* and for *model B*

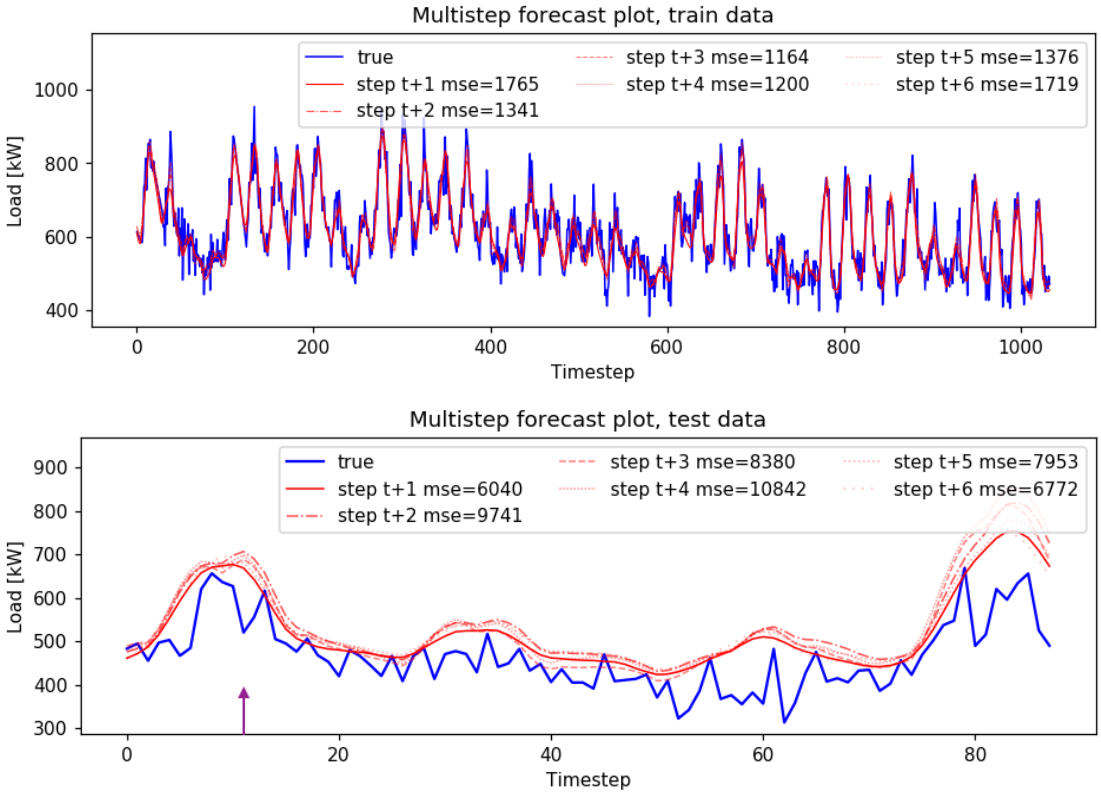


(a) Model *C*. Upper and lower plot is for train and test respectively.

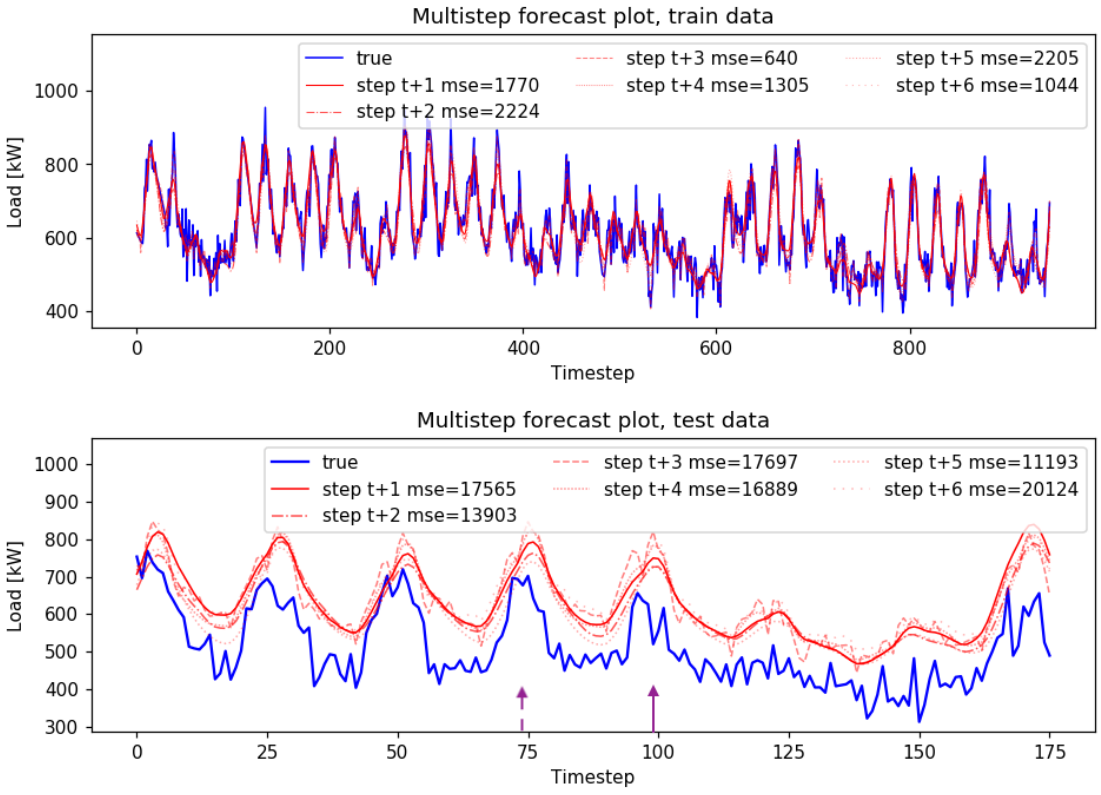


(b) Model *D*. Upper and lower plot is for train and test respectively.

Figure A.10: Multistep forecast plots for *model C* and *model D*.



(a) Model *E*. Upper and lower plot is for train and test respectively.



(b) Model *F*. Upper and lower plot is for train and test respectively.

Figure A.11: Multistep forecast plots for *model E* and *model F*.

Appendix B

Example Python Codes

B.1 RNN model with Keras - Example Code

The following code is the Python script used to make the direct multi-step forecast *model D*. The latter part of the script is used for the making of the forecast plot in **??** and the real-time forecast plots in figure 4.10.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
MODEL_NAME = 'D'  # name of model

#%% IMPORTS
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.preprocessing import  MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error

from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Input, Dense, GRU, LSTM,
    Masking, Dropout
from tensorflow.python.keras.optimizers import RMSprop, SGD, Adam
from tensorflow.python.keras.callbacks import EarlyStopping,
    ModelCheckpoint, TensorBoard, ReduceLROnPlateau
from tensorflow.python.keras.preprocessing.sequence import
    TimeseriesGenerator
np.random.seed(230) #random seed for reproducibility

```

```
20 def mean_absolute_percentage_error(y_true, y_pred):
21     y_true, y_pred = np.array(y_true), np.array(y_pred)
22     return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
23
24 #%% IMPORT DATA
25 df = pd.read_excel('../../data/processed/set04/data.xlsx', index_col
       = 'ValueTime', parse_date=True)
26 df = df[['Maskinrom']]
27 # dealwith missing data
28 df = df.asfreq('30min')
29 df = df.groupby(pd.Grouper(freq='h')).sum(ignore_nan=True)
30 df.columns = ['target']
31 df[df<300.0] = np.nan   # most likely sources of error
32 df.loc['2019-08-20 04:00:00',:] = np.nan
33 df[df==0] = np.nan   #masking value is 0
34 # missing value technique
35 FILL_METHOD = 'interpolate'    # ['0', '-1', 'average', 'interpolate
       ', 'bfill', 'ffill'] # use -1 or 0 for Masking layer
36 if FILL_METHOD in ['0','-1','average','bfill','ffill']:
37     if FILL_METHOD in ['0', '-1', 'average']:
38         fillvalue = int(FILL_METHOD) if FILL_METHOD in ['0', '-1']
              else np.mean(df.values[~np.isnan(df.values)])
39         df.fillna(value = fillvalue, inplace=True)
40     else:
41         df.fillna(method = FILL_METHOD, inplace=True)
42 elif FILL_METHOD in ['interpolate']:
43     df.interpolate('linear', inplace=True)
44 df.plot(fontsize=13)
45 # add potential other features, weather etc. here
46
47 #%% CREATING FEATURES
48 CREATE_TIMEFEATURES = True
49 CREATE_LAGGED_TIMEFEATURES = False  # not needed, as done by
       TimeSeriesGenerator
50 N_LAGSTEPS = 336        #14 days
51 N_FORECASTSTEPS = 6     #steps ahead to forecast
52
53 # Construct the input data, X
54 df_x = df.copy()
55
56 if CREATE_TIMEFEATURES:
57     # explicitly create time-dependent features
58     df_x['week'] = df_x.index.week.astype(str)
59     df_x['dayofweek'] = df_x.index.dayofweek.astype(str)
60     df_x['hour'] = df_x.index.hour.astype(str)
61
62     # create dummys for the time features
63     dummies = pd.get_dummies(df_x[['week', 'dayofweek', 'hour']])
64     df_x = df_x.drop(['week', 'dayofweek', 'hour'], axis = 1)
```

```
    df_x = pd.concat([df, dummies], axis = 1)

if CREATE_LAGGED_TIMEFEATURES:
    # create lagged values
    lagsteps = np.arange(1,N_LAGSTEPS)  #list of lag timesteps to
        construct
    for lag in lagsteps:
        colname = 'target'+str(lag)
        df_x[colname] = df_x[['target']].shift(lag)
    df_x = df_x.dropna()    #remove nans at start an end

# Construct output/target data, y
df_y = df.copy()
lead_steps = np.arange(1, N_FORECASTSTEPS+1)
for lead in lead_steps:
    colname = 'target'+str(lead)
    df_y[colname] = df_y[['target']].shift(-lead)
df_y = df_y.drop('target', axis=1)
#removes last empty rows with nan
df_x = df_x.iloc[:-N_FORECASTSTEPS,:]
df_y = df_y.iloc[:-N_FORECASTSTEPS,:]
x = df_x.values
y = df_y.values
# save some dataset parameters
num_features = x.shape[1]
num_targets = y.shape[1]
num_obs = len(x)
# minmax scaling datasets
x_scaler = MinMaxScaler()
x_scaled = x_scaler.fit_transform(x)
y_scaler = MinMaxScaler()
y_scaled = y_scaler.fit_transform(y)

# train, test and prediction split
TRAIN_SIZE = 0.9
N_LOOKBACK = 24*2 # length of timestep dimension in Keras for
    training batches
BATCH_SIZE_TRAIN = 64 #prøv 64 neste gang. 128
BATCH_SIZE_TEST = 1
num_train = int(num_obs * TRAIN_SIZE)
num_test = num_obs - num_train
#split to scaled test and train, and create validation dataset
x_train_scaled =  x_scaled[0:num_train]
x_val_scaled = x_scaled[num_train:]
x_test_scaled = x_scaled[num_train-N_LOOKBACK:]
y_train_scaled_list = y_scaled[0:num_train]
y_val_scaled_list = y_scaled[num_train:]
y_test_scaled_list = y_scaled[num_train-N_LOOKBACK:]  #for plotting
    in end
```

```
111
112 #%% Define model arcitecture
113 USE_DROPOUTLAYER = True  #må sette meg inn  i først
114 DROPOUT_SHARE = 0.1
115 N_UNITS = 100
116 N_EPOCHS = 70
117 STATEFUL = False
118 ACTIVATION = 'sigmoid' # ['linear', 'sigmoid', 'Relu', etc...]
119
120 def build_model():
121     model = Sequential()
122     model.add(LSTM(units=N_UNITS,
123                    input_shape = (N_LOOKBACK, num_features),
124                    return_sequences=True,
125                    stateful = STATEFUL))
126     model.add(LSTM(units=N_UNITS,
127                    stateful=STATEFUL))
128     if USE_DROPOUTLAYER:
129         model.add(Dropout(DROPOUT_SHARE))
130     model.add(Dense(1, activation=ACTIVATION))
131     return model
132
133 #prepare lists for direct multistep forecast strategy
134 models = [build_model() for _ in range(num_targets)]
135 scores = []
136 train_predictions = []
137 test_predictions = []
138
139 # Start training and prediction for process for model for each
    forecast step
140 for i, model in enumerate(models):
141     print(i)
142     print(model)
143     y_train_scaled = np.reshape(y_train_scaled_list[:,0], newshape=(
        y_train_scaled_list.shape[0],1))
144     y_val_scaled   = np.reshape(y_val_scaled_list[:,0], newshape=(
        y_val_scaled_list.shape[0],1))
145     y_test_scaled  = np.reshape(y_test_scaled_list[:,0], newshape=(
        y_test_scaled_list.shape[0],1))
146
147     #%% define timeseries batch generators for the chosen y's
148     def batch_generator(batch_size, sequence_length):
149         """
150         Generator function for creating random batches of training-
            data.
151         """
152         # Alternative batch generator borrowed from Hvass Labs*
153         while True:
154             # Allocate a new array for the batch of input-signals.
```

```
155             x_shape = (batch_size, sequence_length, num_features)
156             x_batch = np.zeros(shape=x_shape, dtype=np.float16)
157             # Allocate a new array for the batch of output-signals.
158             y_shape = (batch_size, sequence_length, 1)
159             y_batch = np.zeros(shape=y_shape, dtype=np.float16)
160             # Fill the batch with random sequences of data.
161             for i in range(batch_size):
162                 # Get a random start-index.
163                 # This points somewhere into the training-data.
164                 idx = np.random.randint(num_train - sequence_length)
165                 # Copy the sequences of data starting at this index.
166                 x_batch[i] = x_train_scaled[idx:idx+sequence_length]
167                 y_batch[i] = y_train_scaled[idx:idx+sequence_length]
168             yield (x_batch, y_batch)
169 
170     generator = batch_generator(batch_size=BATCH_SIZE_TRAIN,
        sequence_length=N_LOOKBACK)
171 
172     # Generators provided from Keras - TimeseriesGenerator
173     train_data_gen_shuffle = TimeseriesGenerator(x_train_scaled,
        y_train_scaled,
174         length=N_LOOKBACK, sampling_rate=1,stride=1,
175         batch_size=BATCH_SIZE_TRAIN, shuffle = True)
176     train_data_gen = TimeseriesGenerator(x_train_scaled,
        y_train_scaled,
177         length=N_LOOKBACK, sampling_rate=1,stride=1,
178         batch_size=BATCH_SIZE_TRAIN, shuffle = False)
179     val_data_gen = TimeseriesGenerator(x_val_scaled, y_val_scaled,
180         length=N_LOOKBACK, sampling_rate=1,stride=1,
181         batch_size=BATCH_SIZE_TEST, shuffle=True)
182     test_data_gen = TimeseriesGenerator(x_test_scaled, y_test_scaled
        ,
183         length=N_LOOKBACK, sampling_rate=1,stride=1,
184         batch_size=BATCH_SIZE_TEST)
185 
186 #test the data generators
187 if False:
188     x_batch, y_batch = next(generator)
189     print(x_batch.shape)
190     print(y_batch.shape)
191     x_batch, y_batch = train_data_gen[0]
192     print(x_batch.shape)
193     print(y_batch.shape)
194     x_batch, y_batch = test_data_gen[0]
195     print(x_batch.shape)
196     print(y_batch.shape)
197 
198     #%% define all callbacks for model imoprovance
199     USE_EARLYSTOPPING = True
```

```
200     PATIENCE = 20
201     USE_REDUCELR = True
202
203     path_checkpoint = '24_checkpoint.keras' #save checkpoint
204     #monitor that saves the latest best model regards to validation
            loss
205     callback_checkpoint = ModelCheckpoint(filepath=path_checkpoint,
206                                           monitor='val_loss',
207                                           verbose=1,
208                                           save_weights_only=True,
209                                           save_best_only=True)
210     #early stopping will end an epoch/training if validation
211     # does not improve for PATIENCE amount of steps/epochs.
212     callback_early_stopping = EarlyStopping(monitor='val_loss',
213                                             patience=PATIENCE,
                                                    verbose=1)
214     #Mysterious module for feedback
215     callback_tensorboard = TensorBoard(log_dir='./24_logs/',
216                                        histogram_freq=0,
217                                        write_graph=False)
218     #reduces learning rate to appropriate number for impr. learning
219     callback_reduce_lr = ReduceLROnPlateau(monitor='val_loss',
220                                            factor=0.1,
221                                            min_lr=1e-4,
222                                            patience=2,
223                                            verbose=1)
224     callbacks = [] #collecting all callbacks in a list
225     if USE_EARLYSTOPPING: #add early stopping if chosen
226         callbacks.append(callback_early_stopping)
227     callbacks.append(callback_checkpoint)
228     #callbacks.append(callback_tensorboard)
229     if USE_REDUCELR: # add reduceLR if chosen
230         callbacks.append(callback_reduce_lr)
231
232     # Compile the model
233     model.compile(loss='mse',
234                   metrics = ['mae','mse'],
235                   optimizer='adam')
236
237     # Train the model with train set generator, while vaildating
            against validation data
238     history = model.fit_generator(train_data_gen_shuffle,
239                                   epochs=N_EPOCHS,
240                                   steps_per_epoch = 100,
241                                   use_multiprocessing=False,
242                                   callbacks=callbacks,
243                                   validation_data=val_data_gen,
244                                   verbose=1).history
245     #reload the best model from PATIENCE amount of epochs earlier
```

```
246     try:
247         model.load_weights(path_checkpoint)
248     except Exception as error:
249         print("Error trying to load checkpoint.")
250         print(error)
251 
252     #%% create and save loss plots
253     ax = pd.DataFrame(history)[['loss', 'val_loss']].plot(logy=True,
            figsize=(10,5))
254     fig = ax.get_figure()
255     fig.savefig('direct'+ str(MODEL_NAME) + str(i) + '_losscurve1.
            png')
256     ax = pd.DataFrame(history)[['mean_squared_error', '
            val_mean_squared_error']].plot(figsize=(10,5))
257     fig = ax.get_figure()
258     fig.savefig('direct'+ str(MODEL_NAME) + str(i) + '_losscurve1.
            png')
259 
260     #%% after training, evaluate and do a final forecast
261     score = model.evaluate_generator(test_data_gen)
262     scores.append(score)
263     trainPredict = model.predict_generator(train_data_gen)
264     testPredict=model.predict_generator(test_data_gen)
265 
266     train_predictions.append(trainPredict)
267     test_predictions.append(testPredict)
268 
269     #checkpoint save of current prediction arrays as .npy files
270     np.save(str(MODEL_NAME)+'_test_predictions', test_predictions)
271     np.save(str(MODEL_NAME)+'_train_predictions', train_predictions)
272     np.save(str(MODEL_NAME)+'_scores', scores)
273 
274 #%% After finished training, create plots and calculate mse
275 test_predictions = np.array(test_predictions)
276 train_predictions = np.array(train_predictions)
277 
278 # choice for loading externally stored predictions (.npy-files)
279 if False:
280     test_predictions = np.load('D_test_predictions.npy')
281     train_predictions = np.load('D_train_predictions.npy')
282 
283 #%% Plotting tools
284 linestyle_tuple = [
285         '-','-.','--',
286      (0, (1, 1)),
287      ':',
288      (0, (1, 4)),
289      (0, (5, 10)),
290      (0, (5, 5)),
```

```
      (0, (5, 1)),
      (0, (3, 10, 1, 10)),
      (0, (3, 5, 1, 5)),
      (0, (3, 1, 1, 1)),
      (0, (3, 5, 1, 5, 1, 5)),
      (0, (3, 10, 1, 10, 1, 10)),
      (0, (3, 1, 1, 1, 1, 1))]
#inverse transform forecast results
trainPred = y_scaler.inverse_transform(np.array(train_predictions).
    squeeze().T)
testPred = y_scaler.inverse_transform(np.array(test_predictions).
    squeeze().T)
# get the true values
trainTrue = y[0:num_train,0][-len(trainPred):]
testTrue = y[num_train:,0][-len(trainPred):]
# Changeable design parameters
figsize = (10,3)
dpi=110

plt.figure(figsize=figsize, dpi=dpi)
plt.plot(trainTrue, color = 'blue',label='true', linewidth=1)
for i, vec in enumerate(trainPred.T):
    alph=1/((i/2)+1)
    plt.plot(np.arange(i, len(vec)+i), vec, label='step t+'+str(i+1)
        + ' mse=' + str(int(mean_squared_error(vec, trainTrue))),
            color='red', ls=linestyle_tuple[i], alpha = alph,
                linewidth=.7)
plt.title('Multistep forecast plot, train data')
plt.xlabel('Timestep')
plt.ylabel('Load [kW]')
plt.ylim(None, max(trainTrue)+200)
plt.legend(loc=1, ncol=3, fancybox=True)
plt.show()


plt.figure(figsize=figsize, dpi=dpi)
plt.plot(testTrue, color = 'blue',label='true')
for i, vec in enumerate(testPred.T):
    alph=1/((i/2)+1)
    plt.plot(np.arange(i, len(vec)+i), vec, label='step t+'+str(i+1)
        + ' mse=' + str(int(mean_squared_error(vec, testTrue))),
            color='red', ls=linestyle_tuple[i], alpha=alph,
                linewidth=1)
plt.title('Multistep forecast plot, test data')
plt.xlabel('Timestep')
plt.ylabel('Load [kW]')
plt.ylim(None, max(testTrue)+300)
plt.legend(loc=1, ncol=3, fancybox=True)
plt.show()
```

```
334
335 #%% STEP-WISE REAL-TIME FORECAST ANIMATION, step by step
336 if False:
337     import time
338     beginning = 0
339     step = beginning
340     figsize = (5,3)
341     dpi=110
342     PLOT_TEST = True #choose to animate train or test set forecasts
343
344     if PLOT_TEST:
345         while True:
346             xaxis = np.arange(0, len(testPred))
347             tmp_pred_array=[testPred[step+i,i] for i in range(
                    testPred.shape[1])]
348             plt.figure(figsize=figsize, dpi=dpi)
349             plt.plot(xaxis[:step+1], testTrue[:step+1], label='True
                    historic')
350             plt.plot(xaxis[step:step+num_targets+5], testTrue[step:
                    step+num_targets+5], alpha = 0.5, label='True
                    baseline')
351             plt.plot(np.arange(step, step+num_targets),
                    tmp_pred_array, 'r.-', label='Forecasted baseline')
352
353             for i, vec in enumerate(testPred.T):
354                 alph=0.5
355                 plt.plot(np.arange(beginning+i+1, step+i+1), vec[
                        beginning+i+1:step+i+1], label='step t+'+str(i+
                        1),
356                          color='red', ls=linestyle_tuple[1+i], alpha
                              = alph)
357
358             plt.xlim((step-15, step+num_targets+5))
359             plt.title('Forecasted VS True values - test set \
                    nTimestep: '+str(step-1))
360             plt.xlabel('Timestep')
361             plt.ylabel('Load [kW]')
362             plt.legend(loc='center left')
363             plt.show()
364             time.sleep(0.1)
365             step+=1
366
367     if not PLOT_TEST:
368         import time
369         while True:
370             xaxis = np.arange(0, len(trainPred))
371             tmp_pred_array=[trainPred[step+i,i] for i in range(
                    trainPred.shape[1])]
372             plt.figure(figsize=figsize, dpi=dpi)
```

```
373             plt.plot(xaxis[:step+1], trainTrue[:step+1], label='True
                    historic')
374             plt.plot(xaxis[step:step+num_targets+5], trainTrue[step:
                    step+num_targets+5], alpha = 0.5, label='True
                    baseline')
375             plt.plot(np.arange(step, step+num_targets),
                    tmp_pred_array, 'r.-', label='Forecasted baseline')
376
377             for i, vec in enumerate(trainPred.T):
378                 alph=0.5
379                 plt.plot(np.arange(beginning+i+1, step+i+1), vec[
                        beginning+i+1:step+i+1], label='step t+'+str(i+
                        1),
380                          color='red', ls=linestyle_tuple[1+i], alpha
                              = alph)
381
382             plt.xlim((step-15, step+num_targets+50))
383             plt.title('Forecasted VS True values - train set \
                    nTimestep: '+str(step-1))
384             plt.xlabel('Timestep')
385             plt.ylabel('Load [kW]')
386             plt.legend(loc='center left')
387             plt.show()
388             time.sleep(0.1)
389             step+=1
```

B.2 Asset class in Python

A modelling framework flexibility estimation and making flexplots

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt #plot tool
import seaborn as sns           #styles for plots
import numpy as np              #Numpy arrays
sns.set_style('darkgrid')       # set plot style
class Asset():
    """ A class used to model a flexible asset, estimate its
        flexible power and create a flexplot. """
    ### ----- Initializing function ----- ###
    def __init__(self, name, power_min, power_max, power_baseline,
        power_steadystate, dt = 1, max_runtime = None):
        """
        Parameters
        ----------
        name : str
            The name of the asset
        power_min and power_max : array of lenght H
            Sets the minimum respectively maximum allowed  power
               consumption for the asset
        power_baseline : array of lenght H
            Represents the baseline power consumption (e.g. a
               forecast).
        power_steadystate : array of lenght H
            The electrical consumptipn that would lead to a steady
               state situation (no charging of asset/no change in
               energy/SoC level)
        dt : float
            Temporal resolution of the arrays. Indicate time between
                each element, unit in hours.
        max_runtime : float
            A maximum runtime for the flexibility source that should
                constrain the length of estimated flexibility (
               default=None).
        NOTES:
        Each element in the arrays represent a value for consecutive
            timesteps. Negative values means that the asset
           produces power. Unit: kWh/h
        """
        #Assign input attributes
        self.Name = name
```

```
31          ##Zero is inserted to arrays because of later calculation
                convenience
32          self.Power_max = np.hstack([[0], np.array(power_max)])
33          self.Power_min = np.hstack([[0], np.array(power_min)])
34          self.Power_baseline = np.hstack([[0], np.array(
                power_baseline)])
35          self.Power_ss = np.hstack([[0],np.array(power_steadystate)])
36          self.dt = dt
37          self.max_runtime = max_runtime
38
39          #Assign calculated attributes
40          self.Power_charge = self.Power_baseline - self.Power_ss
41          self.Fneg = - self.Power_baseline + self.Power_min
42          self.Fpos = - self.Power_baseline + self.Power_max
43          self.timelineE = np.arange(0,len(power_baseline)*dt +dt, dt)
44          self.timelineF = self.timelineE - dt/2
45          self.__energystorage = False
46
47      ### ----- Methods ----- ###
48      def add_energy_storage(self, initial_SoC, E_delta=None, E_min=
            None, E_max=None):
49          """
50          Assigns an energy storage to the asset.
51
52          Parameters
53          ----------
54          initial_SoC : str, optional
55              The sound the animal makes (default is None)
56          E_delta : str, optional
57               Range of allowed energy levels in the storage, units:
                     kWh
58          E_min : str, optional
59              The minimum allowed energy level for the energy storage,
                     units: kWh
60          E_max : str, optional
61              The maximum allowed energy level for the energy storage,
                     units: kWh
62          NB: 2 of either E_delta (scale), E_min and E_max must be
                given.
63
64          Raises
65          ------
66          ValueError
67              If not 2 of the parameters E_delta (scale), E_min and
                    E_max
68              are given.
69          """
70          if initial_SoC < 0 or initial_SoC > 1:
71              print('NB!: SoC is out of allowed limits')
```

```
72          if E_delta and E_min!=None and not E_max:
73              self.E_delta = E_delta
74              self.E_min   = E_min
75              self.E_max   = E_min + E_delta
76          elif E_delta and not E_min!=None and E_max:
77              self.E_delta = E_delta
78              self.E_min   = E_max - E_delta
79              self.E_max   = E_max
80          elif not E_delta and E_min!=None and E_max:
81              self.E_delta = E_max - E_min
82              self.E_min   = E_min
83              self.E_max   = E_max
84          else:
85              raise ValueError('You must assign precisely 2 of the 3
                    parameters E_delta, E_min and E_max.')
86          self.SoC_state = initial_SoC
87          self.__energystorage = True
88          self.E_state = float(self.E_min + (self.SoC_state * self.
                E_delta))
89
90      def __apply_storage_constraints(self):
91          """
92          Calculates the energy trajectories and constrains the
                flexibilityestimates. Trajectories for different choices
                 of the consumption is calcualted, in the case of
                choosing baseline, max pos flex and max neg. flex
                activation. This method also updates the flexibility
                estimates constrained by exceeding the energy storage
                capacity limits.
93          """
94          #integration weights used for summing and visualizing energy
                 trajectories
95          integration_weights = np.full(len(self.Power_baseline), self
                .dt)
96          integration_weights[0] = 0
97
98          #calculate energy storage level trajectories for baseline
                and flex activation
99          self.E_traj_pos = self.E_state + np.add.accumulate(self.
                Power_charge+self.Fpos) * integration_weights
100         self.E_traj_neg = self.E_state - np.add.accumulate(-self.
                Power_min + self.Power_ss) * integration_weights
101         self.E_traj_baseline = self.E_state + np.add.accumulate(self
                .Power_charge) * integration_weights
102
103         # constrain E trajectories that overshoot the energy storage
                 limits.
104         # save the original and set overshooting values equal to the
                 limit.
```

```
105         self.E_traj_pos_overshoot = self.E_traj_pos.copy()
106         self.E_traj_neg_overshoot = self.E_traj_neg.copy()
107 
108         index_pos_overshooting = np.where(self.E_traj_pos > self.
                E_max)[0]
109         index_neg_overshooting = np.where(self.E_traj_neg < self.
                E_min)[0]
110         self.E_traj_pos[index_pos_overshooting] = self.E_max
111         self.E_traj_neg[index_neg_overshooting] = self.E_min
112 
113         # Constrain parts of Fpos/Fneg that led to overshooting
                energy storage limits
114         self.Fpos[index_pos_overshooting] = +(self.E_traj_pos[
                index_pos_overshooting] - self.E_traj_pos[
                index_pos_overshooting-1])/self.dt - self.Power_charge[
                index_pos_overshooting]
115         self.Fneg[index_neg_overshooting] = +(self.E_traj_neg[
                index_neg_overshooting] - self.E_traj_neg[
                index_neg_overshooting-1])/self.dt - self.Power_charge[
                index_neg_overshooting]
116         self.Fpos[self.Fpos<0] = 0
117         self.Fneg[self.Fneg>0] = 0
118 
119     def __apply_time_constraints(self):
120         self.Fpos[self.max_runtime+1:] = 0
121         self.Fneg[self.max_runtime+1:] = 0
122 
123     def make_flexplot(self):
124         """ Vizualization of flexibility estimate results,
                consistent of four subplots. """
125         if self.max_runtime:
126             self.__apply_time_constraints()
127         ### Create figure
128         if self.__energystorage:
129             fig, [ax2,ax3,ax,ax1] = plt.subplots(4, 1, figsize=(6*
                    0.95,8*0.95), dpi=100, sharex=True, gridspec_kw = {'
                    height_ratios':[1,1,3,1]})
130             ax2.set_title('Asset: ' + str(self.Name) + ' | Current
                    SoC: ' + str(self.SoC_state) + '\n\n' + 'Forecasted
                    baseline/load schedule')
131 
132             #create the line arrays for the max and min energy
                    storage capacity
133             self.__apply_storage_constraints()
134             E_max_curve = np.full(len(self.timelineE), self.E_max)
135             E_min_curve = np.full(len(self.timelineE), self.E_min)
136         elif not self.__energystorage:
137             fig, [ax2,ax1] = plt.subplots(2, 1, figsize=(6*0.95,3*
                    0.95), dpi=100, sharex=True, gridspec_kw = {'
```

```
                height_ratios':[1,1]})
138             ax2.set_title('Asset: ' + str(self.Name) + '\n\n' + '
                    Forecasted baseline/load schedule')
139
140         if self.__energystorage:
141
142             ## PLOT TWO - PLANNED CHARGING OF STORAGE
143             ax3.step(self.timelineF[1:], self.Power_charge[1:],
                    where='mid', color='b', label='$\hat{P}_{(dis)charge
                    }$')
144
145             #max and min power
146             ax3.step(self.timelineF[1:], self.Power_charge[1:] +
                    self.Fpos[1:], where='mid', label='$P_{(dis)charge,
                    max}$', color='.5', ls='--')
147             ax3.step(self.timelineF[1:], self.Power_charge[1:] +
                    self.Fneg[1:], where='mid', label='$P_{(dis)charge,
                    min}$', color='.5', ls='-.')
148             ax3.set_ylabel('Planned storage  \ncharging [kWh/h]')
149             ax3.set_title('Forecasted plan of energy storage
                    charging')
150             ax3.legend(loc = 'center left', bbox_to_anchor=(1.01,
                    0.5),ncol=1)
151
152             ## THIRD PLOT
153             # draw max and min energy storage limit lines
154             ax.set_title('Energy storage trajectories for baseline
                    and flex. options')
155             ax.plot(self.timelineE, E_max_curve,label='$E_{max}$',
                    color='k', ls='-.', marker='')
156             # positive
157             if self.E_max > self.E_state:
158                 ax.plot(self.timelineE, self.E_traj_pos, color='g',
                        ls='-', marker='^', label='$E(t, F_{p}(t))$')
159                 ax.plot(self.timelineE, self.E_traj_pos_overshoot,
                        color='0', ls=':', alpha=.3)
160             # planned route
161             ax.plot(self.timelineE, self.E_traj_baseline, color='b',
                     ls='-', label = '$E_{plan}(t)$')
162             # negative
163             if self.E_min < self.E_state:
164                 ax.plot(self.timelineE, self.E_traj_neg, color='r',
                        ls='-', marker='v', label='$E(t, F_{n}(t))$')
165                 ax.plot(self.timelineE, self.E_traj_neg_overshoot,
                        color='0', ls=':', alpha=0.3)
166             ax.plot(self.timelineE, E_min_curve,label='$E_{min}$',
                    color='k', ls='--', marker='')
167
168             # fill space with color
```

```
169             ax.fill_between(self.timelineE,self.E_traj_pos, self.
                    E_traj_baseline,
170                             color='green', hatch='////', alpha=.25,
                                label='E space for \npositive flex.'
                                )
171             ax.fill_between(self.timelineE, self.E_traj_neg, self.
                    E_traj_baseline,
172                             color='red', hatch="\\\\\\\\", alpha=
                                .25, label='E space for \nnegative
                                flex.')
173             # details
174             ax.set_ylabel('Stored energy\n[kWh]')
175             ax.set_ylim((self.E_min - self.E_delta/5, self.E_max+
                    self.E_delta/5))
176             ax.legend(loc = 'center left', bbox_to_anchor=(1.01,
                    0.5),ncol=1)
177
178         ## FIRST PLOT -  PLAN/SCHEDULE
179         #forecasted baseline/plan/scheduled consumption
180         ax2.step(self.timelineF[1:], self.Power_baseline[1:],where='
                mid',  label='$\hat{P}_{baseline}$', color='darkorange',
                 ls='-', marker='.')
181         #max and min power
182         ax2.step(self.timelineF[1:], self.Power_max[1:], where='mid'
                , label='$P_{max}$', color='.5', ls='--')
183         ax2.step(self.timelineF[1:], self.Power_min[1:], where='mid'
                , label='$P_{min}$', color='.5', ls='-.')
184         ax2.set_ylabel('Forecasted \nBaseline [kWh/h]')
185         ax2.legend(loc = 'center left', bbox_to_anchor=(1.01, 0.5),
                ncol=1)
186         ax2.set_xlim((0, self.timelineE[-1]))
187         ax2.set_xticks(self.timelineE)
188
189         ## FOURTH PLOT - FLEXIBILITY
190         # plot bar charts
191         ax1.bar(self.timelineF, self.Fpos, width=self.dt, alpha=.5,
                color='c', hatch='||', label='$F_{p}(t)$')
192         ax1.bar(self.timelineF, self.Fneg, width=self.dt, alpha=.5,
                color='m', hatch='' ,label='$F_{n}(t)$')
193         # put values on the bar chart
194         for i in range(1, len(self.Fpos)):
195             if sum(self.Fpos)>0:
196                 ax1.text(self.timelineF[i]-.04, max(self.Fpos)*.2,
                        str(self.Fpos[i]), fontsize=11, fontstyle='
                        normal')
197             if sum(self.Fneg)<0:
198                 ax1.text(self.timelineF[i]-.04, min(self.Fneg)*.4,
                        str(self.Fneg[i]), fontsize=11)
199         # details
```

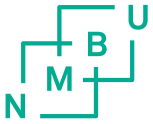
```
200         ax1.set_title('Estimated available flexibility')
201         ax1.legend(loc = 'center left', bbox_to_anchor=(1.01, 0.5),
                ncol=1)
202         ax1.set_ylabel('Flexible power \n[kWh/h]')
203         ax1.set_xlabel('Forecast timestep [h]')
204 
205         fig.show()
206 #%%
207 if __name__ == "__main__":
208     # EXAMPLE 1
209     ## Battery, passive schedule, constant planned SoC level
210     battery1 = Asset('Battery, passive',[-30]*6,[30]*6, [0]*6, [0]*
            6, dt =0.25)
211     battery1.add_energy_storage( 0.5, E_max = 65, E_min = 65*0.0)
212     battery1.make_flexplot()
213 
214     # EXAMPLE 2
215     ## Battery, active schedile, charging then discharging
216     battery2 = Asset('Battery, active',[-30]*6,[30]*6, [30,30,30,-
            30,-30,-30], [0]*6, dt =0.25)
217     battery2.add_energy_storage(0.5, E_max = 65, E_min = 65*0.0)
218     battery2.make_flexplot()
219 
220     # EXAMPLE 3
221     ## Water heater with alternative energy source
222     max_runtime = 4
223     wbaseline = np.array([300, 280, 270, 290, 340, 310])
224     waterheater1 = Asset('Waterheater', [0]*len(wbaseline),
            wbaseline, wbaseline, wbaseline, 1, max_runtime=max_runtime)
225     waterheater1.make_flexplot()
226 
227     # EXAMPLE 4
228     ## Machine room
229     machineroom1 = Asset('machineroom',
230     [538,540,523,501,469,420,378,350],
231     [838,840,823,801,769,720,678,650],
232     [676,701,641,609,606,496,482,521],
233     [676,701,641,609,606,496,482,521], dt =1)
234     machineroom1.add_energy_storage( 0.6, E_max = 9000, E_delta =
            1000)
235     machineroom1.make_flexplot()
```

B.3 Python Code for creating the flexplots in machine room use-case

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# CREATING THE FLEXPLOTS IN MACHINE ROOM USE-CASE

import numpy as np
from Asset import Asset # Import Asset class for usage

# Constructing baseline matrixes for the forecasts and true realized values*
# drom 'RNN multistep forecast result' table presented in the master thesis
bl_pred=np.array([607, 638, 692, 678, 710, 730])
bl_real=np.array([574, 668, 737 ,673, 734, 768])

bl_pred=np.array([[513, 596, 673, 704, 663, 681],
                  [578, 650, 693, 698, 641, 697],
                  [647, 671, 683, 713, 715, 684],
                  [0,0,0,0,0,0],
                  [0,0,0,0,0,0],
                  [0,0,0,0,0,0],
                  [692,718, 655, 598, 532, 493]])

[524, 542, 584, 697, 694, 677, 702, 642, 610, 606, 497, 482]

bl_real=np.array([[524, 542, 584, 697, 694, 677],
                  [542, 584, 697, 694, 677, 702],
                  [584, 697, 694, 677, 702, 642],
                  [0,0,0,0,0,0],
                  [0,0,0,0,0,0],
                  [0,0,0,0,0,0],
                  [702, 642, 610, 606, 497, 482]])

# constants for the power the asset can deviate with from the baseline.
plusmax = 50     #kWh/h
minusmax = -80   #kWh/h

#EVENT (a)
start=0
machineroom_stepA = Asset('machineroom (a)', bl_pred[start]+minusmax, bl_pred[start]+plusmax,
                          bl_pred[start], bl_pred[start], dt=1)
machineroom_stepA.add_energy_storage( 0.6, E_max = 8000, E_delta = 600)
```

```
machineroom_stepA.make_flexplot()
#EVENT (b)
start=1
machineroom_stepB = Asset('machineroom (b)', bl_pred[start]+minusmax, bl_pred[start]+plusmax,
                          bl_pred[start], bl_pred[start], dt=1)
machineroom_stepB.add_energy_storage( 0.6, E_max = 8000, E_delta = 600)
machineroom_stepB.make_flexplot()
#EVENT (c)
start=2
machineroom_stepC = Asset('machineroom (c)', bl_pred[start]+minusmax, bl_pred[start]+plusmax,
                         bl_pred[start], bl_pred[start], dt=1)
machineroom_stepC.add_energy_storage( 0.6, E_max = 8000, E_delta = 600)
machineroom_stepC.make_flexplot()
#EVENT (d) - activateD the four first negative bid slots.
#Anticipated decrease in SoC level: 80*4 /600 = 0.5333
#New SoC = 0.6- 0.5333 = 0.06667
start=6
machineroom_stepC = Asset('machineroom (d)', bl_pred[start]+minusmax, bl_pred[start]+plusmax,
                          bl_pred[start], bl_pred[start], dt=1)
machineroom_stepC.add_energy_storage( 0.067777, E_max = 8000, E_delta = 600)
machineroom_stepC.make_flexplot()

# Delivered flex, based on the assumptions on dispatch ability, would be
F_delivered = bl_real[2][:4]-bl_pred[2][:4] -80
# and the real delivered flexible ENERGY would be
E_delivered = np.sum(F_delivered)

#The error of delivered flexibility is
error = [-80]*4 - F_delivered
print(error)
# result: [ 63 -26 -11  36]
```

Thank you.



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

Postboks 5003
NO-1432 Ås
Norway