Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electric Power Engineering

Marius Bagle

Investigation into the impact of thermal energy flexibility on cost optimal design and operation of Zero Emission Buildings

Master's thesis in Energy and Environmental Engineering
Supervisor: Karen Byskov Lindberg
June 2019

NTNU
Norwegian University of
Science and Technology

Marius Bagle

Investigation into the impact of thermal energy flexibility on cost optimal design and operation of Zero Emission Buildings

Master's thesis in Energy and Environmental Engineering
Supervisor: Karen Byskov Lindberg
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electric Power Engineering

NTNU
Norwegian University of
Science and Technology

Abstract

Zero Emission Buildings (ZEBs) are energy efficient buildings that produce on-site renewable energy, in order to compensate for their consumption. The ZEB-concept is based on the 2010 report by EU's Energy Performance of Buildings Directive (EPBD), which suggests that all buildings constructed after 2020 should reach "near zero energy level" [1]. In previous research on energy systems in ZEBs, deterministic linear optimization techniques, in conjunction with a wide array of input data, such as load data, temperatures and technology prices has been used to determine the cost-optimal design of technology investments in low energy buildings. Usually, the heat demand of the buildings considered has been treated as an aggregated load.

The main purpose of this thesis is further development of a Mixed Integer Linear Program (MILP), implemented in the open-source general purpose programming language Python, using the modelling extension library Pyomo. The starting point of the work was the two-stage stochastic model developed in [2], transitioning back to a deterministic framework. At first, the separation of the heat demand into two separate loads is carried out, one for space heating and one for domestic hot water. Then, a model based on point-source technologies is synthesized. The first of two main objectives is to analyze and compare the operation and investment of the point-source model and the already existing waterborne model, both with and without the ZEB-constraint. The emission constraints are defined in such a way as to consider the emissions in the operational phase of the building, an ambition level known as "ZEB-O EQ" [3]. The input data used for the optimization is based on simulated load data of the heat and electricity demand, developed in [4], [5] and [6]. Data from 2012, considered to be an average climatic year [4], is used. Since the separation of the heat demand into two different loads causes a drastic increase in the number of variables, a simple reduction technique, selecting the week with the highest space heating load from each season, is used to construct a reduced scenario.

The second main objective of the thesis is to investigate the load flexibility of the ZEB, using the thermal mass of the building as a short-term thermal energy storage. A two-node model representing the thermal mass of the building is implemented, in both the point-source and the waterborne model. Then, the impact of adding this storage is analyzed and compared for the respective systems. Since there is some uncertainty associated with the parameters of the two-node model, a sensitivity analysis is performed, in order to determine both the suitability of the two-node representation in a MILP-framework, and also to find a range of values for the cost reduction that can be expected when using the building thermal mass as an energy storage.

The results show that the waterborne system is the cost-optimal choice for the energy system in a passive house, both with and without emission constraints. A significant part of its advantage lies in the greater efficiency of the waterborne heat pumps, in addition the flexibility inherent in the waterborne system, since the technologies can operate on both the SH- and DHW-load. Furthermore, the grid impact of the waterborne system is more favorable, as the duration curve for total electricity import is significantly flatter than for the point-source system. When adding the building thermal mass as a storage technology, a reduction in peak load capacity can be seen for both systems, which suggests that the thermal mass can be used as a substitute for the peak load technologies, e.g. the electric boiler, in passive house energy systems. Furthermore, significant decreases in the net present value of both the total system cost and operational cost can be seen. The most promising cases were found when both systems were forced to obey the ZEB-constraint with the thermal mass as a storage technology, showing reductions in operational costs of 8.60 % and 7.79 % (compared to no thermal mass/no-BITES) for the point-source and waterborne systems, respectively. Additionally, a similar reduction in total electricity import was seen in these two cases, suggesting that the the on-site production from the photo-voltaic panels are used to pre-heat building for the evening, when spot prices generally are higher. The sensitivity analysis shows that the thermal mass representation used exhibits a relatively small sensitivity to its parameters. The values considered, which in the most extreme case varied by five orders of magnitude, yielded a range for the total cost reduction of between ca. 1300 € and 2500 € through the lifetime of the building.

Sammendrag

Zero Emission Buildings (ZEB) er energieffektive bygninger som produserer fornybar energi ”on-site”, for å kompensere for forbruk. ZEB-konseptet er basert på rapporten fra EUs direktiv om energieffektivitet av bygninger (EPBD) utgitt i 2010, som foreslår at alle bygninger bygget etter 2020 skal nå ”near zero energy level” [1]. I tidligere forskning på energisystemer i ZEBs har deterministiske lineære optimaliseringsteknikker, sammen med et bredt spekter av data, som lastdata, temperaturer og teknologipriser blitt brukt til å bestemme den kostnadsoptimale utformingen av energissystemer i lavenergibygninger. Vanligvis har varmelasten i bygningene blitt behandlet som en aggregert last.

Hovedformålet med denne oppgaven er videreutviklingen av et Mixed Integer Linear Program (MILP), implementert i det generelle programmeringsspråket Python, ved hjelp av modellbyggingsbiblioteket Pyomo. Utgangspunktet for arbeidet var den to-trinns stokastiske modellen utviklet i [2], transformert tilbake til et deterministisk rammeverk. Først ble separasjonen av varmelasten i to separate komponenter utført, en for romoppvarming og en for varmtvann. Deretter syntetiseres en modell basert på punktvarmekilder. Det første av to hovedmål er å analysere og sammenligne drift og investering av punktkildemodellen og den allerede eksisterende vannbårne modellen, både med og uten ZEB-begrensningen. Emissionsbegrensningene er definert på en måte som kun tar hensyn til utslippene i driftsfasen av bygningen, et ambisjonsnivå kjent som ”ZEB-O EQ” [3]. Dataene som brukes for optimaliseringen er basert på simulerte lastdata for varme- og strømforbruket, utviklet i [4], [5] og [6]. Data fra 2012, regnet som et gjennomsnittlig klimaår [4], brukes. Siden separasjonen av varmetilførselen i to forskjellige komponenter for hver teknologi fører til en drastisk økning i antall variabler, brukes en enkel reduksjonsteknikk, som velger uken som inneholder tidssteget med det høyeste romoppvarmingsbehovet fra hver sesong til å konstruere et redusert scenario.

Det andre hovedformålet med oppgaven er å undersøke lastfleksibilitet i ZEBs ved å bruke bygningens termiske masse som et kortsiktig energilager. En to-node modell som representerer bygningens termiske masse er implementert, både i punktvarmekildesystemet og i det vannbårne systemet. Så blir virkningen av å legge til dette lageret analysert og sammenlignet for de respektive systemene. Siden det er noe usikkerhet knyttet til parametrene til i to-node modellen, utføres en sensitivitetsanalyse for å undersøke både to-node modellens egnethet i et MILP-rammeverk, og å finne et spekter av verdier for kostnadsreduksjonen som kan forventes ved bruk av bygningens termiske masse som energilager.

Resultatene viser at det vannbaserte systemet er det kostnadsoptimale valget for energisystemet i et lavenergihus, både med og uten utslippskrav. En betydelig del av denne fordelen ligger i de vannbårne varmepumpenes effektivitet, i tillegg til fleksibiliteten i det vannbårne systemet, siden teknologiene kan operere på både romoppvarmings- og varmtvannslasten. Videre er påvirkningen på kraftnettet av det vannbårne systemet mer gunstig, da varighetskurven for total import av elektrisitet er betydelig flatere enn for punktvarmesystemet. Når man legger til bygningens termiske masse som energilager, kan man se en reduksjon i topplastkapasiteten for begge systemer, noe som tyder på at termisk masse kan brukes som erstatning for topplastteknologier i lavenergibygg, f.eks. den elektriske kjelen. Videre kan betydelige reduksjoner i netto nåverdi av både total systemkostnad og driftskostnad ses. De mest lovende casene ble funnet da begge systemene ble tvunget til å adlyde ZEB-begrensningen med termisk masse som lager, som viste reduksjoner i driftskostnader på 8,60% og 7,79% (sammenlignet med ingen termisk masse / noBITES) for punktkilde og vannbårne systemer. I tillegg ble det observert en tilsvarende reduksjon i totalimport av elektrisitet i disse to tilfellene, noe som tyder på at ”on-site” produksjonen fra solcellepaneler brukes til å forvarme bygningen før kveldstid, da både spotprisene og romoppvarmingsbehovet generelt er høyere. Sensitivitetsanalysen viser at den anvendte representasjonen for bygningens termiske masse utviser en forholdsvis liten følsomhet for sine parametere. De vurderte verdiene, som i det ekstreme tilfellet varierte med fem størrelsesordener, ga en rekkevidde for den totale kostnadsreduksjonen på mellom ca. 1300 € og 2500 € gjennom byggets levetid.

Preface

This thesis marks the end of the five year Master's degree in Electric Power Engineering and Smart Grids. It has been carried out in the spring of 2019.

I would first like to thank my thesis supervisor, associate professor Karen Byskov Lindberg of the Department of Electrical Power Engineering at NTNU and senior researcher at SINTEF Community. The door to Lindberg's office was always open whenever I ran into a trouble spot or had a question about my research or writing. She consistently allowed this paper to be my own work, but steered me in the right the direction whenever she thought I needed it.

Finally, I must express my very profound gratitude to my family and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Marius Bagle

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Perhaps the greatest challenges facing mankind in the 21^{st} century is to reduce the impact of global warming. The European Union is committed the goal of the United Nations Framework Convention on Climate Change (UNFCCC), which is to limit the temperature rise to 2 °C [7]. In order to fulfill this goal, strong reductions in greenhouse gases have to be achieved. One Representative Concentration Pathway (RCP) scenario, RCP2.6, actually predicts that negligible, or even slightly negative emissions has to be become the norm by 2100 [7]. Buildings take up a large share of the total energy consumption, both in the EU and in Norway. According to [8], they account for approximately 36 % of the total greenhouse gas emissions in Europe, and have a large potential for mitigation. In [1], a revision of the EU's Energy Performance of Buildings Directive, it is stated that all buildings constructed in the EU after 2020 shall be able to reach nearly zero energy level. The energy performance of nearly zero energy buildings (nZEBs) is high, both because of low energy demands, a result of constructing the house in accordance with passive house principles, and that these demands can be covered by on-site renewable generation. From 2009 to 2017, the Norwegian Research Center on Zero Emission Buildings (ZEB Center) was a leader in the joint European efforts to investigate the possibilities and challenges associated with ZEBs. Currently, the ZEB project is transitioning into Zero Emission Neighborhoods in Smart Cities (ZEN), in cooperation with the center for Environment-friendly Energy Research [9].

ZEBs has a different impact on the energy system than nonZEBs, since that they have lower energy demands. Additionally, some of the electricity produced by the on-site renewable technologies, such as PV-panels, will be fed back to grid, especially if no batteries are present to store the electricity. This presents a challenge for the grid operators, since the power system is not designed for bi-directional power flow [10]. Thus, it is of great interest to examine the duration curves for the import and export of electricity. Emissions caused by the operation of the building energy system are represented by a weighing factor, e.g. a CO_2 equivalent, which defines the amount of CO_2 (usually in kg) associated with importing 1 kWh of electricity. The value of the CO_2-factor for electricity is hard to determine precisely, and is hotly debated topic [need source]. Naturally, it depends on the resources used in the generation of electricity (the energy mix), which in Europe to a large extent is non-renewable. Norway, on the other hand, has an energy mix which almost exclusively consists of hydro power, which leads to a relatively low CO_2-factor [11]. In some ways, this can said be to have had a detrimental effect on the heating systems present in the Norwegian building stock, since low electricity prices has lead to few incentives to invest in high efficiency heating technologies such as heat pumps. Instead, Norway is one of the countries in the world with the most widespread use of direct electrical (ohmic) heating [12]. However, with the emergence of ever more HVDC-connections to continental Europe and Britain, and subsequently more energy trade, it is increasingly difficult to determine the origin of the electricity that is actually supplied to the end-user. Hence, one should not consider these CO_2-factors to be values set in stone, but rather as an approximation, meant to reflect the aggregated energy mix in a given scenario.

The authors of [8] claim that upgrading energy systems can lead to a reduction in operational costs of up to 80 %. Therefore, finding cost-optimal design of the energy systems in ZEBs is of vital importance, since Photo-voltaic panels currently are expensive investments [13]. This thesis builds on an optimization model developed in [2], which in turn is based on the work in [4]. The idea is to use a Mixed Integer Linear Program (MILP) to find the optimal investments in technologies and the optimal operational pattern of these technologies, given a certain input. In the above-mentioned works, the heat demand was considered as an aggregated load, whereas it is decoupled into two separate loads in this thesis. Furthermore, the thermal mass of the building is added as a storage technology, to investigate the flexibility potential that can be achieved by pre-heating the building in peak-load situations.

1.2 Approach and Limitations

The optimization model studied in this thesis is a deterministic mixed integer linear program (MILP), separating the heat demand into two separate loads; space heating and domestic hot water. The objective of the model is to minimize the net present value (NPV) of both the investment costs and the operational costs (the sum of these two is henceforth called the total discounted system cost, or simply total cost). The operation is optimized on an hourly basis, that is, for the heat or power production of each selected technology, the optimal output is found for each hour of the year. The lifetime of the building is set to 60 years, and the investments take place in the beginning of year 1, and reinvestments, which arise out of the fact that the technologies' lifetime is shorter than the lifetime of the building, are discounted back to year 1. The load data is based on a multiple linear regression approach, outlined in [4], [5], [6].

One consequence of decoupling the heat demand is that number of variables increases drastically, especially for the waterborne model, since the heating technologies in this configuration can operate on both the SH- and DHW-load. This causes convergence issues for the branch and bound algorithm, as the tree that has to be traversed in order to find a valid integer solution from the LP-relaxation becomes significantly larger. Thus, a reduced dataset is used. The method used to reduce is rather crude, but its use is justified in the sense that the results of the different cases can be compared directly.

As far as the zero emission aspect goes, the main limitation of this thesis is that it only considers the emissions in the operational phase. For a full accounting of the emissions caused by the building, other phases need to be taken into consideration, such as the production of the construction materials, the construction of the building itself and the production of the technology equipment. The balancing level used in this thesis is called "ZEB-O EQ", which means that only emissions associated with the operation of the building energy system, excluding the production of the technology equipment, is compensated for by on-site electricity generation [3].

1.3 Structure

The thesis is structured in the following manner:

- Chapter 2: Gives an overview of the most essential theoretical concepts to the modelling framework. Among these are the ZEB-concept itself, the basic principles of linear programming (including a short description of the branch and bound algorithm), demand side management (DSM), a basic outline of the technologies included in the model, the basics of heat storage (first law of thermodynamics) and the two-node representation of the building thermal mass. Additionally, an alternative representation of the building thermal mass is developed, using a circuit analogy.
- Chapter 3: Contains a description of the MILP optimization model. First, the variables and parameters of the model are tabulated. Then, the objective function and model constraints are described. Lastly, a clear distinction is made between the point-source and waterborne model, with figures to illustrate the different systems in clear manner.
- Chapter 4: presents the input data used for the optimization.
- Chapter 5: Gives a thorough treatment of the main cases studied, with a discussion of the results at the end of each section. Then, the sensitivity analysis with respect to the parameters of the two-node representation is performed.
- Chapter 6: Delivers the final conclusion and suggestions for further work.
- Appendices includes the Python/Pyomo code used for the optimization. Appendix A contains the code for the waterborne system, and Appendix B the code for the point-source system.

Chapter 2

Theory

In this chapter, some essential concepts for the thesis is presented. The intention is not to give exhaustive theoretical descriptions, but rather a background on which the modelling framework can be built. First, the Zero Emission Building Concept is presented, along with a short description of the Mixed Integer Linear Programming and the technologies included in both the point-source and waterborne models. Then, the concept of demand side management is explained, with a subsequent focus on thermal energy storage and modelling of the thermal mass.

2.1 Zero Emission Building-concept

The concept of Zero Energy/Emission Buildings was introduced by the EU's Energy Performance of Buildings Directive (EBPD) in 2010 [2]. In [1], it is stated that all buildings built in the EU after 2020 are to be *nearly zero energy buildings*. The definition of such a building is that it has very high energy performance. First of all, the energy demand of the building should be low, additionally, this demand should be covered to a significant degree by energy from renewable sources, on-site or from sources in close proximity to the building [1]. The research center on Zero Emission Buildings provides a definition leaning more towards the emission perspective: *"A zero emission building produces enough renewable energy to compensate for the building's total greenhouse gas emissions throughout its lifetime"* [3]. In this context, the concept of weighing factors must be introduced. The fundamental idea is to assign a crediting factor f_i to each energy carrier, such that an accumulated balance of the environmental impact of each can be conducted for a given time period t. Either the primary energy factor (PEF) or CO_2-factor can be used as crediting factor, with the former implying a Zero *Energy* Building, and the latter implying a balance more in line the emission perspective; Zero *Emission* or Zero *Carbon* Building [14] [3]. Then, the ZEB-balance can be introduced:

$$\sum_i import \cdot f_i - \sum export_i \cdot f_i = G \quad \forall i \in \mathcal{I} \quad [4] \tag{2.1.1}$$

where $\mathcal{I}$ is the set of all available energy carriers, and the period over which the accounting is done usually is set to a year. In other words, a ZEB is a building that can compensate for the accumulated emissions throughout the phases of its study [3].

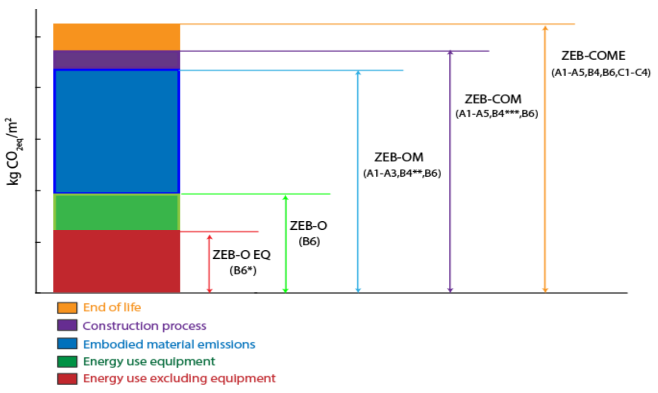

Figure 2.1.1: ZEB-phases. Adopted from [3].

For a complete accounting of the building's lifetime, all phases of have to be considered, including emissions associated with production of construction materials, the construction of the building itself, emissions caused by production of the energy technologies, emissions caused by the operation of the energy system and end-of-life emissions. In figure 2.1.1, this is presented visually. The ambition level in this thesis is ZEB-0 EQ, i.e. emissions in the operational phase, excluding equipment.

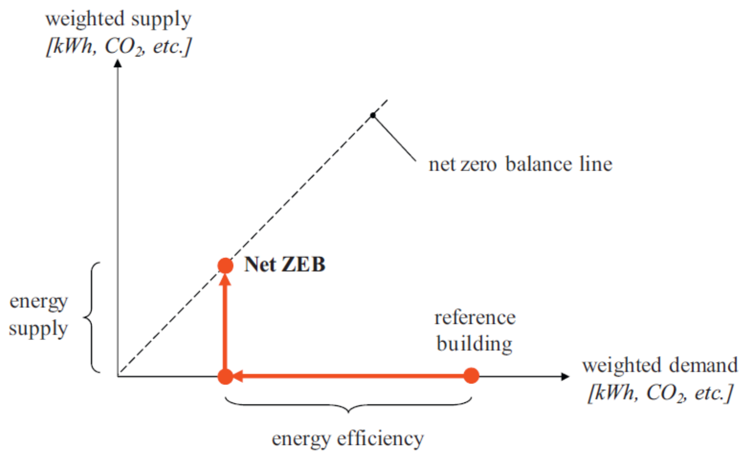

Figure 2.1.2: ZEB-balance. Adopted from [3].

Figure 2.1.2 shows the close relation between ZEBs and building houses in accordance with passive house principles. When the energy demand of the building is low, it is easier to reach the net ZEB-balance. In this thesis, the CO_2-factor is used as the weighing metric. Furthermore, the balance equation is rewritten as:

$$P_{ZEB} \cdot \sum_i import \cdot f_{CO_2} \leq \sum_i export \cdot f_{CO_2} \quad \forall i \in \mathcal{I} \tag{2.1.2}$$

for the time period t and all technologies i in $\mathcal{I}$ of the building energy system, where $P_{ZEB} = 0$ corresponds to $G = G_{ref}$, the total emissions with no ZEB-constraint, and $P_{ZEB} = 1$ corresponds to $G = 0$.

2.2 Mixed-Integer Linear Programming

The energy programming problem in this thesis is formulated as a Mixed-Integer Linear Program problem. That is to say, it is on the form:

$$\begin{aligned} \text{minimize } & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to } & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \end{aligned}$$

where some or all of the elements x_j of the solution vector $\boldsymbol{x}$ take integer values. It is necessary to restrict some variables to integer values due to the non-continuous nature of some decisions [15]. The constraints can be classified into three main groups: Technology constraints, balance constraints (for space heating, domestic hot water and electricity) and emission constraints. The objective is to minimize the total costs, given the set of restrictions. The linear model is implemented in Python, using the Gurobi solver and the modelling extension library Pyomo.

2.2.1 Branch and Bound

Problems of this type are most often solved using the branch-and-bound algorithm. It can be described as state-space search, where each state is a more restricted version of the original problem. First, the LP-relaxation of the original problem is solved. Then, if this solution does not hold up to the integer constraints, a variable that has an integer restriction in the original problem, but is fractional in the LP-solution, is *branched* on, which creates two sub-MIPS. This process can be repeated for each of these new nodes (states). The process is visualized in the figure below:

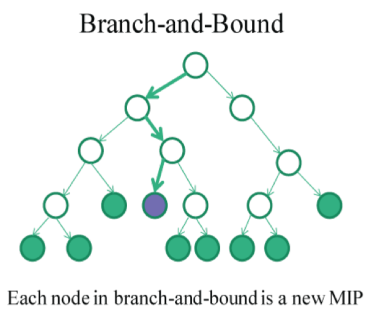

Figure 2.2.1: Illustration of the branch-and-bound process. Adopted from [15].

During the search, the best state at any given time is called the *incumbent*. Nodes which have yet to be branched upon are called *leaves*. When a node is *fathomed*; that is to say that the LP-relaxation is found to be infeasible, or its objective value is found to be less optimal than the incumbent, it and all of its possible branches are discarded, since no further branching will yield the optimal solution. If the gap between the incumbent value (upper bound) and the best (lower) bound is zero, optimality is demonstrated, and the search can be terminated. In practical applications, this is not always possible, and a gap must be tolerated in some cases [15] [16].

2.3 Technology Description

In this chapter, a brief description of the technologies represented in the optimization model will be given. In a MILP-optimization framework, the technologies have to be modelled on a highly generalized level, but it is still preferable for the technologies to retain some of their differentiating characteristics. For instance, the COP for the Heat Pumps is fed into the model as time-series, where the source and sink temperature for each timestep is considered. Here, the difference in performance between a ground-source heat pump and an air-source heat pump will manifest itself, as the ambient air-temperature will fluctuate through the year, while the groundwater assumed as the source for the GSHP will stay relatively constant.

2.3.1 Solar Panels

A PV-cell works by allowing photons to knock electrons free atoms from a semiconductor, typically made of silicon. In order for electricity to flow, an electric field needs to be established. This is done by so-called doping of the silicon in the cell, using phosphorous in the top layer, yielding a negative charge, and boron in the top layer, resulting in a positive charge. A PV-panel is made up of many such cells, while a PV-module in turn consists of several panels [17].

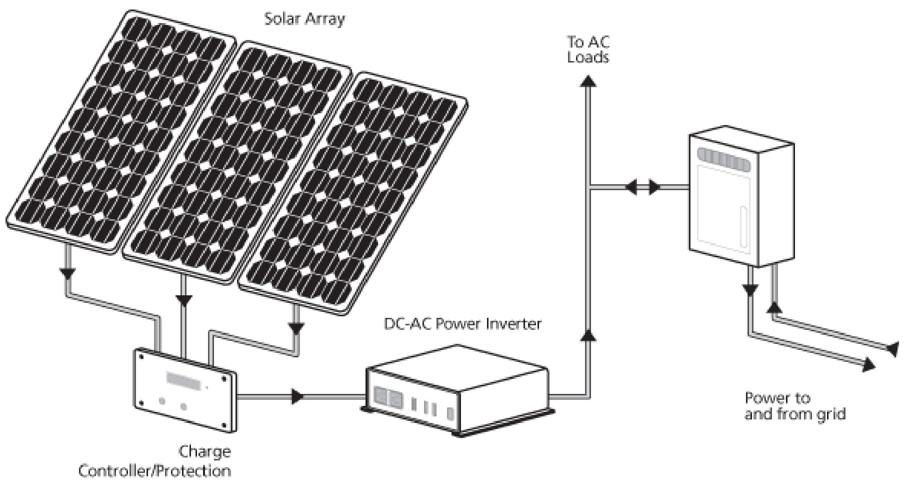

Figure 2.3.1: Principal sketch of grid-connected PV-module. Adopted from [18]

To enable to electricity flowing from the PV-module to be consumed by the load or exported to the grid, an inverter is needed, which converts the direct current (DC) of the panels to alternating current (AC). A simplified sketch of a typical configuration for a module is shown in figure 2.3.2. PV-panels is an essential part of a building energy system aspiring to maintain the zero emission operation, as a significant amount of electricity has to be exported in order to compensate for the power consumed by heat pumps, electric boilers and electronic appliances in winter.

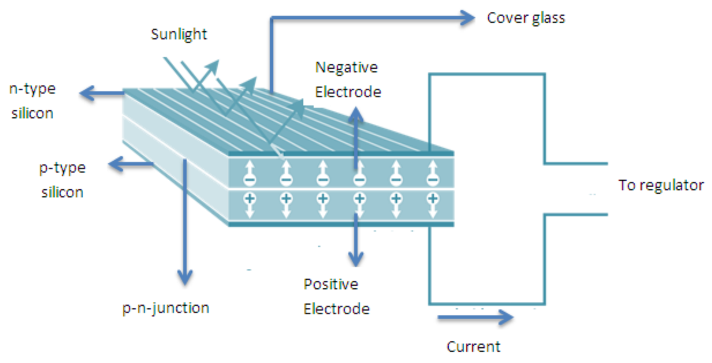

Figure 2.3.2: Principal drawing of PV-cell. Adopted from [19]

Usually, the calculation of the potential power output from a PV-panel is done using computer software, since it involves many variables, such as panel temperature, orientation of the panel in relation to the sun among others [20]. In [21], a simplified approach is suggested, which is the one used in this work:

$$P_{pv}(kW) = P_{stc} \cdot \frac{I_t}{I_{stc}} \cdot \eta_{rel} \tag{2.3.1}$$

where P_{pv} is the power generated, P_{stc} is the output at standard conditions, $I_t(W/m^2)$ is the irradiation at time t and I_{stc} is the irradiation at standard conditions: 25 °C and 1000 W/m^2. The efficiency is calculated for each time step according to the following equation:

$$\eta_{rel} = 1 + k_1 ln(I') + k_2 ln(I') + T_t'(k_3 + k_4 ln(I') + k_5 ln^2(I')) + k_6 T_t^2 \tag{2.3.2}$$

where $I' = I_t/I_{stc}$, $T' = T_{amb} + c \cdot i_t - T_{stc}$, where the coefficient c °C $[W^{-1}m^2]$ denotes to which degree the module is heated by the solar irradiation. The output of 2.3.1 gives the potential production per kW installed (kWh/kWp). The orientation angle for the module is assumed to be the optimal one for a module installed in Oslo, namely 40 °C [22].

2.3.2 Heat pumps

Another technology that is important from an emission reduction perspective is the heat pump. A heat pump works by moving thermal energy in the opposite direction of spontaneous heat transfer, moving heat a lower temperature to a region with higher temperature (heating mode), or heat at a higher temperature to a region with lower temperature (cooling mode), using a relatively small amount of high-grade energy (typically electricity) for the process. The most common design for a heat pump involves four principal components: a condenser, an expansion valve, an evaporator and a compressor. The medium used for the heat transfer is called a refrigerant [23].

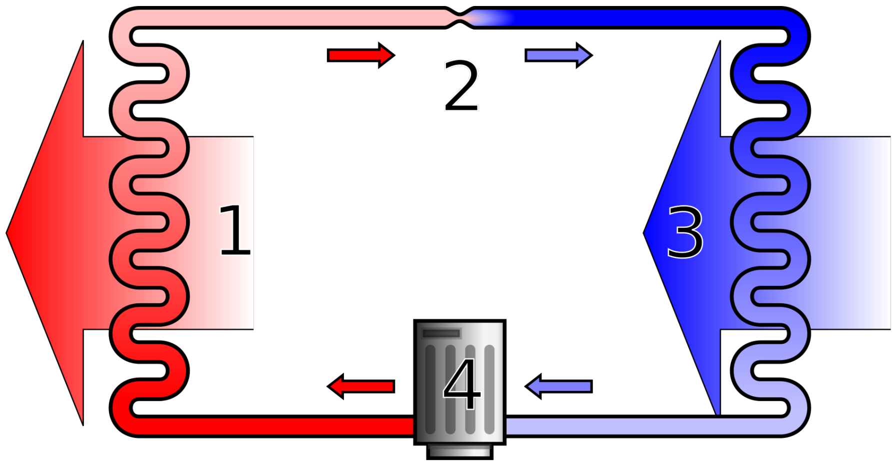

Figure 2.3.3: Simplified sketch of heat pump and its main components. Adopted from [23]

In figure 2.3.3, the main working principles of a heat pump is illustrated. The refrigerant is circulated through the system in its gaseous state by a compressor (4). When the heat is discharged in the condenser (1), the high-temperature and high-pressure vapor is cooled to a high-pressure, moderate temperature liquid. Then, the condensed working fluid is passed through a pressure-lowering device, most commonly an expansion valve (2). The low-pressure refrigerant subsequently enters the evaporator (3), another heat exchanger, where the it absorbs heat and boils. The refrigerant, now at high temperature and low pressure, enters the compressor, and the cycle is repeated [23]. The upper limit for the efficiency of a heat pump operating in heating mode is given by considering the reverse Carnot cycle, which yields the following expression [24]:

$$COP = \frac{T_H}{T_H - T_L} = \frac{1}{1 - \frac{T_L}{T_L}} \tag{2.3.3}$$

where T_H is the temperature of the sink, T_L the temperature of the source and COP is the ratio of delivered heat to work. An expression for the actual COP can be written as [24]:

$$COP = \frac{Q_h}{W_{in}} \tag{2.3.4}$$

where Q_H is the delivered heat, and W_{in} the work required to transfer this heat from source to sink.

Heat pumps are separated into two main categories based on the location of the external heat exchanger. Either heat is drawn from the ambient air, or from below ground. Air-source heat pumps (ASHP) are the most common, using a small ground or wall-mounted outdoor unit. An advantage of this heat pump type is that they are easy to retrofit into existing houses, and that they require relatively little space [23]. Two main varieties of the ASHP exist: air-to-air (henceforth denoted A2A) and air-to-water (henceforth denoted ASHP). The first of these heats the air of a room directly, using a wall-mounted indoor unit. Multi-split systems allows for multiple rooms to heated, by connecting a single compressor to several indoor units. In this thesis, however, the A2A will be limited to provide heating for one room through a modelling restriction, which states that only 40 % of the space heating demand can be covered by the heat pump at any given time. A2A heat pumps can provide additional services, such as dehumidification and air purification (removal of odours, smoke, bacteria etc.) [23].

Air-to-water heat pumps, on the other hand, are integrated into a waterborne central heating system (where heat convection is achieved either through the use of radiators or underfloor heating), providing heating for the whole building, in addition to water heating (domestic hot water). Besides the outdoor compressor unit, the ASHP requires a compact heat exhanger and control unit in close proximity to the the hot-water cylinder to transfer heat from the heat pump's refrigerant [23]. Some systems integrate these units with the compressor, which yields a somewhat larger outdoor unit.

The second main category of heat pumps is called ground source heat pumps (henceforth denoted GSHP). They use copper or plastic tubes buried below ground-level as the external heat exchanger. This allows them access to higher quality heat, at the cost of more expensive and disruptive installation. The system loop can be either open-loop or closed-loop.

With and open-loop system, water is extracted from and rejected directly back into rivers or groundwater sources. These sources provide a stable source of heat, usually about 5 to 10 °C, however, environmental regulations and activity (acidity, corrosion etc.) can create challenges for this type of system. Therefore, closed-loop systems are more common. In these systems, a sealed loop is used to retrieve heat from the surrounding soil or rock. Direct expansion systems circulate the refrigerant directly from the compressor trough copper tubes. Indirect systems, on the other hand, have a two-stage circuit, with water and antifreeze circulating in plastic tubes absorbing the heat from the source, subsequently transferring heat to the refrigerant circuit via a secondary heat exchanger. The additional stage means that direct expansion systems are slightly more efficient, but as more refrigerant is required and regulations on leakage have tightened, the indirect systems are currently the more popular choice [23].

The performance of the heat pump is the key factor in determining the savings it can offer, both in terms of economics and environmental impact. As already shown, this performance is highly dependant on the temperature difference between the external heat collector (source) and the output to the building (sink), also known as the ”lift” [23]. In practice, the COP drops by between 0.6 and 1.0 for every 10 °C temperature decrease, yielding 0.6-1.0 kW less heat output per kW of electricity. To optimize the performance, the ”lift” must be made as small as possible. Thus, a relatively cool heating loop and warm external loop are desirable [23]. The first condition can be met by increasing the heating surface, using either fan-assisted radiators or underfloor heating. Heating the air directly with air-to-air heaters can improve upon this further, by lowering the output temperature even more. Modern heat pumps are also able to provide high temperature when required, e.g. for domestic hot water purposes. State-of-the-art hydronic heat pumps can supply hot water at above 65 °C, which means that they can function as a standalone unit, without any auxiliary heating. However, it must be kept in mind that this is done at the expense of performance, as the COP inevitably will be lower when water at such a high temperature is supplied.

The second condition, namely the desire for a relatively warm external loop, is where the GSHP has a notable advantage over the ASHP and A2A in terms of performance. Naturally, the greatest heat demand takes place in winter, when the air temperatures are at their lowest. The ground temperature, on the other hand, quickly converges to the annual mean as the depth is increased. Therefore, the GSHP offer a higher average COP through the year. Even at nominal conditions, they tend to offer a higher COP. Since the specific heat capacity of air is so much lower than that of soil or water, the heat extraction process using air is the source is more energy intensive, because more air has to be passed through the heat exchanger. Thus, for the ASHP and A2A heat pump, more electricity per kW of heat has to fed to the compressor. In this thesis, polynomials are used to calculate the COP for a given sink temperature. [25] and [23] suggest the following relationship for calculating the COP:

$$COP = k_0 - k_1 \cdot \Delta T - k_2(\Delta T)^2 \tag{2.3.5}$$

where the k-values are based on polynomial regression, where the data points are heat output and input electricity at certain temperatures (retrieved from manufacturers data), and ΔT is the difference between source and supply temperature, given by the following equation:

$$\Delta T = T_{supply} - T_{source} \tag{2.3.6}$$

To calculate the supply temperature, the equation below is used [4]:

$$T_{sup} = AT_{amb}^2 + BT_{amb} + C \tag{2.3.7}$$

where the coefficients A, B and C are given by the building standard.

2.3.3 Direct Electrical Heating

Electrical Boiler

An electrical boiler is a device that uses electricity to directly heat water. For space-heating purposes, it is typically used in peak-load situations. In the modelling framework developed in this work, it can be used for both space-heating and domestic hot water purposes in the waterborne configuration, and for domestic hot water use only in the point-source configuration. Typically, it has an efficiency of close to 100 %. The main advantage of the electric boiler is that is a simple device, and thus easy to install. This is reflected in the lack of a fixed cost in the model (no installation costs). The downside is that it tends to be expensive to operate [26].

Simple Resistance Heating

An electrical resistance heater works in a similar manner as the electrical boiler, heating the surrounding air instead of the water. The heat developed in the resistor is given by the ohmic losses, $P = I^2R$. For the heater to work properly, materials with high resistivity and small variation of resistivity with temperature are used [27]. It is a cheap alternative to heat pumps as a peak-load technology, and is the most ubiquitous heating technology in the Norwegian household [12]. In this work, the resistance heater is an option only in the point-source configuration.

2.3.4 Fireplace

The fireplace is another well-established technology for space heating in the Norwegian building stock [12]. It is introduced as a peak-load alternative to resistance heating for situations in which the electricity price is high. One aspect worth mentioning when modelling this device is that it requires manual refilling of wood for a sustained power output. This can be put in as a time restriction, for instance by only allowing the fireplace to operate on the load between 16:00 and 24:00.

2.3.5 Biomass Boiler

The biomass boiler used in the model is a pellet-fired, fully automatic biomass heater. For the model configuration with hydronic heating, it is allowed to operate on both space heating and domestic hot water loads. According to [28], using biomass in boilers has been found to offer many economical and environmental benefits, such as financial savings, conservation of fossil fuel resources and CO_2- and NO_x- emissions reduction. In addition, it is estimated in [29] that the thermal energy potential from biomass regeneration in Norway is 140 TWh each year.

2.4 Demand Side Management

To make it easier for a building to reach the goal of a strict ZEB-balance over a year, it is reasonable to assume that it will be necessary to invest in some energy storage technologies. Due to the intermittent nature of the energy production from renewable sources such as wind and solar, the capacity of these technologies has to be dimensioned in such a way that enough electricity is exported when the conditions are right (i.e., the sun is shining and the wind is blowing). This would preferably done in combination with an investment in energy storage technologies, to introduce the possibility of load shifting [4]. The potential for flexible load shifting is greater when investing in batteries, since the battery can operate on both the thermal electric demand and specific electric demand, whereas a thermal storage can only operate on the thermal electric demand [4]. However, the focus here will be on thermal storage, as research with regard to the viability of batteries in a ZEN/ZEB-context already has been done in [14] and [2].

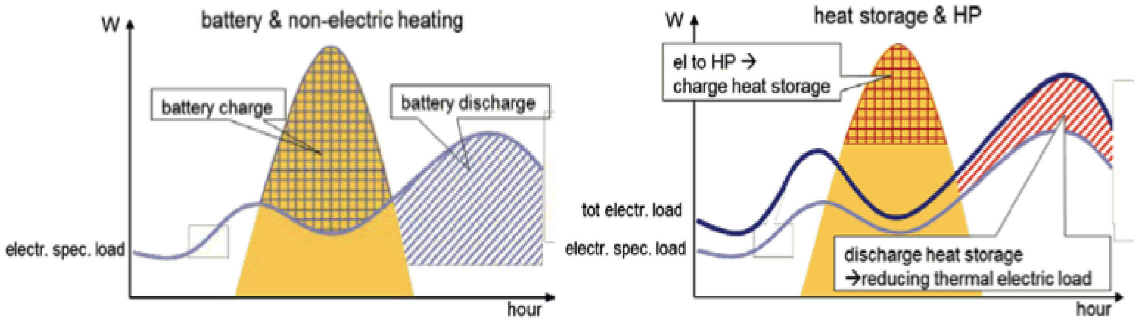

Figure 2.4.1: Conceptual plots showing potential for load shifting for battery and heat storage. Adopted from [4].

Figure 2.4.1 shows this concept for PV-production with a battery and a heat storage. The power production takes place in the middle of day, with the intensity given by a bell-like curve. In the plot on the left, the battery is charged by the superfluous PV-production during the day, and distributed to the load in the evening. Thus, the electric load is reduced [4]. In the plot on the right, the superfluous electricity from the PV-production is used to power a heat-producing technology, for instance a heat pump, which in turn charges the heat storage. This heat can then be used in the evening, when the heat load is higher and the PV does not produce power, avoiding having to import power from the grid. Notice that the area on which the battery can operate is larger than the area for the heat storage, since the total electric load is the sum of the electric specific load and the thermal electric load [16].

2.5 Thermal Energy Storage

In the following subsection, the basic principles of thermal energy storage (TES) will be treated. The modelling of TES in work will be constrained to obeying only the first law of thermodynamics, i.e. changes in entropy are not considered. The energy storage capacity of a storage medium at uniform temperature with an associated temperature difference is:

$$Q_s = (mC_p)\Delta T_s \tag{2.5.1}$$

where Q_s is the total heat capacity when operating over the entire energy difference, C_p the specific heat capacity of the medium and m the storage mass. An energy balance for the system in figure 2.5.1 can be derived:

$$(mC_p)_s \frac{dT_s}{dt} = \dot{Q}_{in} - \dot{Q}_{out} - (\sigma_s)(T_s - T_a) \tag{2.5.2}$$

where $\dot{Q}_{in}$ and $\dot{Q}_{out}$ is the rate of addition and removal of energy to and from the storage, and T_a is the ambient temperature for the tank surroundings. σ_s represents the thermal conductivity of the tank walls. To find the tank temperature at time $t = i + 1$, a simple forward Euler first-order approximation can be performed, where the term $\frac{dT_s}{dt}$ is rewritten to $\frac{T_s^{i+1} - T_s^i}{\Delta t}$ [30]:

$$T_s^{i+1} = T_s^i + \frac{\Delta t}{(mC_p)_s}[Q_{in} - Q_{out} - \sigma_s(T_s^i - T_a)] \tag{2.5.3}$$

where Δt usually is set to be an hour, the rates Q_{in} and Q_{out} are assumed constant through the hour.

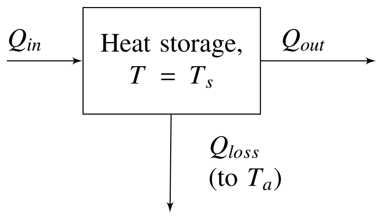

Figure 2.5.1: Flow of energy in simple heat storage with fully mixed storage medium. Inspired by figure 8.3.2 in [30].

2.5.1 Hot Water Tank

From the simple general model outlined above, a specific model for a hot water tank can be derived. The hot water tank is by far the most ubiquitous form of energy storage in the existing building stock. Fixing the timestep to one hour, and rewriting the terms Q_{in} and Q_{out} to $Q_{ch.}$ and $Q_{disch.}$ (which better describes the tank as a sort of battery analogy), the temperature of the tank can be written as [31]:

$$T_{HWT(t)} = T_{HWT(t-1)} + \Big(\frac{Q_{ch.}^{HWT} \cdot \eta_{ch.}}{V \cdot Cp_{water} \cdot \rho_{water}}\Big) - \Big(\frac{Q_{disch.}^{HWT}/\eta_{disch.}}{V \cdot Cp_{water} \cdot \rho_{water}}\Big) \tag{2.5.4}$$

$$T_{HWT(t)} = T_{HWT(t-1)} + \frac{1}{V \cdot Cp_{water} \cdot \rho_{water}}\Big(Q_{ch.}^{HWT} \cdot \eta_{ch.} - \frac{Q_{disch.}^{HWT}}{\eta_{disch.}} - \sigma_s(T_{HWT(t-1)} - T_{a(t)})\Big) \tag{2.5.5}$$

where the mass m is substituted for the volume V and the density ρ_{water} of the water. Charging and discharging efficiencies $\eta_{ch.}$ and $\eta_{disch.}$ are also included. These terms account for the losses that occur when the hot water at the top of the tank and the cold water at the bottom is mixed during the charging and discharging of the tank [31]. This causes some of the heat stored in the tank to be irretrievable. To avoid nonlinearities in the tank model (it will be implemented in a linear framework, after all), these efficiencies are assumed to be constant, although they will depend on the (non-constant) temperature difference between the tank and the ambient. Further, it should be pointed out that this model of the tank assumes a uniform water temperature (that is, no stratification). Instead of treating the tank in terms of its temperature, we can write the balance with respect to energy stored [31]:

$$Q^{HWT}_{stored(t)} = Q^{HWT}_{stored(t-1)} + Q^{HWT}_{ch.} \cdot \eta_{ch.} - \frac{Q^{HWT}_{disch.}}{\eta_{disch.}} - Q^{HWT}_{loss(t)} \tag{2.5.6}$$

where $Q^{HWT}_{stored(t)}$ and $Q^{HWT}_{stored(t-1)}$ is the energy stored in the tank in timesteps t and $t-1$, respectively. The term $Q^{HWT}_{loss(t)}$ can be written as [31]:

$$T^{HWT}_{loss(t)} = (T_{HWT(t)} - T_a) \cdot U \cdot A_{HWT} \tag{2.5.7}$$

where U [kW/m^2] is the heat transmission coefficient of the tank, A_{HWT} is the surface area of the tank (minus the bottom area, which is not in contact with the air), and the term $U \cdot A_{HWT}$ corresponds to the term σ in the equations 2.5.3 and 2.5.5.

2.5.2 Building Internal Energy, Thermal Mass

Another possible form of energy storage is to use the thermal mass of the building. This is done in the context of a larger district heating system in [31]. In this work, the two-node model from that paper that is implemented. The two-node model is based a number of thermal response tests done in [32] and the modelling in [33]. The two nodes represent a "shallow" component, which is assumed to consist of the building space heating system, the indoor air, and the parts of the building that easily transfer heat to the indoor air (i.e. furniture and the outer layers of the walls). The amount of heat in the shallow storage is assumed to be directly proportional to the indoor air temperature. The "deep" component represents the structural elements of the building [31]. The following difference equations describe the dynamics of the two-node model:

$$Q^{shallow}_{stored(t)} = Q^{shallow}_{stored(t-1)} + Q^{shallow}_{ch(t)} - Q^{shallow}_{disch.(t)} - Flow_t - Q^{shallow}_{loss(t)} \tag{2.5.8}$$

$$Q^{deep}_{stored(t)} = Q^{deep}_{stored(t-1)} + Flow_t - Q^{deep}_{loss(t)} \tag{2.5.9}$$

where $Flow_{(t)}$ denotes the energy exchange between the deep and shallow components of the two-node model in a given timestep t. It is calculated in the following manner:

$$Flow(t) = \left(\frac{Q^{shallow}_{stored(t)}}{Q^{shallow}_{cap}} - \frac{Q^{deep}_{stored(t)}}{Q^{deep}_{cap}} \right) \cdot K \tag{2.5.10}$$

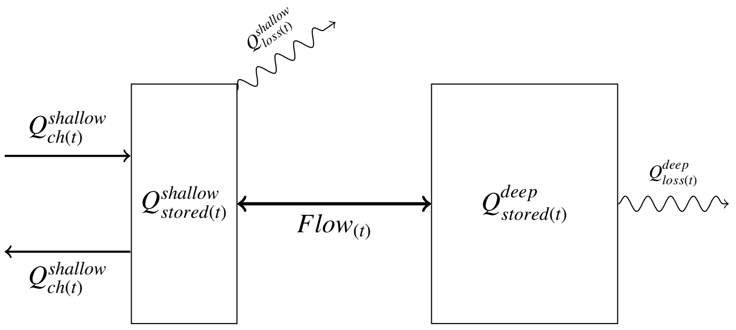

Figure 2.5.2: Two-node model for building internal energy. Inspired by figure 2 in [31]

where $Q^{shallow}_{cap}$ and Q^{deep}_{cap} are the maximum storage capacities of the shallow storage and the deep storage, and K is the heat transfer coefficient, which in the work of Carlsson [33] is defined as the heat transfer between the two nodes when one storage is fully charged and the other is fully discharged. As can be seen from the equations, heat will tend to flow from the shallow storage to the deep storage when the shallow storage is charged to a higher relative level than the deep storage and vice versa.

Because the building heating system is part of the shallow storage, this is the only component that can be directly charged or discharged. It is also the only component that can operate on the building space heating load. The losses from the shallow and deep storage components are defined through the following equations:

$$Q^{shallow}_{loss(t)} = Q^{shallow}_{stored(t-1)} \cdot (1 - K^{shallow}_{loss}) \tag{2.5.11}$$

$$Q^{deep}_{loss(t)} = Q^{deep}_{stored(t-1)} \cdot (1 - K^{deep}_{loss}) \tag{2.5.12}$$

where $K^{shallow}_{loss}$ and $K^{shallow}_{loss}$ are the heat loss coefficients of the shallow and deep components. Transmission losses due to heat convection in the structural elements of the building are represented by $Q^{deep}_{loss(t)}$. The losses from the shallow storage are assumed to be caused by the ventilation only, as other losses, such as radiation through windows, air leakage etc. are small in comparison. The losses from the shallow component can be found with the following equation:

$$Q^{shallow}_{loss(t)} = \dot{V} \cdot A_{building} \cdot \rho_{air} \cdot Cp_{air} \cdot \Delta T \tag{2.5.13}$$

where $\dot{V}$ is the ventilation flow rate, $A_{building}$ is the area of the building used as thermal energy storage, ρ_{air} and Cp_{air} are the density and specific heat of air, and ΔT is the change in temperature. Another way to find the heat loss coefficients, which also makes it possible to find the coefficient for the deep component, is by using the following relation [34]:

$$K_{loss} = e^{-\frac{1}{\tau}} \tag{2.5.14}$$

where τ is the time constant of the component in question. According to [31], this parameter will vary between 100-350 hours, depending on how "heavy" the building is. For the modelling done in this paper, the values from that work will be used. The following table summarizes the parameters that will be used in the model:

Table 2.5.1: Parameters of shallow and deep components of BITES. Taken from [31].

Parameter	Shallow storage	Deep storage
Storage capacity: Q_{cap} [kWh]	12.5	90
Heat loss coefficient: K_{loss}	0.9913	0.9963
Max total loss [Wh/h]	394	
Heat transfer constant: K [Wh/h]	7881	
Ventilation flow rate: $\dot{V}[m^3/m^2s]$	0.00035	
$A_{building}[m^2]$	250	

Using the values for K_{loss} from the above table yields time constants $\tau_{shallow}$ and τ_{deep} of 115h and 267h, respectively. The usefulness of the building internal energy as a thermal energy storage will depend on how the building is constructed (materials etc.) and the degree to which the building temperature is allowed to fluctuate from the set-point temperature, ΔT, which in [31] is set to 1 K. In this thesis, only overheating is considered, so a deviation ΔT of 2 K from the set-point is tolerated. With the area-adjusted values from [31], this yields an extensive heat capacity of 6.25 kWh/K for the shallow storage, which is in the range found in [34], where buildings built in the seventies were found to have a storage capacity of about 8 kWh/K, and buildings from the eighties about 5 kWh/K. For the deep storage, the same reasoning yields an extensive heat capacity of 45 kWh/K.

2.5.3 Electrical analogy

In the previous section, a two-node model for BITES from [31] was presented. In this section, the similarity between this two-node model and a thermal network with three resistors and two capacitors will be shown. The comparison is based on an analogy between the flow of heat and the charge flow in electric circuits.

Table 2.5.2: Thermal-electric analogy. Adopted from [35].

Type	Thermal	Electric
Quantity	Heat Q [J]	Charge q [C]
Potential	Temperature T [K]	Voltage V [V]
Flux	Heat transfer rate $\dot{Q}$ [J/s]	Current i [C/s = A]
Flux density	Heat flux $\dot{q}$ [W/m^2]	Current density I [A/m^2]
Resistance	Thermal resistance R [K/W]	Electrical resistance R [Ω]
Conductivity	Thermal conductivity [W/(mK)]	Electrical conductivity σ [1/(Ωm)]
Lumped linear model	Newton's law of cooling $\Delta T = \dot{Q}R$	Ohm's law $\Delta V = IR$
Charge-preserving element	Extensive heat capacity C [kJ/K]	Electrical capacitance C [C/V = F]

Table 2.5.2 shows the different types of properties of heat flow and their electric counterpart. Consider as an example the potentials; just as a voltage difference across a resistor will give rise to an electrical current, a temperature difference across a heat-conducting element gives rise to a flow of heat. It is important to keep in mind that this analogy has limited practical use for detailed modelling of heat flow [36]. In particular, to model heat flow in a lumped-capacitance network, the temperature of each element has to be assumed constant. For this approximation to hold within a 5 % error margin, the Biot number, the ratio between convective at the surface of a body to the conduction within the body, has to be less than 0.1 [37].

In this work, the analogy is used as a justification for the use of time constants, which have a clear definition in a resistor-capacitor circuit. Furthermore, the analogy is used to evaluate the meaningfulness of certain BITES-parameters (K_{flow}, and can be used to constrain the ranges of these in a sensitivity analysis. It can also be used to find appropriate values for resistances to ambient, such that the BITES-model includes a loss element that depends on the outside temperature.

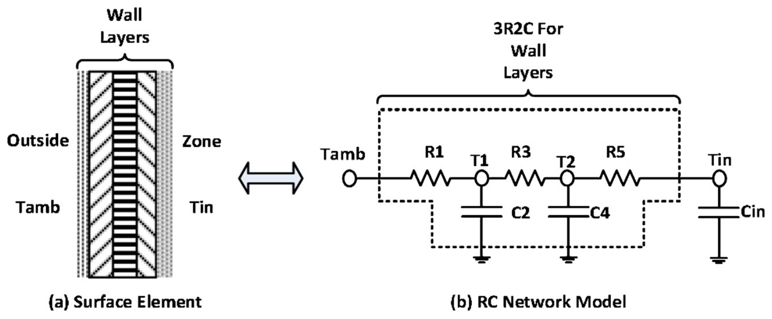

Figure 2.5.3: Circuit for wall thermal network. Adopted from [38]

In figure 2.5.3, a thermal network analogy for a wall with three layers is presented. The capacitance C_{in} is the heat capacity of the inside air and furniture, similar to the shallow storage in the previous section, with the exception that the shallow/outer walls are not included. C_4 is the capacitance between the shallow walls and the structural elements of the wall, and C_2 is the capacitance between the structural layers and the part of the wall facing the outside. The resistances are the reciprocal values of the heat conductivity, indicating resistance to heat flow. Note also the temperatures at the junctions, T_1, T_2, and T_{in}, representing the thermal potential.

Taking this abstraction further, we assume that we can represent all of the walls of the building as one single wall. Defining the the capacitance C_{ss} to be the aggregated capacitance of the outer wall layers, the inside air and the furniture (shallow storage), and the capacitance C_{ds} to be the capacitance of the deeper wall layers, we may draw the circuit:

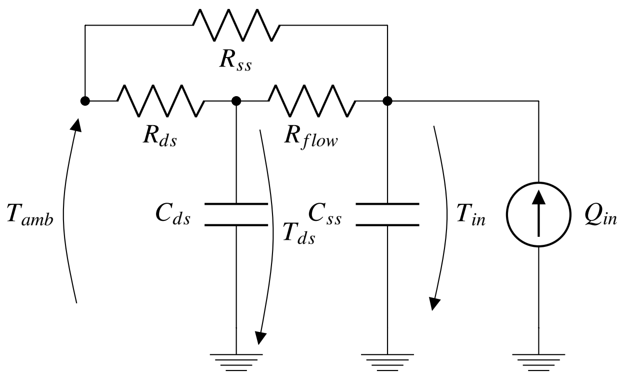

Figure 2.5.4: Thermal circuit for two-node BITES-model.

where the current source Q_{in} represents the heat added to the shallow storage, R_{ss} the thermal resistance between the shallow storage to ambient, R_{ds} the thermal resistance between the deep storage and ambient, and R_{flow} the thermal resistance between the shallow storage and deep storage. Keeping in mind that the current through a capacitor can be written as $i = C\frac{dv}{dt}$ [39], we can write out expressions for the heat flows (currents) based on the node-voltage method. For the heat added to the shallow storage:

$$\dot{Q}_{in} = C_{ss}\frac{dT_{in}}{dt} + \frac{T_{in} - T_{ds}}{R_{flow}} + \frac{T_{in} - T_{amb}}{R_{ss}} \tag{2.5.15}$$

For the heat flowing into the deep storage:

$$\frac{T_{in} - T_{ds}}{R_{flow}} = C_{ds}\frac{dT_{ds}}{dt} + \frac{T_{ds} - T_{amb}}{R_{ds}} \tag{2.5.16}$$

Discretizing these equations at the time step t (turning the differential $\frac{dT}{dt}$ into the difference $\Delta T = T[t] - T[t-1]$) and assuming constant temperatures at this timestep, the first-order differential equations become first-order difference equations:

$$C_{ss}(T_{in}[t] - T_{in}[t-1]) = \dot{Q}_{in}[t] \cdot \Delta t - \frac{T_{in}[t] - T_{ds}[t]}{R_{flow}} \cdot \Delta t - \frac{T_{in}[t] - T_{amb}[t]}{R_{ss}} \cdot \Delta t \tag{2.5.17}$$

$$C_{ds}(T_{ds}[t] - T_{ds}[t-1]) = \frac{T_{in}[t] - T_{ds}[t]}{R_{flow}} \cdot \Delta t - \frac{T_{ds}[t] - T_{amb}[t]}{R_{ds}} \cdot \Delta t \tag{2.5.18}$$

which resemble equations 2.5.8 and 2.5.9 closely. Noting that $\dot{Q} \cdot \Delta t = Q$ and that $C \cdot T = Q$, we may write:

$$Q_{ss}[t] = Q_{ss}[t-1] + Q_{in}[t] - Q_{flow}[t] - Q_{ss}^{loss}[t] \tag{2.5.19}$$

$$Q_{ds}[t] = Q_{ds}[t-1] + Q_{flow}[t] - Q_{ds}^{loss}[t] \tag{2.5.20}$$

where the difference from equations 2.5.8 and 2.5.9 resides in how Q_{flow} and the losses are defined. In the above equations, Q_{flow} is calculated directly as the temperature (energy content) difference divided by the resistance R_{flow}, whereas in 2.5.8 and 2.5.9, the flow is given by the difference in relative (to max capacity) energy content. In 2.5.8 and 2.5.9, the losses are calculated directly from the time constants via equation 2.5.14, that is, they are independent of the outside temperature T_{amb}. In the circuit analogy, they are calculated directly as the temperature difference divided by the relevant thermal resistances. The time constants can be defined as [39]:

$$\tau_{ss} = R_{eq,ss}C_{ss} \tag{2.5.21}$$

$$\tau_{ds} = R_{eq,ds}C_{ds} \tag{2.5.22}$$

2.5.4 Building Materials/Time constants of buildings

It is possible to expand the definition of the time constant from the previous section. Specifically, the definition of the thermal resistance can be broadened. In the first-order model, the thermal resistance was considered as a single parameter for the different storages. If we separate out the conductance from transmission (conduction) in the building materials and the conductance from ventilation (convection), the thermal resistance can be written as [34]:

$$R = \frac{1}{G_{tr} + G_v} \tag{2.5.23}$$

where G_{tr} [W/K] is the thermal conductance from transmission, and G_v [W/K] the thermal conductance from ventilation. The time constant can then be expressed as [34]:

$$\tau = \frac{\sum(m \cdot c_p)}{G_{tr} + G_v} \tag{2.5.24}$$

where $\sum(m \cdot c_p)$ is the heat storing capacity of all masses in the storage. We see that the time constant can be influenced by the building mass, the transmission losses and the ventilation losses. Thus, the time constant of the house can be increased not only by increasing the building mass and selecting materials with high heat capacity, but also by reducing the losses [34].

Table 2.5.3: Heat conduction and capacity values for different materials. Adopted from [34].

Material	Specific heat conductivity [W/(mK)]	Specific heat capacity
Brick	0.45	1.49
Concrete	2.7	1.83
Concrete, lightweight	0.13	0.4
Gypsum board	0.1	0.88
Wood (oak)	0.19	1.7
Wood (pine)	0.14	1.5
Glass-wool	0.045	0.062
Insulation (styrofoam)	0.035	0.01
Cork floor	0.1	0.36
Air, 0 °C	0.024	0.0013

The heat flow through a wall segment is given by:

$$q = \frac{k \cdot A}{x}(T_{in} - T_{out}) \tag{2.5.25}$$

where k is the specific heat conductivity, A is the area of the wall section, x the thickness of the wall, and T_{in}, T_{out} are the indoor and outdoor temperatures. Considering table 2.5.3, it is clear that the heat flow through a wall segment will vary to a large degree on the materials being used for insulation/conduction. The difference in conductivity, as mentioned in the previous section, is three orders of magnitude. Naturally, a material with low thermal conduction is useful for conduction. For thermal storage, a material with high heat capacity is preferred. As can be seen from table 2.5.3, no material satisfies both of these demands, so a wall is usually made of several materials, for instance wood for structural qualities, styrofoam for insulation, and brick on the outside for the sake of aesthetics [34].

Chapter 3

Model Description

3.1 Introduction

The model used is based on the work in [2]. It is a mixed-integer linear program implemented in Python, using the modelling library Pyomo. In [2], a stochastic formulation of the problem was investigated. In this work, however, the model is kept in a deterministic framework. The main work done in this thesis is decoupling the heat balance into two equations, one for space heating and another for domestic hot water. Hence, different options for building heating systems can be investigated, and using the thermal mass of the building as an energy storage can also be explored. (it would not make sense to let this storage operate on an aggregated heat load).

Table 3.1.1: Key information for the building studied.

Area	250 m^2
Location	Oslo, Norway
Type	Single family home
Standard	Passive house

The template building used for the case study is the same as the one used in [2]. Important information about the building is presented in table 3.1.1.

3.2 Nomenclature

In this section, an exhaustive description of the sets, parameters and variables of the model will be given. Parameters and variables for both model configurations are listed together, the different configurations are treated later in the chapter.

Table 3.2.1: Sets and indices for the model.

Set	Index	Description
$\mathcal{I}^{dhw}$	i	DHW-technology i
$\mathcal{I}^{sh}$	i	SH-technology i
$\mathcal{I}$	i	Technology i ($\mathcal{I}^{dhw} \cup \mathcal{I}^{sh} = \mathcal{I}$)
$\mathcal{I}^{z}$	i	Storage technology i
ϵ	e	Import energy carrier e
$\mathcal{T}$	t	Hourly time step t
Υ	yr	Yearly time step of modelling period
$\mathcal{S}$	s	Scenario s

Table 3.2.2: Declaration of parameters for the model.

Strategic Parameters		
C_i^{fxd}	Fixed investment cost for technology i	€
C_i^{spe}	Specific investment cost for technology i	$€/kW$ (kWh)
C_i^{run}	Yearly running cost for technology i	% of C_i^{spe}
C_{ep}^{fxd}	Monthly fixed grid tariff for ep (incl. VAT)	€
C_{ps}^{fxd}	Monthly fixed grid charge for ps pricing (incl. VAT)	$€/kW$
C_{ep}^{spe}	Monthly specific grid tariff for ep (incl. VAT)	$€/kWh$
C_{ps}^{spe}	Monthly specific grid tariff for ps pricing (incl. VAT)	$€/kWh$
C^{pty}	Penalty charge for ps pricing (incl. VAT) i	$€/kWh$
Y^{sub}	Power subscription for ps pricing (incl. VAT)	kW
C^{bf}	Price of bio fuel (pellets)	$€/kWh$
C^{wo}	Price of wood fuel (pellets)	$€/kWh$
R	Discount rate	-
η_i	Efficiency of technology i	-
β_i	Charging/discharging rate of technology of storage technology i	-
L_i	Expected lifetime of technology i	$Years$
$\overline{X_i}$	Upper capacity bound for technology i	$kW(kWh)$
$\underline{X_i}$	Upper capacity bound for technology i	$kW(kWh)$
Operational Parameters		
C_t^{spot}	Spot price of electricity at time step t	$€/kWh$
$COP_{t,ashp}^{sh}$	Coefficient of performance for ASHP for SH at time step t	-
$COP_{t,ashp}^{dhw}$	Coefficient of performance for ASHP for DHW at time step t	-
$COP_{t,gshp}^{sh}$	Coefficient of performance for GSHP for SH at time step t	-
$COP_{t,gshp}^{dhw}$	Coefficient of performance for GSHP for DHW at time step t	-
$COP_{t,a2a}^{sh}$	Coefficient of performance for A2A for SH at time step t	-
D_t^{el}	Building electricity demand at time step t	kWh/h
D_t^{sh}	Space heating demand at time step t	kWh/h
D_t^{dhw}	Domestic hot water demand at time step t	kWh/h
T_t	Outdoor temperature at time step t	°C
Y_t^{pv}	Specific production of PV at time step t	kW/kWh
Control Parameters		
$\overline{X}^{imp}$	Maximum grid import capacity	kW
$\underline{X}^{exp}$	Maximum grid export capacity	kW
G_e	CO_2-factor for energy carrier e	gCO_2eq/kWh
G_{ref}	Yearly emissions reference (consider cutting)	gCO_2eq/yr
PE_e	PE-factor for energy carrier e	kWh_{PE}/kWh
PE_{ref}	Reference emissions (consider cutting)	kWh_{PE}/yr
γ	Relaxation coefficient for ZEB-restriction	$\in (0,1)$
Λ^{ep}	Activation of energy pricing	0/1
Λ^{ps}	Activation of power subscription pricing	0/1
Λ_i	Pre-activation of technology i	0/1
Λ^{imp}	Activation of import	0/1
Λ^{exp}	Activation of export	0/1

Table 3.2.3: Variables for the model.

Strategic Decision Variables		
x_i	Installed capacity for technology i	$kW(kWh)$
δ_i	=1 if technology i is installed	1/0
Operational Decision Variables		
$q^{sh}_{t,ashp}$	Heat generated by the ASHP for SH at time step t	kWh/h
$q^{dhw}_{t,ashp}$	Heat generated by the ASHP for DHW at time step t	kWh/h
$q^{sh}_{t,gshp}$	Heat generated by the GSHP for SH at time step t	kWh/h
$q^{dhw}_{t,gshp}$	Heat generated by the GSHP for DHW at time step t	kWh/h
$q^{sh}_{t,bb}$	Heat generated by the BB for SH at time step t	kWh/h
$q^{dhw}_{t,bb}$	Heat generated by the BB for DHW at time step t	kWh/h
$q^{sh}_{t,eb}$	Heat generated by the EB for SH at time step t	kWh/h
$q^{dhw}_{t,eb}$	Heat generated by the EB for DHW at time step t	kWh/h
$q^{sh}_{t,a2a}$	Heat generated by the A2A for SH at time step t	kWh/h
$q^{sh}_{t,fp}$	Heat generated by the FP for SH at time step t	kWh/h
$q^{sh}_{t,po}$	Heat generated by the PO for SH at time step t	kWh/h
q^{hwt}_{t}	Net heat to hot water tank (HWT) at time step t	kWh/h
q^{hs}_{t}	Net heat to heat storage/accumulator (HS) at time step t	kWh/h
q^{ss}_{t}	Net heat to shallow storage (SS) at time step t	kWh/h
y^{pv}_{t}	Electricity generated by PV at time step t	kWh/h
$y^{sh}_{t,ashp}$	Electricity consumed by the ASHP for SH at time step t	kWh/h
$y^{dhw}_{t,ashp}$	Electricity consumed by the ASHP for DHW at time step t	kWh/h
$y^{sh}_{t,gshp}$	Electricity consumed by the GSHP for SH at time step t	kWh/h
$y^{dhw}_{t,gshp}$	Electricity consumed by the GSHP for DHW at time step t	kWh/h
$y^{sh}_{t,a2a}$	Electricity consumed by the A2A for SH at time step t	kWh/h
$y^{sh}_{t,eb}$	Electricity consumed by the EB for SH at time step t	kWh/h
$y^{dhw}_{t,eb}$	Electricity consumed by the EB for DHW at time step t	kWh/h
$f^{sh}_{t,bb}$	Fuel consumed by bio boiler (BB) for SH at time step t	kWh/h
$f^{dhw}_{t,bb}$	Fuel consumed by bio boiler (BB) for DHW at time step t	kWh/h
$f^{sh}_{t',fp}$	Fuel consumed by fireplace (FP) for SH at time step t'	kWh/h
y^{ch}_{t}	Electricity charged from the battery at time step t	kWh/h
y^{dch}_{t}	Electricity discharged from the battery at time step t	kWh/h
y^{imp}_{t}	Electricity imported from the grid at time step t	kWh/h
y^{exp}_{t}	Electricity exported to the grid at time step t	kWh/h
y^{pty}_{t}	Electricity exceeding subscription at time step t	kWh/h
z^{hwt}_{t}	Energy content of hot water tank (HWT) at time step t	kWh/h
z^{hs}_{t}	Energy content of heat storage/accumulator (HS) at time step t	kWh/h
z^{ss}_{t}	Energy content of BITES shallow storage (SS) at time step t	kWh/h
z^{ds}_{t}	Energy content of BITES deep storage (DS) at time step t	kWh/h
z^{ba}_{t}	Energy content of battery (BA) at time step t	kWh/h
δ^{ch}_{t}	=1 if battery is charging at time step t	1/0
δ^{dch}_{t}	=1 if battery is discharging at time step t	1/0
Functions		
$c_{inv}(s)$	Discounted investment costs of scenario s	€
$c_{run}(s)$	Discounted operational costs of scenario s	€
Objective function		
C_{tot}	Total system cost	€

3.3 Objective Function

For the MILP problem, the objective function is given by:

$$C_{tot} = min\Big(c_{inv}(s) + c_{run}(s)\Big) \tag{3.3.1}$$

where $c_{inv}(s)$ is the investment cost function and $c_{run}(s)$ the operational cost function, given the scenario s. The investment cost function can be written as:

$$c_{inv}(s) = \sum_{i\in\mathcal{I}}(C_i^{spe}x_i + C_i^{fxd}\delta_i \cdot \alpha_i(R, L_i, \gamma_n) \tag{3.3.2}$$

where the final discounting factor, α_i, takes into forced reinvestments and the remaining lifetime of each technology into account:

$$\alpha_i(R, L_i, \gamma_n) = \frac{1-(1+R)^{-(Y_n - L_iK)}}{1-(1+R)^{-L_i}} \cdot \frac{1}{(1+R)^{-L_i}} + \sum_{k=0}^{K-1} \frac{1}{(1+R)^{kL_i}} \tag{3.3.3}$$

The operational costs, c_{run}, is the sum of the cost of operation and the maintenance costs, fuel costs, electricity costs and the grid charge. Activation of either of the options for the grid charge are given by the binary variables Λ^{ps} and Λ^{ep}. In this work, power subscription is chosen the electricity pricing mechanism. The power subscription model is based the work in [40] and [41]. The expression is written as:

$$\begin{aligned} c_{run}(s) = \Bigg(& \sum_{i\in\mathcal{I}}(C_i^{run}C_i^{spe}x_i) + \sum_{t\in\mathcal{T}} y_t^{imp}C_t^{spot}(s) \cdot 1.25 - y_t^{exp}C_t^{spot}(s) + f_t^{imp}C^f \cdot 1.25 \\ + \quad & (12 \cdot C_{ep}^{fxd} + \sum_{t\in\mathcal{T}} y_t^{imp})\Lambda^{ep} + (12 \cdot C_{ps}^{fxd} \cdot \Big(1 + Y_{sub}\Big) + C_{ps}^{pty}\sum_{t\in\mathcal{T}}(y_t^{pty}) + C_{ps}^{spe}\sum_{t\in\mathcal{T}}(y_t^{imp}))\Lambda^{ps}\Bigg) \cdot \lambda(\Upsilon_n, R) \end{aligned} \tag{3.3.4}$$

where λ is the total capitalization factor, which is used to obtain the present value of all yearly running costs for all years Υ_n in the modelling period. Run costs are summed to the end of each year, which is shown by the second fraction in equation 3.3.3:

$$\lambda(\Upsilon, R) = \frac{1-(1+R)^{-\Upsilon}}{R} \cdot \frac{1}{(1+R)^1} \tag{3.3.5}$$

3.4 Constraints

3.4.1 Capacity Constraints

The installed capacity of each technology i is zero if it is not a part of the solution. M is a large number, often called "big M", and δ_i is the binary activation value:

$$x_i \leq \delta_i M \quad \forall i \in \mathcal{I} \tag{3.4.1}$$

A two-sided constraint is imposed on the model to ensure that the installed capacity for each technology is between pre-defined lower and upper bounds:

$$\underline{X_i}\delta_i \leq x_i \leq \overline{X_i}\Lambda_i \tag{3.4.2}$$

3.4.2 ZEB-constraint

In order to ensure zero emission operation of the building energy system, the ZEB-constraint must be enforced. As mentioned in the theory section [reference], either CO_2-factors or PE-factors can be used. In this work, the CO_2-factor will be used:

$$P_{ZEB}\sum_{t\in\mathcal{T}}\Big(y_t^{imp}G_{el} + f_t^{imp}G_f\Big) \leq \sum_{t\in\mathcal{T}} y_t^{exp}G_{el} \tag{3.4.3}$$

where P_{ZEB} is percentage ZEB, with a value 0 giving noZEB, and a value of 1 signifying ZEB. On the left side of the inequality, the emissions caused by production of imported electricity and the burning of fuel is summed over the operating year. On the right side, the same is done with the emissions displaced by the export of self-generated electricity.

3.4.3 Technology Constraints

The energy produced by on-site technologies is limited by their capacities. For the heat producing technologies, the following must hold:

$$q^{sh}_{t,i} + q^{dhw}_{t,i} \leq x_i \quad \forall t \in \mathcal{T}, i \in \mathcal{I} \tag{3.4.4}$$

where $q^{sh}_{i,t}$ and $q^{dhw}_{i,t}$ is the heat produced by technology i at timestep t for space heating and domestic hot water, respectively. For the electricity produced by the PV-panels, the following constraint must be satisfied:

$$y^{pv}_t = x^{pv} Y^{pv}_t \Lambda_{pv} \quad \forall t \in \mathcal{T} \tag{3.4.5}$$

where Y^{pv}_t[kWh/kWp] is the specific PV-production, x^{pv} [kW] the invested capacity, given by the solar irradiation and the equations for PV-panels ,2.3.1 and 2.3.2.

For the heat pumps, the COP must be taken into account. Since the heating loads are separated, one constraint for each purpose is needed for the waterborne heat pumps:

$$q^{sh}_{t,ashp} = y^{sh}_{t,ashp} COP^{sh}_{t,ashp} \Lambda_{ashp} \quad \forall t \in \mathcal{T} \tag{3.4.6}$$

$$q^{dhw}_{t,ashp} = y^{dhw}_{t,ashp} COP^{dhw}_{t,ashp} \Lambda_{ashp} \quad \forall t \in \mathcal{T} \tag{3.4.7}$$

$$q^{sh}_{t,gshp} = y^{sh}_{t,gshp} COP^{sh}_{t,gshp} \Lambda_{gshp} \quad \forall t \in \mathcal{T} \tag{3.4.8}$$

$$q^{dhw}_{t,gshp} = y^{dhw}_{t,gshp} COP^{dhw}_{t,gshp} \Lambda_{gshp} \quad \forall t \in \mathcal{T} \tag{3.4.9}$$

where $COP^{sh}_{t,i}$, $COP^{dhw}_{t,i}$ is the COP for space and water heating of heat pump i at time step t, respectively. Similarly, $y^{sh}_{t,i}$ and $y^{dhw}_{t,i}$ is the feed-in electricity of heat pump i at timestep t for these same purposes. Because the air-to-air heat pump can only operate on the SH-load, one constraint suffices for this technology:

$$q^{sh}_{t,a2a} = y^{sh}_{t,a2a} COP_{t,a2a} \Lambda_{a2a} \quad \forall t \in \mathcal{T} \tag{3.4.10}$$

where $COP_{t,a2a}$ is and $y_{t,a2a}$ is the COP and the feed-in electricity of the heat pump at timestep t, respectively. In addition, an extra restriction is put on the air-to-air heat pump, as it is assumed to have only one indoor unit, with a limited ability to transport heat to other rooms:

$$q^{sh}_{t,a2a} \leq 0.4 \cdot D^{sh}_t \quad \forall t \in \mathcal{T} \tag{3.4.11}$$

where D_{sh} is the space heating load. Similarly, the production of heat from the electric boiler depends directly on the electricity consumption:

$$q^{sh}_{t,eb} = y^{sh}_{t,eb} \eta_{eb} \Lambda_{eb} \quad \forall t \in \mathcal{T} \tag{3.4.12}$$

$$q^{dhw}_{t,eb} = y^{dhw}_{t,eb} \eta_{eb} \Lambda_{eb} \quad \forall t \in \mathcal{T} \tag{3.4.13}$$

For the point-source system, the electric boiler can only operate on the DHW-load. The electric radiator is introduced as peak load technology for the point-source system:

$$q^{sh}_{t,po} = y^{sh}_{t,po} \eta_{po} \Lambda_{po} \quad \forall t \in \mathcal{T} \tag{3.4.14}$$

For the fuel-driven technologies, heat production depends on the amount of fuel imported. For the bio boiler:

$$q^{sh}_{t,bb} = f^{sh}_{t,bb} \eta_{bb} \Lambda_{bb} \quad \forall t \in \mathcal{T} \tag{3.4.15}$$

$$q_{t,bb}^{dhw} = f_{t,bb}^{dhw}\eta_{bb}\Lambda_{bb} \quad \forall t \in \mathcal{T} \tag{3.4.16}$$

For the fireplace:

$$q_{t',fp}^{sh} = f_{t',fp}^{sh}\eta_{fp}\Lambda_{fp} \quad \forall t' \in \mathcal{T}' \tag{3.4.17}$$

$$q_{t',fp}^{sh} = 0 \quad \forall t' \notin \mathcal{T}' \tag{3.4.18}$$

where $t' = t - floor(\frac{t}{24}) \cdot 24$ and $\mathcal{T}' = \{16, 17, ..., 24\}$, ensuring that it can only operate between 16:00 and 24:00.

3.4.4 Storage Constraints

Heat Storage

Energy content in the storage technologies BA, HWT, SS and DS must be less than or eqaul to the invested storage capacities:

$$z_t^i \leq x_i\Lambda_i \quad \forall i \in \mathcal{I}^z \quad \forall t \in \mathcal{T} \tag{3.4.19}$$

For the hot water tank (HWT), a charging variable is defined, ensuring that the first law of thermodynamics is obeyed. This variable corresponds corresponds to both the charging and discharging variables in equation 2.5.6. The ambient loss factor from 2.5.6 is left out:

$$q_t^{hwt} = z_{t-1}^{hwt} - z_t^{hwt} \quad \forall t \in \mathcal{T} \tag{3.4.20}$$

where $q^{wt} \leq 0$ denotes charging of the storage. The charging rate is limited by the following constraint:

$$\left|q_t^{hwt}\right| \leq x^{hwt}\beta_{hwt} \quad \forall t \in \mathcal{T} \tag{3.4.21}$$

where β_{hs} is the charging rate of the storage. The constraints for the accumulator tank (HS) are defined in exactly the same manner. From a physical standpoint, the temperature difference in this tank is much smaller, as the supply temperature for the space heating usually is around 30 °C. Thus, for the same energy content, the volume of the tank has to be bigger. However, this is outside the scope of this work.

For the Building Internal Thermal Energy Storage, or BITES, the modelling is done in more detail (see 2.5.2). This thermal storage is modelled as a two-node network, a shallow part (abbreviation SS); representing the heating system, indoor air and surface layer of the walls, and a deep part (abbreviation DS), corresponding to the structural elements of the building [31]. The charging and discharging variables from the equations in 2.5.2 are merged into q^{ss}, with negative values representing charging, and positive values charging. First, the balance for the shallow storage:

$$z_t^{ss} = z_{t-1}^{ss} + q_t^{ss} - Flow - q_{loss}^{ss} \quad \forall t \in \mathcal{T} \tag{3.4.22}$$

where z_t^{ss} is the state at time step t, z_{t-1}^{ss} the state at time step $t-1$, q_t^{ss} the above-mentioned charging variable, $Flow$ the cross-node flow (positive flow means that heat flows to DS), and q_{loss}^{ss} the loss factor. The cross-node flow is defined in the following manner:

$$Flow = \left(\frac{z_t^{ss}}{x^{ss}} - \frac{z_t^{ds}}{x^{ds}}\right) \cdot K_{flow} \quad \forall t \in \mathcal{T} \tag{3.4.23}$$

where x^{ss} and x^{ds} have to be given as parameters of the model, in order to preserve the linearity of the model. The parameter K_{flow} is the heat transfer coefficient, given in kWh/h. The loss factor q_{loss}^{ss} is given by the following equation:

$$q_{loss}^{ss} = z_t^{ss} \cdot (1 - K_{loss}^{ss}) \quad \forall t \in \mathcal{T} \tag{3.4.24}$$

where K_{loss}^{ss} is the loss factor, given by the time constant τ_{ss} of the SS:

$$K_{loss}^{ss} = e^{-\frac{1}{\tau_{ss}}} \tag{3.4.25}$$

The equations for the deep storage are analogous, with the exception that there is no direct charging term, so all the heat entering this storage comes via the cross-node flow:

$$z_t^{ds} = z_{t-1}^{ds} + Flow - q_{loss}^{ds} \quad \forall t \in \mathcal{T} \tag{3.4.26}$$

$$q_{loss}^{ds} = z_t^{ds} \cdot (1 - K_{loss}^{ds}) \quad \forall t \in \mathcal{T} \tag{3.4.27}$$

$$K_{loss}^{ds} = e^{-\frac{1}{\tau_{ds}}} \tag{3.4.28}$$

The charging of the SS is not constrained. The reasoning behind this is that the heating system is incentivized to be as small as possible. For both the waterborne and point-source system, the total size will not go above 7 kW (see table 5.3.1). In addition, the SH- and DHW-load have to be covered at every time step, which puts a natural constraint on the charging.

Battery

For the battery, the following constraint must hold:

$$z_t^{ba} = z_{t-1}^{ba} + y_t^{ch}\eta^{ch} - y_t^{dch}\frac{1}{\eta^{dch}} \quad \forall t \in \mathcal{T} \tag{3.4.29}$$

where the charging and discharging efficiencies η^{ch} and η^{dch} take the charging losses into account. The charging and discharging are also constrained:

$$y_t^{ch} \leq (x^{ba} - z_{t-1}^{ba})\frac{1}{\eta^{ch}}\Lambda^{ba} \quad \forall t \in \mathcal{T} \tag{3.4.30}$$

$$y_t^{dch} \leq z_{t-1}^{ba}\eta^{dch}\Lambda^{ba} \quad \forall t \in \mathcal{T} \tag{3.4.31}$$

In order to ensure the mutual exclusivity of charging and discharging within one timestep, the following logical constraints are introduced:

$$y_t^{ch} \leq \delta^{ch} M \quad \forall t \in \mathcal{T} \tag{3.4.32}$$

$$y_t^{dch} \leq \delta^{dch} M \quad \forall t \in \mathcal{T} \tag{3.4.33}$$

$$\delta_t^{ch} + \delta_t^{dch} \leq 1 \quad \forall t \in \mathcal{T} \tag{3.4.34}$$

The charging rates are restricted through the following equations:

$$y_t^{ch} \leq x^{ba}\beta^{ba} \quad \forall t \in \mathcal{T} \tag{3.4.35}$$

$$y_t^{dch} \leq x^{ba}\beta^{ba} \quad \forall t \in \mathcal{T} \tag{3.4.36}$$

3.4.5 Grid Interaction Constraints

The maximum import of electricity from the grid in timestep t is bounded:

$$y_t^{imp} \leq \overline{X}^{imp}\delta_t^{imp} \quad \forall t \in \mathcal{T} \tag{3.4.37}$$

In the same manner, the maximum export to the grid is restricted:

$$y_t^{exp} \leq \overline{X}^{exp}\delta_t^{exp} \quad \forall t \in \mathcal{T} \tag{3.4.38}$$

The mutual exclusivity of import and export is enforced through the associated price difference. Thus, no constraint is required.

Power Subscription Pricing

Activation of the power subscription tariff model activates the following constraints:

$$y_t^{imp} - Y^{sub} \leq y_t^{imp} \quad \forall t \in \mathcal{T} \tag{3.4.39}$$

$$y_t^{pty} \geq 0 \quad \forall t \in \mathcal{T} \tag{3.4.40}$$

The parameter Y_{sub} is set to 5 kW for all cases in the main results.

3.4.6 Load Constraints

In order to ensure that the electricity and heat demand of the building is met, load constraints needs to be defined. On a general level, the electricity balance can be defined in the following manner:

$$D_t^{el} = y_t^{imp} + y_t^{pv} + y_t^{dch} - y_t^{ch} - \sum_{i \in \mathcal{I}^{dhw}} y_t^i - \sum_{i \in \mathcal{I}^{sh}} y_t^i \quad \forall t \in \mathcal{T} \tag{3.4.41}$$

Here, the electricity consumption of the heat producing technologies are split into two sets, $\mathcal{I}^{dhw}$ and $\mathcal{I}^{sh}$, to facilitate separate balances for these two loads. For the model configuration with a hydronic heating system, these two sets will be identical, as all available energy technologies are based on pressurized water as the energy carrier. For the model configuration with point-source heating, however, these sets will be disjoint, since the technologies available for space heating heat the air directly. For the domestic hot water demand, the following balance can be defined:

$$D_t^{dhw} = z_{t-1}^{hs} - z_t^{hs} + \sum_{i \in \mathcal{I}^{dhw}} q_t^i \quad \forall t \in \mathcal{T} \tag{3.4.42}$$

where the discharging of the heat storage is considered, as well as the sum of contributions from the hot water technologies. For the space heating demand, the following expression is defined:

$$D_t^{sh} = z_{t-1}^{ss} - z_t^{ss} + \sum_{i \in \mathcal{I}^{sh}} q_t^i \quad \forall t \in \mathcal{T} \tag{3.4.43}$$

3.5 Model Permutations

In the previous section, the constraints of both the point-source and waterborne model were presented. Here, the different system configurations, i.e. point-source and waterborne, will be presented separately, with and without BITES. The load constraints will be written out explicitly for each configuration. The model with waterborne heating and no BITES for the SH-load corresponds to the model in [2], with the exception that the the HWT can operate only on the DHW-load. However, there is a possibility to invest in an accumulator tank (HS), which serves as a buffer for the SH-load.

3.5.1 Waterborne heating system, without BITES

This is the model with waterborne heating system without Shallow storage. Balance equations can be written based on the figure and the load equations in 3.4.6. First, the electricity balance (The battery is ignored for simplicity):

$$D_t^{el} = y_t^{imp} + y_t^{pv} - y_{t,eb}^{dhw} - y_{t,eb}^{sh} - y_{t,gshp}^{dhw} - y_{t,gshp}^{sh} - y_{t,ashp}^{dhw} - y_{t,ashp}^{sh} \quad \forall t \in \mathcal{T} \tag{3.5.1}$$

With the exception of the battery being ignored, this is just the sums in equation 3.4.41 written out explicitly with the relevant technologies. The balance for the space heating demand:

$$D_t^{sh} = q_{t,eb}^{sh} + q_{t,gshp}^{sh} + q_{t,ashp}^{sh} + z_{t-1}^{hs} - z_t^{hs} \quad \forall t \in \mathcal{T} \tag{3.5.2}$$

The balance for the hot water demand:

$$D_t^{dhw} = q_{t,eb}^{dhw} + q_{t,gshp}^{dhw} + q_{t,ashp}^{dhw} + z_{t-1}^{hwt} - z_t^{hwt} \quad \forall t \in \mathcal{T} \tag{3.5.3}$$

Now, it is possible to see how much a certain technology contributes to each of these loads. Since the space heating system is waterborne, all the technologies are assumed to be able to operate on both loads, although the heat pumps will do so at different COPs (see section 2.3.2).

3.5.2 Waterborne heating system, with BITES

This is the model with waterborne heating and the added possibility of storing energy in the thermal mass of the building. As was mentioned in 2.5.2, this stored energy can only operate on the SH-load, since there is no way to transfer the energy back into the heating system. The balance equations for D_{dhw} and D_{el} are the same as in 3.5.1. The balance for

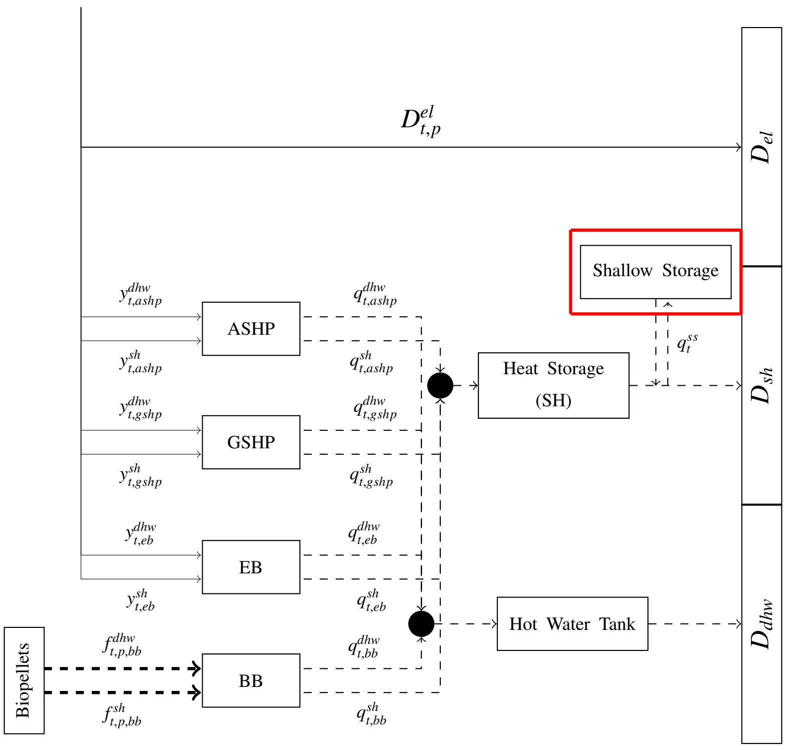

Figure 3.5.1: Waterborne system energy flow.

D_{sh}, however, now has a new element, the heat discharged or charged from the shallow storage. The red line surrounding the shallow storage node denotes the inclusion of this element in the model. Adding this element, a new balance can be written for the space heating:

$$D^{sh}_t = q^{sh}_{t,eb} + q^{sh}_{t,gshp} + q^{sh}_{t,ashp} + z^{hs}_{t-1} - z^{hs}_t + q^{ss}_t \quad \forall t \in \mathcal{T} \tag{3.5.4}$$

where q^{ss}_t is the energy released or absorbed by the building shallow storage. Now the system has the possibility to preheat the building several timesteps ahead of the actual energy use, i.e. at night or in the middle of the day, when the spot price is favorable.

3.5.3 Point-based heating system, without BITES

This is the model with a heating system based on point-heat sources, that is a heating system based on sources such as electric radiators (PO), air-to-air (A2A) heat pumps and traditional fireplaces (FP). Since these technologies transfer heat out into the air (airborne), they cannot operate on the DHW-load. Conversely, the boiler technologies, the electric boiler (EB) and the bio boiler (BB) cannot operate on the SH-load. The energy balances can be written out in a similar manner to what is done in sections 3.5.1 and 3.5.2. The electricity balance:

$$D^{el}_t = y^{imp}_t + y^{pv}_t - y^{dhw}_{t,eb} - y^{sh}_{t,a2a} - y^{sh}_{t,po} \quad \forall t \in \mathcal{T} \tag{3.5.5}$$

The balance for the space heating demand:

$$D^{sh}_t = q^{sh}_{t,p,po} + q^{sh}_{t,a2a} + q^{sh}_{t,fp} \quad \forall t \in \mathcal{T} \tag{3.5.6}$$

The balance for the hot water demand:

$$D^{dhw}_t = q^{dhw}_{t,eb} + z^{hwt}_{t-1} - z^{hwt}_t \quad \forall t \in \mathcal{T} \tag{3.5.7}$$

3.5.4 Point-based heating system, with BITES

This is the model with a point-heat heating system and BITES added as an available storage technology. Again, the red line surrounding the shallow storage denotes the inclusion of this element to the model:

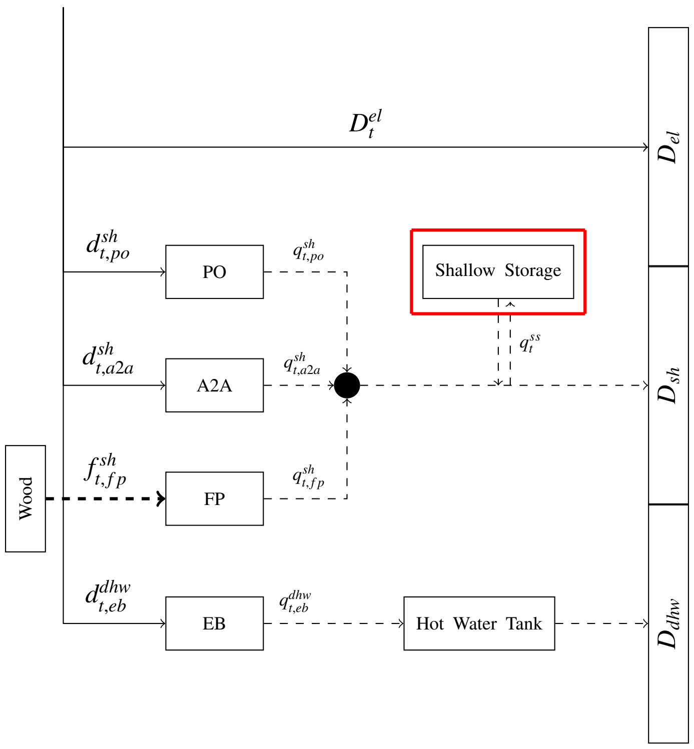

Figure 3.5.2: Point-source system energy flow.

The balance equations for D_{dhw} and D_{el} are the same as in 3.5.1. The balance for D_{sh} takes on an additional element in the same manner as in 3.5.2, to denote the energy which can be released or absorbed by the building thermal mass. Adding the term q_t^{ss}, the balance becomes:

$$D_t^{sh} = q_{t,po}^{sh} + q_{t,a2a}^{sh} + q_{t,fp}^{sh} + q_t^{ss} \quad \forall t \in \mathcal{T} \tag{3.5.8}$$

Chapter 4

Input Data

4.1 Introduction

In this chapter, the input data used in the optimization model will be presented and described. The time series data is generated based on a statistical approach described in [5] and [6], whereas the technology data is based on the reasearch in [2] and various publicly available reports.

Table 4.1.1 below gives an overview of the origin of the data.

Table 4.1.1: List of data and origin.

Parameter	Origin
1.) Electricity data	Based on [6], [4]
2.) Heat load data	Based on [5], [4]
3.) Temperature [°C]	Measurement data, given in [4] [2]
4.) Spot price (€/kWh)	NO1 spot price from Nordpoolspot.no [42]
5.) Exchange rate (NOK/€)	Assumed to be 10 for simplicity
6.) Solar irradiation [kW/m^2]	Irradiation from [43]
7.) PV generation (kWh/kWp)	Given by equations 2.3.1 and 2.3.2 and 6.)
8.) COP heat pumps	Given by equations 2.3.5 and 2.3.6 and 3.)
9.) BITES parameters	Given by [31], [34]
10.) Technology data	Based on regression analysis in [2]

4.2 Technology Data

In this section, the technology data will be presented and discussed. The data is mainly obtained from the Norwegian manufacturers. The prices for the PV, ASHP, GSHP, EB, BB, BA and HS is based on regression analysis using data points from several manufacturers, which yields a linear cost function for each technology. The first component is a fixed cost (given in €, independent of installed capacity), and represents costs such as monitoring equipment, mounting, installation etc. The second component is a specific cost (€/kW), which is dependant upon the installed size of the technology. For a more detailed review of the process, see appendix C in [2]. The third component is the run cost of each technology, given as a percentage of the specific investment cost. In the case of the FP and PO, these assumptions are simply based upon the run costs of the other technologies.

Table 4.2.1: Technology costs.

i (Technologies from [2])	Fixed cost [€]	Specific cost [€/kW]	Run costs[%]	Comment
PV	255	1870	0.01	Fixed costs: Mounting and installation
ASHP	6740	428	0.02	Fixed costs: Mounting and installation
GSHP	11955	961	0.02	Fixed costs: Mounting, installation, well
EB	0	134	0.02	Assumed w/ integrated (HWT)
BB	2221	229	0.03	Fixed costs: pellets storage and feeder
BA	0	707	0.0	Price in €/kWh
HS/HWT	0	83	0.0	Price in €/kWh
i (Technologies added)				
BITES	0	0	0	Assumed free-of-charge
PO	0	100	0	Based on [44]
FP	250	131	0.01	Based on [45]
A2A	570	317.5	0.01	Based on price of Mitsubishi Kaiteki [46]

The technology efficiency is also based on manufacturers' data, and is retrieved from [2]. Lifetimes are estimated from expected lifetimes of the respective technologies and the warranties, given in [39]. Note that the COPs of the heat pumps and the efficiency of the PV are denoted as time-series, COP_t^i and η_t^{PV}. In addition, the COP time-series are split into DHW- and SH-components, as the COP will be different depending on whether the heat pump is used for space heating or domestic hot water purposes.

Table 4.2.2: Technology performance data.

i (Technologies from [2])	L_i [years]	Efficiency	Lower-upper bound	Comment
PV	25	η_t	1-100 kWp	Prices in €/kWp
ASHP	20	$COP_{ashp,t}^{dhw}$, $COP_{ashp,t}^{sh}$	1.5-100 kW	Efficiency depends on temperature
GSHP	20	$COP_{gshp,t}^{dhw}$, $COP_{gshp,t}^{sh}$	1.5-100 kW	Efficiency depends on temperature
EB	20	0.98	0.5-100 kW	
BB	15	0.91	1.5-100 kW	
BA	10	rt = 0.95, β = 0.433	1-100 kWh	β is charging/discharging rate
HS/HWT	20	η = 0.99, β = 0.667	1-100 kWh	β is charging/discharging rate
i (Technologies added)				
BITES	60	Given by equations in 2.5.2	DS=90, SS=12.5	Based on findings in [31], [34]
PO	10	1.00	0-100 kWh	All energy goes towards heating
FP	60	0.84	0-7 kW	Based on [47]
A2A	20	$COP_{a2a,t}^{sh}$	1-100 kW	Only operates on SH-load

Table 4.2.3 below presents the values of the rest of the parameters.

Table 4.2.3: Strategic/control parameters.

Parameter	Value	Comment
C_{ps}^{fxd}	8.61 €/kW	Fixed monthly charge, per kW of subscription [2]
C_{ps}^{spe}	0.05 €/kWh	Variable charge, per kWh of import [2]
C^{pty}	0.1 €/kWh	Penalty charge, per kWh of import exceeding subscription level [2]
Y^{sub}	5 kW	Subscription level [2]
C^{bf}	0.05	Price in €/kWh, from [2]
C^{wo}	0.088	Price in €/kWh, from [48], assuming material humidity of 10%
R	0.06	Discount rate
G_{el}	17 g/kWh	CO_2-factor of electricity, Norwegian energy mix, from [11]
G_{bf}	14 g/kWh	CO_2-factor of biofuel, from [49]
G_{wo}	15.63 g/kWh	CO_2-factor of wood logs, from [50]
Λ^{imp}	1	Import activated for all cases
Λ^{exp}	1	Export activated for all cases
Λ^{ep}	0	Energy pricing deactivated for all cases
Λ^{ps}	1	Power subscription activated for all cases
Λ_i	$1 \ \forall i \in \mathcal{I}$	Except the noBITES-cases, where $\Lambda_i^{ss} = 0$ and $\Lambda_i^{ds} = 0$

Some comments to table 4.2.2 and 4.2.3 are warranted, as they reflect important modelling assumptions:

- The charging/discharging rate β for the HS was modified, since the original value was considered to be too low. The modified value takes into account that the almost all of the thermal energy in the HS can be discharged within an hour [2]. Furthermore, the heat storage is modelled as a fully mixed, uniform (that is, no stratification) storage medium, with no losses to ambient.

- The PV and heat pump efficiencies are time-series, and will be plotted below.

- The battery efficiency is the round-trip efficiency (see equation 2.5 in [2]).

- The lower and upper bounds for the heating technologies is based on the fact that some of them are intended to cover the base load (ASHP, GSHP, BB), and others the peak load (EB).

- The lifetime of all available technologies is fixed; this parameter will in real life depend to on how much it is utilized. For instance, the battery lifetime will depend heavily on number of charging cycles, in addition to the Depth-of-Discharge of each cycle.

- The capacities for BITES is based on assumptions in 2.5.2, and are defined as single-value constrained variables in the model. In addition, the capacities have to be fed in as parameters to the model. In short, it is assumed that the thermal mass of the building can be sufficiently represented by a two-node model, with one node representing the heating system, indoor air, furniture and outer wall layers, and another representing the deeper wall layers.

- The price of biopellets and wood logs is assumed to be constant for the whole modelling period.

4.3 Time Series

Data for five different years are available, based on the work in [4] [5] [6]. However, only data from one year (2012) will be used, as this year represents an average climatic year [4]. The decoupling of the heat load into space heating (SH) and domestic hot water (DHW) components leads to a large number of variables, which causes the branch-and-bound algorithm to converge slowly in many cases (especially with waterborne heating and ZEB, BITES). Thus, a reduced dataset will be used. The data reduction process used is very simple, we simply find the timestep where the highest space heating load occurs, and set this timestep as the midway point of the week selected. If this timestep is near the beginning or end of the year, we let the starting point and/or end point of the winter/fall season be the first/last timestep of the year, respectively. This will overestimate the need for PV-capacity, as the time steps with the most irradiation most likely (mainly during spring and summer) will be removed. As a result, the investment costs, and so the total cost, will be higher.

However, the optimization results can be still be compared across cases, and the most interesting results with regard to operation will in any case stem from peak load situations. The table below shows the results of the simple selection process:

Table 4.3.1: Selected start and stop hours for reduced scenario.

	Start hour	Stop hour	Dates
Winter	742	910	Jan 31. - Feb. 7
Spring	2129	2297	Mar. 29 - Apr. 5
Summer	4313	4481	Jun. 28 - Jul. 5
Fall	8010	8178	Nov. 29 - Dec. 6

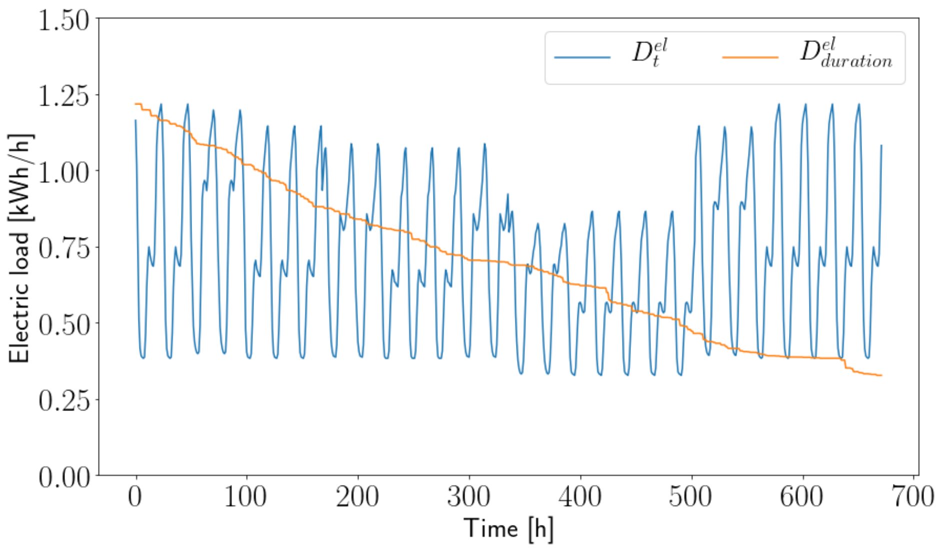

Figure 4.3.1: Electric load for the reduced year (2012).

As can be seen from figure 4.3.1, the electrical load remains fluctuates in a similar manner through the reduced year. This is to be expected, since the electrical load is decoupled from the heating loads. Hence, the load profile shown is the electricity required to supply loads such as kitchen appliances, electronics, lighting etc. However, there is also a clear seasonal trend, which could be ascribed to a decreased need for lighting as the days get brighter. The total electricity demand for the reduced year is 471 kWh, and the peak value is 1.22 kW.

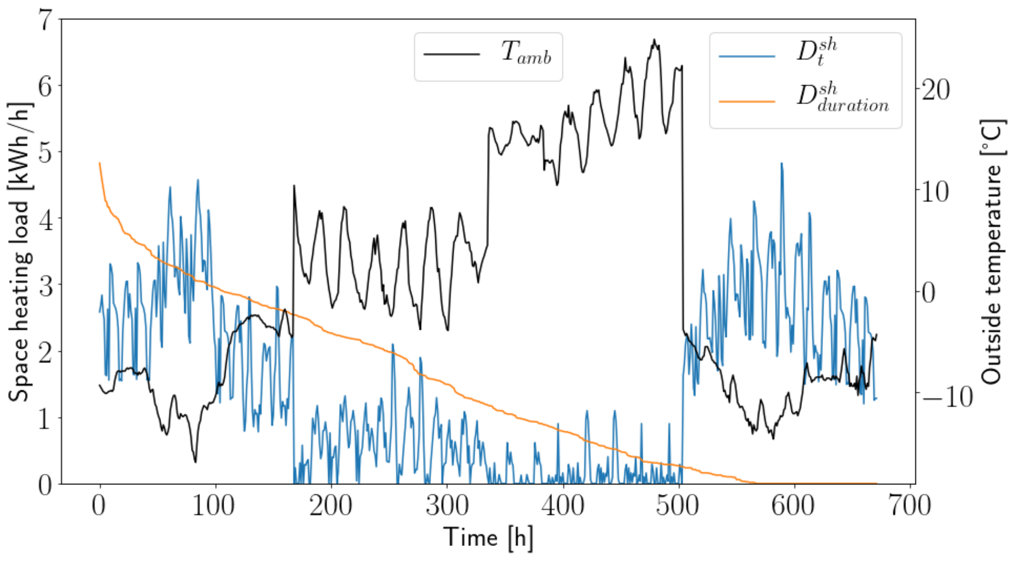

Figure 4.3.2: Space heating load for the reduced year (2012).

Figure 4.3.2 shows the space heating load through the reduced year, its duration curve and the outside temperature. We see that this parameter fluctuates to a much greater degree than the specific electric load, and that it exhibits an inverse relation to the outside temperature. Note that this relation non-linear, as outside temperatures above 15 °C generally leads to a space heating demand of zero [6]. From the duration curve $D_{duration}^{sh}$, we see that the number of hours with no space heating demand is at about 100 (104), which matches well with the number of hours with a temperature above 15 °C (115). Sudden changes in temperature and load mark the seasonal changes. The two winter weeks have similar profiles, while the spring and summer weeks differ more from these and each other. The peak value for D_{sh} is at about 2 kW for the spring week, 1 kW for the summer week, and 4.82 kW for the whole scenario (occurring in the fall week). The total space heating demand is 973.95 kWh.

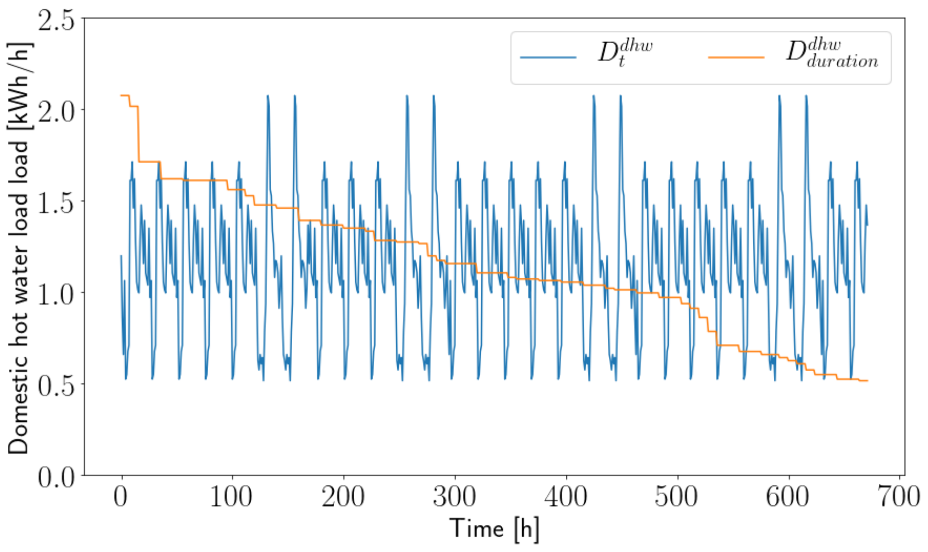

Figure 4.3.3: Domestic hot water load for the reduced year (2012).

In figure 4.3.3, the domestic hot water load is plotted, along with its duration curve. Immediately, we notice this curve is nearly identical for all seasons. The underlying assumption is that the amount of energy used for warm showers in the residence stay the same regardless of the outside temperature. This assumption may not hold in reality, as there may be a tendency to take more frequent and warmer showers when it is cold outside. Another possible deficiency with regard to the accuracy of the curve is the peak load value for the curve, which is at 2.07 kW. Assuming a regular shower-head with a flow rate of 16 l/s, a temperature difference of 30 °C, and heat capacity of water $C_p = 4.184 \frac{kJ}{kgK}$, a 10 minute shower requires 5.58 kWh of energy. If we assume water-saving shower-head, the flow rate, and hence the energy demand, is reduced by 50 %, which is closer to the mark [51].

Another deficiency of the curve is the low number of hours in which D_t^{dhw} is at its minimum, which should correspond to the energy required to keep the hot water tank temperature at its nominal value when energy is drawn from the tank (that is, the energy required to counteract the losses to the ambient air). For a single-home building, one would expect most nighttime hours to contain this value, but this is not the case. Yet another point worth mentioning is the tendency to take showers in the morning or at night, which is not reflected in the curve. In any case, the curve for D_{dhw} is left as is, and can be regarded as the *distributed* domestic hot water load. The total domestic hot water demand is 767.87 kWh.

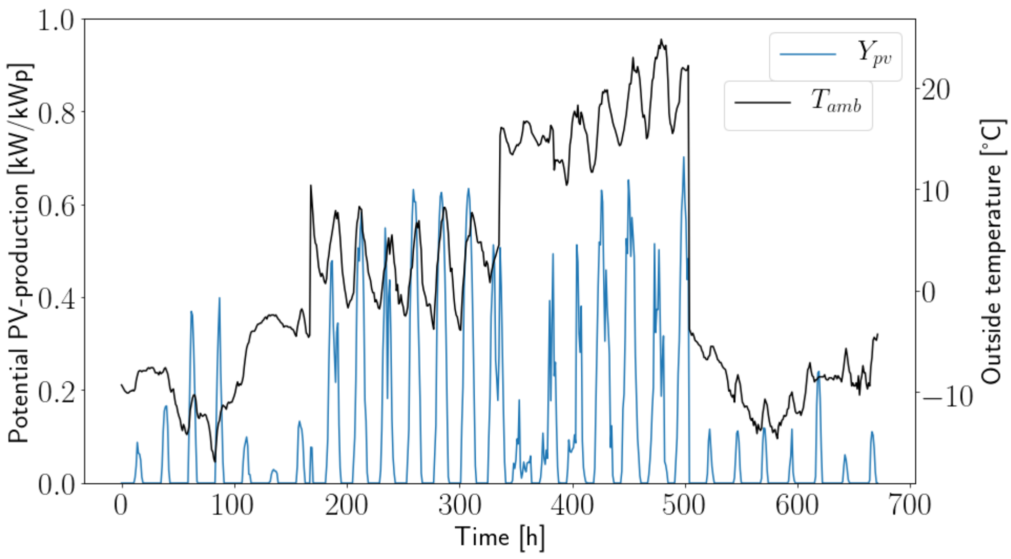

Figure 4.3.4: Potential PV-production for the reduced year (2012).

The PV-production per kW installed is shown in figure 4.3.4. The calculation of these values is based on the equations 2.3.1 and 2.3.2, as well as the following table of coefficients, suggested in [21]:

Table 4.3.2: Coefficient values for calculating potential PV-production.

k_1	k_2	k_3	k_4	k_5	k_6	c
-0.017162	-0.040289	-0.004681	0.000148	0.000169	0.000005	0.05

From the plot, we see that the PV-production is biggest in the spring and summer weeks, with low potential power output in the winter and fall weeks. Additionally, there is a clear daily trend, with intervals of only some hours per day in which the PVs are able to produce power. These intervals increase in length in summer. It is clear that majority of the grid export (y_{imp}) will take place in spring and summer.

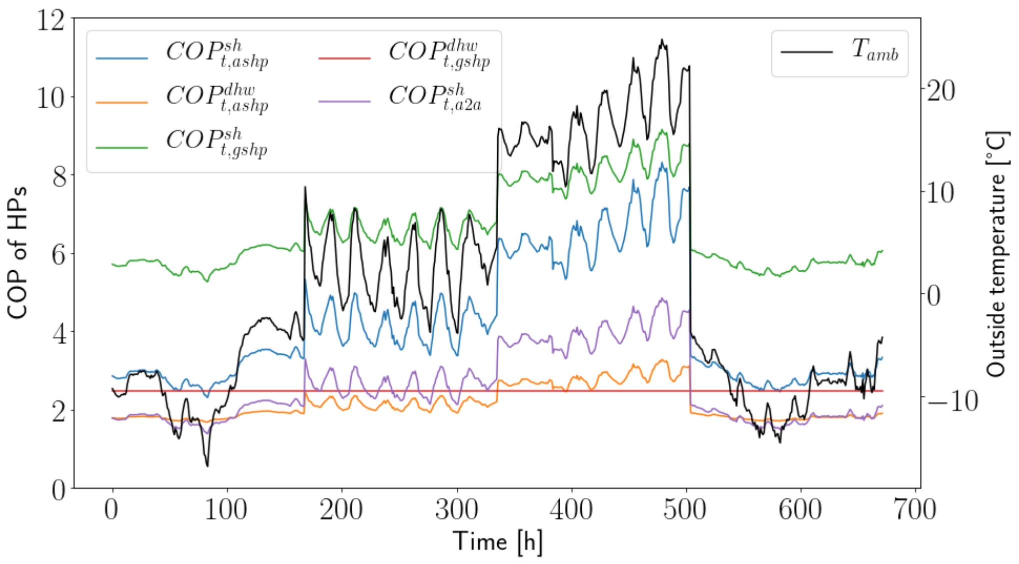

Figure 4.3.5: COPs for the reduced scenario (2012).

Figure 4.3.5 show the time series for the COPs of the heat pumps. The equations 2.3.5 and 2.3.6 are used to calculate the COPs for each timestep and load, in accordance with standard 8 from [4], since the building in question is a passive house (with low supply temperature). With values for A, B and C of -0.0051, -0.5633 and 32.844 respectively, the required supply temperature T_{sup} (for space heating, hydronic system) can be calculated from equation 2.3.7. With this and the following k-values for the waterborne heat pumps, taken from [52]:

Table 4.3.3: Coefficient values for calculating COPs.

	k_0	k_1	k_2
ASHP	7.1299	0.1239	0.0007
GSHP	10.181	0.1839	0.0008
A2A	5.50	0.13	0.0009

the COPs can be calculated. The polynomial for the air-to-air heat pump is based on the table of COPs for different source temperatures in [53], assuming a constant supply temperature of 30 °C. As can be seen from the plot, the COPs for space heating, $COP^{t}_{ashp,t}$ and $COP^{t}_{gshp,t}$, fluctuate the most through the reduced year. They both show a strong dependency on outside temperature T_{amb}, which is to be expected. The large difference COP at low outside temperatures stem from the assumption that the ground water used for the GSHP is at a constant 10 °C throughout the year, yielding relatively stable curve, with a minimum of 5.26 and a maximum of 9.15. A COP of 9.15 is unrealistic, but this will be alleviated by the fact that the space heating demand is low when $COP^{sh}_{gshp,t}$ is high. For the ASHP, the seasonal difference is more pronounced, with a minimum of 2.31 and a maximum of 8.31 for $COP^{sh}_{ashp,t}$. The COPs for domestic hot water use (DHW) is constant for the GSHP, since a constant temperature for the HWT is assumed (65 °C), yielding a constant ΔT of 55 °C. For the ASHP, ΔT is the difference between the required hot water tank temperature and T_{amb}, which gives some variation through the year, but less than for the space heating COP. The COP for the A2A, $COP^{sh}_{t,a2a}$, is at a level below the ASHP and GSHP. The maximum can be seen in summer, at just above 4. This is quite a bit lower than the reported rating for the Mitsubishi Kaiteki 6300, which is claimed by the manufacturer to operate at a COP of 5.52 at nominal conditions (+7 °C) [46]. However, a conservative approach is preferred here.

In order to make the modelling tractable, some assumptions with regard to the time series are necessary:

- The electricity price (that is, the spot price time series) is assumed to be the same for the whole modelling period. These prices will vary a lot from year to year, depending mostly on the amount of rainfall and the outside temperature (assuming electricity is used for heating, which is mostly the case in Norway). Additionally, inflation is not accounted for.

- Related to this is the assumption that the electrical and heat load for the neighborhood is identical every year. In other words, the year chosen (2012) is considered to be an average/representative climatic year. With the way these loads are split up, the heat loads are subject to the most significant variations. The loads remaining for the electricity to supply are household appliances, lighting, electronics etc, so for the specific electrical load, this assumption should be accurate.

Chapter 5

Results and Discussion

5.1 Introduction

In this chapter, the results of the optimization will be presented and discussed. The optimization will be run on eight different system configurations, with the data from the 2012 chosen as the basis for the optimization runs, since this year represents an average climatic year [4]. In addition, because of convergence issues with the branch and bound algorithm, a reduced dataset is used. First, the optimization will be carried out on the system with point-source heating without the ZEB-restriction and without the possibility to use the building's thermal mass as an energy storage. This is referred to as the PH_noZEB-case. Now, the ZEB-restriction is added, to see how the system changes when it is forced to maintain net zero emission operation. It is expected that adding the ZEB-restriction will force significant investments in PV-panels, as this is the only on-site production technology included in the model. This case is denoted as PH_ZEB. The same is done for the waterborne system, with the cases denoted as WB_noZEB and WB_ZEB. At the end of that section, the operation and cost of the systems are compared.

Then, the building internal thermal energy storage is added as an available storage technology, at no cost. The same process is repeated, with the point-source system being treated first (cases PH_noZEB-B and PH_ZEB-B), and the waterborne system thereafter (WB_noZEB-B and WB_ZEB-B).

Finally, a sensitivity analysis with respect to the BITES-parameters is carried out. Only one case will be selected for sensitivity analysis, as it is unfeasible (and most likely unnecessary) to carry it out on all cases. The most obvious parameter to select for such an analysis is the size of the shallow storage (SS), or the degree to which the building is allowed to overheat. For instance, what happens to the cost reduction increase as this parameter is allowed to increase? And how does it affect the dimensioning of the heating system? Furthermore, we have the deep storage (DS), or more intuitively, the "heaviness" of the building construction. Another parameter that lends itself to analysis is K_{flow}, or the tendency of the heat to flow from the shallow storage to the deep storage.

There are eight different cases, four for each system. The cases are demarcated in the table below.

Table 5.1.1: List of abbreviations for the different cases.

	noBITES		BITES	
	noZEB	ZEB	noZEB	ZEB
Point-source	PH_noZEB	PH_ZEB	PH_noZEB-b	PH_ZEB-b
Waterborne	WB_noZEB	WB_ZEB	WB_noZEB-b	WB_ZEB-b

5.2 Abbreviations

A list of essential abbreviations for the technologies is presented below.

ASHP	Air-source heat pump
A2A	Air-to-air heat pump
BA	Battery
BITES	Building Internal Thermal Energy Storage
COP	Coefficient of performance
DHW	Domestic Hot Water
DS	Deep storage of BITES
EB	Electric Boiler
GSHP	Ground-source heat pump
HS	Heat storage (Accumulator tank, SH)
HWT	Hot Water Tank (DHW)
PV	Photo-voltaic panels
SH	Space Heating
SS	Shallow storage of BITES
ZEB	ZEB-constraint activated
noZEB	ZEB-constraint deactivated

5.3 Main Results

Table 5.3.1 shows the invested capacities and costs for all cases. This table will serve as a point of reference when going through the operation of each case, as well as when comparing relevant cases. The heating technologies for the waterborne system are found in the category **SH&DHW**, since these technologies can operate on both DHW- and SH-loads. For the point-source system, the heating technologies are split into two categories, **SH** and **DHW**. The entries that do not apply to a certain case are marked n.a. (not applicable).

Table 5.3.1: Invested capacities and costs for all cases.

	noBITES				BITES			
	Point-source		Waterborne		Point-source		Waterborne	
Constraint	noZEB	ZEB	noZEB	ZEB	noZEB	ZEB	noZEB	ZEB
Case name	PH_noZEB	PH_ZEB	WB_noZEB	WB_ZEB	PH_noZEB-b	PH_ZEB-b	WB_noZEB-b	WB_ZEB-b
Electricity								
PV [kW]	0	31.35	0	14.67	0	31.59	0	14.54
Import [kWh]	1947.54	1357.02	1178.86	630.06	1972.14	1234.97	1166.59	576.45
Peak imp. [kW]	6.50	6.01	4.23	2.79	5.00	5.00	3.28	2.73
SH&DHW								
ASHP [kW]	n.a.	n.a.	4.66	0	n.a.	n.a.	4.49	0
GSHP [kW]	n.a.	n.a.	0	4.71	n.a.	n.a.	0	4.26
BB [kW]	n.a.	n.a.	0	0	n.a.	n.a.	0	0
EB [kW]	n.a.	n.a.	1.31	1.27	n.a.	n.a.	0	0
SH								
A2A [kW]	1.30	1.44	n.a.	n.a.	1.17	1.37	n.a.	n.a.
PO [kW]	3.52	3.38	n.a.	n.a.	2.52	2.65	n.a.	n.a.
FP [kW]	1.58	1.38	n.a.	n.a.	1.24	0	n.a.	n.a.
DHW								
EB [kW]	2.07	2.07	n.a.	n.a.	1.88	2.40	n.a.	n.a.
Storage								
HWT [kWh]	0	0	0	0	5.22	9.60	2.42	5.66
BAT [kWh]	0	0	0	0	0	0	0	0
SS [kWh]	n.a.	n.a.	n.a.	n.a.	12.5	12.5	12.5	12.5
HS [kWh]	n.a.	n.a.	0	0	n.a.	n.a.	1.00	1.00
Total cost [€]	56,386	119,937	39,666	75,808	54,651	117,201	38,559	74,080
Inv. cost [€]	3778	78,538	12,555	79,350	3923	79,350	12,607	58,803
Op. cost [€]	52,590	41,399	27,111	17,349	50,728	37,851	25,952	15,997

5.3.1 Point-source system, without Building Internal Thermal Energy Storage (BITES)

First, the operation of the system with point-source heating will be investigated. The operation of the SH-, DHW-, and electric loads will plotted and examined. The hours selected for plotting are hours 8 through 80 of the reduced scenario, i.e. 06:00 February 1st to 06:00 February 4th.

without ZEB-restriction (PH_noZEB)

As can be seen from 5.3.2, the only technology selected technology for the servicing of the DHW-load is the electric boiler, which is not very surprising, since it is the only technology in the input that can operate on the DHW-load. What is more surprising, however, is the fact that no heat storage is selected. This means that the cost of the heat storage is not covered by the savings that could be made by importing electricity to heat up the water in the tank when the electricity price is low. Clearly, this is an unrealistic situation, as it is necessary to have some kind of storage for the water to be used for hot water purposes. It reflects a modelling error that arises because of the hourly timestep and the optionality of the heat storage. On the other hand, it may be assumed that the EB is delivered with an integrated hot water tank (HWT) (see table 4.2.1 in section 4.2). With this assumption, the size of the HWT will be the installed capacity of the hot water tank *above* the peak DHW-load.

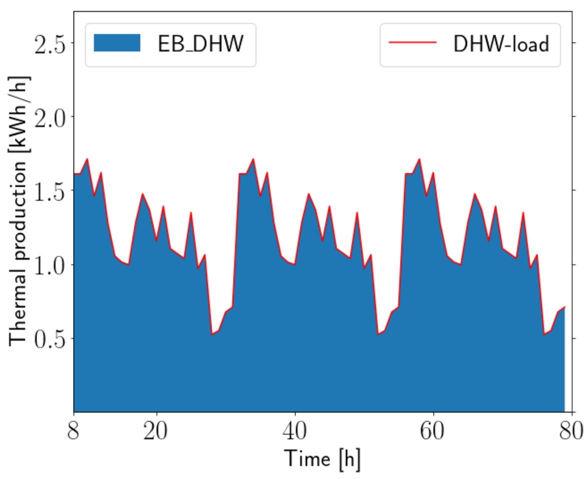

Figure 5.3.1: DHW-balance (PH_noZEB).

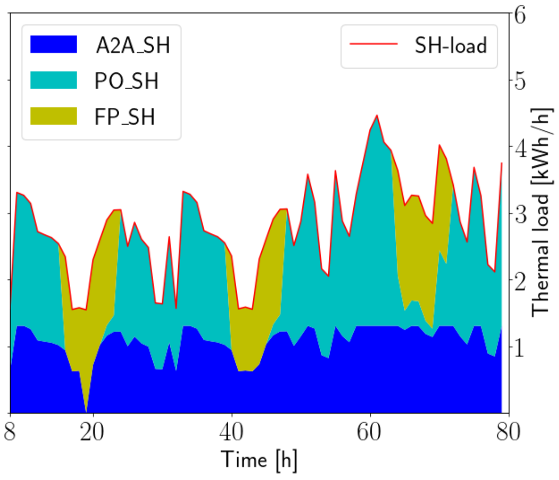

Figure 5.3.2: SH-balance (PH_noZEB).

For the SH-load, all available technologies are selected. The air-to-air heat pump is limited to covering 40 percent of the load at all timesteps 3.4.11, leading to a considerable investment in electric radiators, a cheap but (compared to heat pumps) energy inefficient technology. The invested capacities are 3.52 and 1.30 kW, respectively. In addition, the decision is made to invest in a fireplace with a capacity of 1.58 kW. There is an inherent problem in modelling the cost of this technology as linear, since the fireplace used as basis for the cost structure in this work has a capacity in the range of 2-7 kW [47], with the real power output depending on the amount of wood burning, which is hard to control manually. Thus, an FP capacity of 1.58 kW is in reality an unrealistic size. Perhaps it would make more sense to model it as a binary decision. At least, a lower bound of 2-4 kW should be used in the future. Another challenging aspect of the fireplace modelling is the price of wood, as this is a fuel many people can acquire for free/at a very low cost.

In figure 5.3.2, three days of the SH-balance is shown. It can be readily seen that the air-to-air heat pump functions as the base load, while the electric radiators are used to a varying degree to cover the peak load, alternating between being completely shut off and operating at close to their max capacity of 3.5 kW. The fireplace is only active between the hours of 16:00 and 24:00 each day; this is a modelling restriction put in to reflect the fact that it needs manual refilling of wood in order to maintain the power output. In these hours, the radiators are shut off, indicating that the electricity price in these hours is high compared to the price of wood logs.

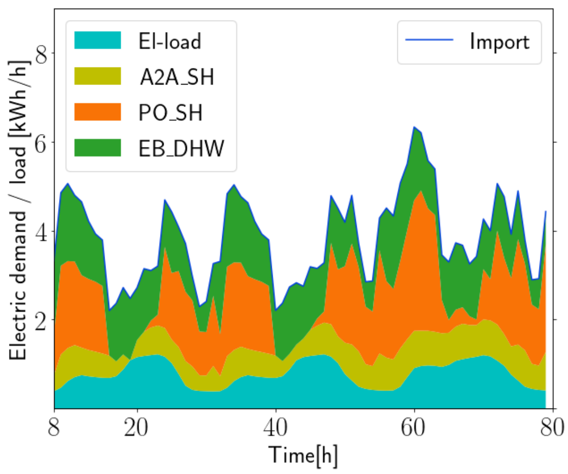

Figure 5.3.3: Total electric balance (PH_noZEB).

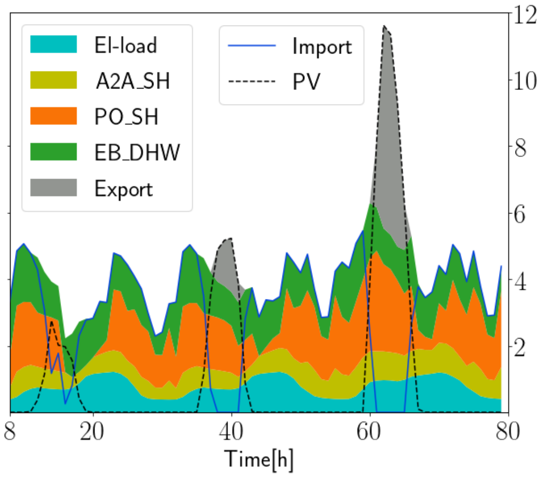

Figure 5.3.4: Total electric balance (PH_ZEB).

Figure shows the operation of the electric load. The lowermost area of the graph is the specific electric load, D_{el}, which is the electricity required to power lights, kitchen appliances, electronics etc. With a point-source heating system, we see that this is a relatively small part of the total electric load. The sum of the green, orange and yellow areas constitute the *thermal* electric load (mentioned in 2.4), which is the electricity required to for heating purposes (SH and DHW). The sum of this thermal electric load and the specific electric load is the *total* electric load. The electricity needed to power the A2A heat pump, y^{sh}_{a2a}, is also a relatively minor part of the total load, since it is inversely proportional to the COP at any given time step. When the PO is needed for load coverage, which primarily happens when the FP is inactive, the total electric load rises significantly. The electricity needed for the EB to heat up water, y^{dhw}_{eb}, follows the plot in figure 5.3.1 closely, since the EB has an efficiency of 0.98. Since there is no incentive to reduce net CO_2-emissions (through the ZEB-restriction), no PV-panels are invested in, and all of the electricity is imported.

With ZEB-restriction (PH_ZEB)

When the ZEB-restriction is added (that is, when the system is forced to compensate for its electricity import and fuel combustion), the operation and dimensioning of the heating system does not change much compared to the previous section (see table 5.3.1). Hence, the plots for the operation of the SH- and DHW-loads are not shown. The most significant difference overall is the investment in PV-panels, which becomes necessary for the system to maintain the ZEB-restriction. Figure 5.3.4 shows the total electric load for three winter days. During these three days, there are three intervals in which the import of electricity, y_{imp}, either decreases markedly or goes all the way down to zero. In these intervals, the PV-panels are producing power. When the production is at a certain level, the system ceases to import electricity entirely, and exports instead. The gray area in the plot represents this export. Because the plot shows the operation in three *winter* days, the PV-panels are not close to producing at their rated capacity of 31 kW. The thermal electric load is slightly smaller than in the previous case (PH_noZEB), since the 0.14 kW is added to the A2A capacity, and 0.14 kW is subtracted from the POs. (However, this does not represent any meaningful change in size). Adding the ZEB-restriction nearly doubles the total discounted system cost, most of which comes from the PV-investment (see table 5.3.1). This can be viewed as the price of ZEB-operation. On the other hand, the operational costs go down from 52,950€ to 41,399€, which is caused by the self-reliance and export mentioned previously. This also causes the total import of electricity and peak load to decrease, which is positive for the grid impact. Note that neither the HWT nor the battery is invested in.

5.3.2 Waterborne system, without Building Internal Thermal Energy Storage (BITES)

Now, the dimensioning and operation of the waterborne system will be investigated. A key assumption when comparing these two cases later is that the in-house distribution system (i.e. pipes and radiators/underfloor heating) come at no cost. This important to keep in mind when the cost of the two main configurations are compared later.

no ZEB-restriction (WB_noZEB)

Figure 5.3.5 shows the operation of the DHW-load for the same three winter days as in the previous section. The base load is covered by the ASHP, which has a capacity of 4.66 kW (see table 5.3.1). Again, it is assumed that the base load technology comes with an integrated HWT large enough to cover the peak DHW-load (2.07 kWh). In certain intervals, the EB kicks in for peak load support. These intervals coincide with the hours in which the SH-load is high, leading to a reduced capacity for the ASHP to contribute towards water heating. In figure 5.3.6, the operation of the SH-load can be seen. The ASHP covers the entire SH-load for the three days shown, and this is the case in general as well. This is because the COP for space heating is higher than for water heating, since there difference between supply and source temperature is smaller. Thus, it will be more optimal to prioritize using the ASHP for space heating, as the relative difference between the performance of the ASHP and the EB is smaller for water heating.

The electric load is shown in figure 5.3.7. Immediately, we notice that the thermal electric load is significantly smaller than for the previous cases, PH_noZEB and PH_ZEB. During the three days of operation shown in the plot, the total electric load only goes above 3 kWh/h on a couple of occasions on the last day, where a high DHW-load coincides with a peak in the SH-load. The peak total electric load is 4.23 kWh/h, compared to 6.01 kWh/h and 6.50 kWh/h for PH_noZEB and PH_ZEB, respectively (see table 5.3.1). The reason for this discrepancy is the large difference in COP/efficiency between the waterborne heat pumps and the heating technologies for the airborne system, in addition to the modelling assumption stating that the A2A heat pump can only cover 40 % of the SH-load at any given time (heat pump in one room, no air ducts for heat transportation, one indoor unit). If this assumption/constraint were to be relaxed, the discrepancy would be smaller, but would still persist, as the A2A heat pump cannot operate on the DHW-load.

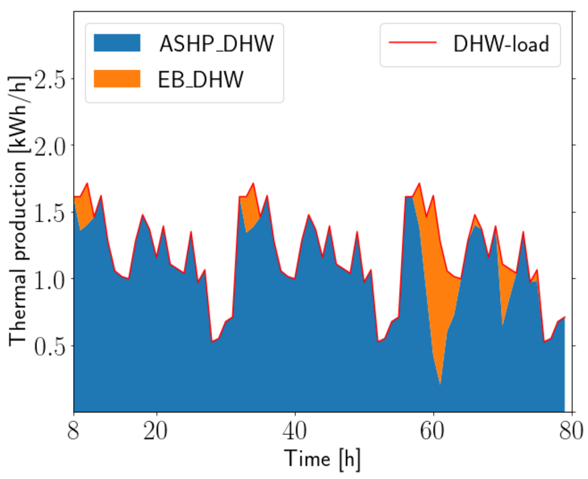

Figure 5.3.5: DHW-balance (WB_noZEB).

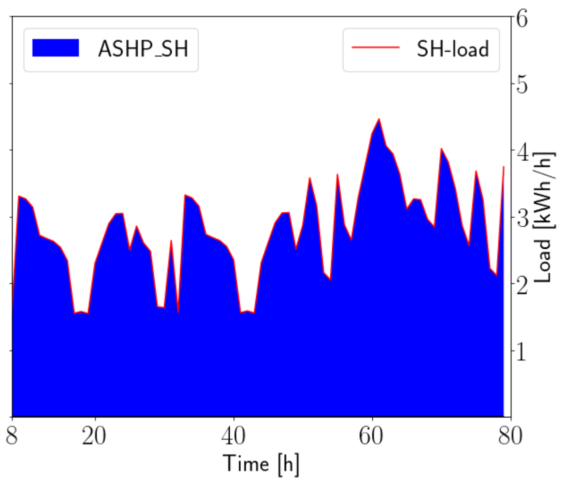

Figure 5.3.6: SH-balance (WB_noZEB).

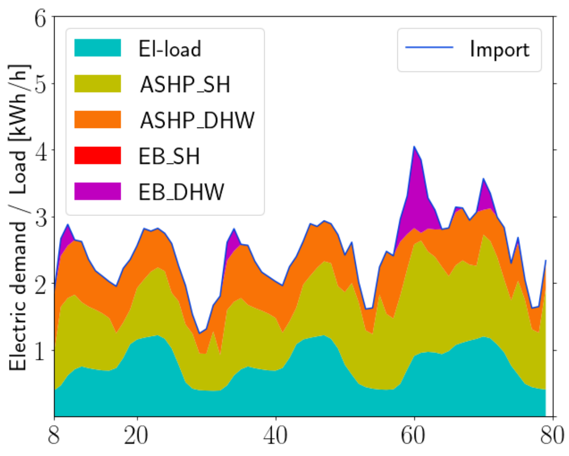

Figure 5.3.7: Total electric balance (WB_noZEB).

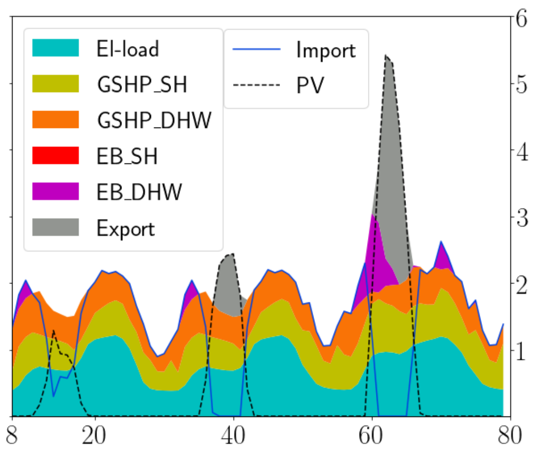

Figure 5.3.8: Total electric balance (WB_ZEB).

with ZEB-restriction (WB_ZEB)

Again, when the ZEB-restriction is added, the PV-panels are invested in. This ensures a net emission of zero during the building's lifetime with the selected CO_2-factor for electricity, with export of electricity when the PV-panels are producing compensating for the import. In addition, the base load technology changes from the ASHP to the GSHP. This occurs because a premium is put on keeping the electricity import low, and since the GSHP operates with a higher COP, it is the preferred choice, despite higher investment costs. The difference in COP between the two is most pronounced for space heating, and especially during cold winter days , as the GSHP is assumed to operate with a constant source temperature of 10 °C, while the heat source of the ASHP is the ambient air. The difference can easily be seen when comparing the green areas of figures 5.3.7 and 5.3.8. The peak total electric load during the three days shown drops by close to 1 kWh/h, which is positive in terms of grid impact. The EB remains at about the same size, and again, none of the energy storages are invested in. The bio boiler is not selected in any of the two cases. This could be because of the relatively small CO_2-factor used (Norwegian el-mix, 17 g/kWh). If it were changed to the suggested EU-value (134 g/kWh), this might change.

Discussion

Again, the total discounted system cost almost doubles when adding the ZEB-restriction, from 39,666€ to 75,808€. The cost structure, however, is different from the point-source cases. For PH_noZEB, the investment costs are 3778€, while they are 12,555€ for the WB_noZEB. This is made up for in full by the greatly reduced electricity demand, as the operational costs are almost cut in half for WB_noZEB, 27,111€ vs. 52,590€ for PH_noZEB. The effects of this reduced electricity demand are carried over to the ZEB-cases, where it can be seen that the required size of the PV-panels in order to remain carbon-neutral is more than halved, from 31.35 kW to 14.67 kW. The decrease in required PV-size is compounded by the switch from the ASHP to the GSHP; as already mentioned, this technology generally operates with a higher COP than the ASHP. This leads to the investment costs for WB_ZEB actually becoming 20,000€ less than for PH_ZEB. Additionally, the operational costs for WB_ZEB are less than half of the costs for PH_ZEB, at 17,349€ vs. 41,399€.

As for the grid impact, some remarks have already been made when going through the plots for the operation of the electric load. Here, a summary of this aspect will be attempted. The total grid import is by far highest for PH_noZEB, at 1947.54 kWh. When the ZEB-restriction is added, the import is reduced to 1357.02 kWh. while the peak load goes down from 6.50 kW to 6.01 kW, mainly because of the self-generated electricity; a consequence of the PV investment. For the waterborne cases, the total import goes down from 1178.86 kWh to 630.06 kWh, i.e. a reduction by almost 50 %. The peak import is reduced from 4.23 kW to 2.79 kW; which is result of both PV-investment and switching the base load technology from ASHP to GSHP. The grid impact of the respective systems differ vastly, not only because of the above-mentioned figures, but also because the *export* to the grid also is reduced. This means that the grid *burden* is reduced; freeing more of the transmission line capacity, so that other systems (possibly point-source) can maintain zero net emission operation.

Taking all of the above into consideration, it is evident that the waterborne system is preferred over the point-source system for both noZEB and ZEB operation. However, it is important to keep in mind that the (economical) *magnitude* of this superiority hinges on a few key assumptions. Firstly, it is assumed that the in-house hydronic distribution system for the waterborne system comes at no cost. Including the investment and maintenance cost of these components would push the balance more in favor of the point-source system, at least somewhat. Furthermore, it is assumed that the A2A heat pump can only cover up to 40% of the SH-load at any given timestep. This restriction is put in to reflect the fact that COP of the A2A has a steep temperature gradient as the outside temperature goes below -15 °C, with the COP assumed to be 0.9 at -35 °C. At such a temperature, the efficiency of the A2A is actually lower than that of the PO (electric radiator). Such as the model is, if this restriction were not in place, the heat production of an A2A with a rated capacity of 3 kW at nominal conditions (+7 °C) would still be able to remain at 3 kW at this temperature, even though the required electricity would be above 3 kW ($\frac{3}{0.9}$). This modelling problem could be solved in another way, with gradual temperature restrictions (e.g. for every 10 °C decrease) limiting the amount of heat production from the A2A. Additionally, the restriction can be seen as a way to represent the A2A being located in one (presumably large) room, with no air ducts to transport the heat to other rooms.

5.3.3 Point-source system, with Building Internal Thermal Energy Storage (BITES)

Now, the operation and dimensioning of the energy system when the thermal mass of the building (BITES) is added as a storage technology. The key question is: How does adding BITES alter the systems (i.e. waterborne and point-source), both with respect to investment and operation? The hours selected for plotting are 0 to 72 of the reduced scenario, i.e. Jan 31st 22:00 - Feb 3rd 22:00.

without ZEB-restriction (PH_noZEB-b)

Again, the point-source case will be examined first. Figure 5.3.9 shows the DHW-balance for three winter days. As can be seen from the plot, a heat storage of approximately 5.5 kW is invested in for this model configuration. In the plot, the charging of the heat storage takes place when the area representing the storage heat flow, q_{hs}, is above the line representing the DHW-load, and the discharging takes place when this same area is below the line. The load is covered by the electric boiler and the heat storage operating in tandem, with the electric boiler being shut off intermittently and the charged heat storage delivering hot water to the load. We can assume that the model has found the timesteps in which the spot price is unfavorable, typically in the afternoon, and has allowed the heat storage to charge up fully or partially to cover the load in these hours.

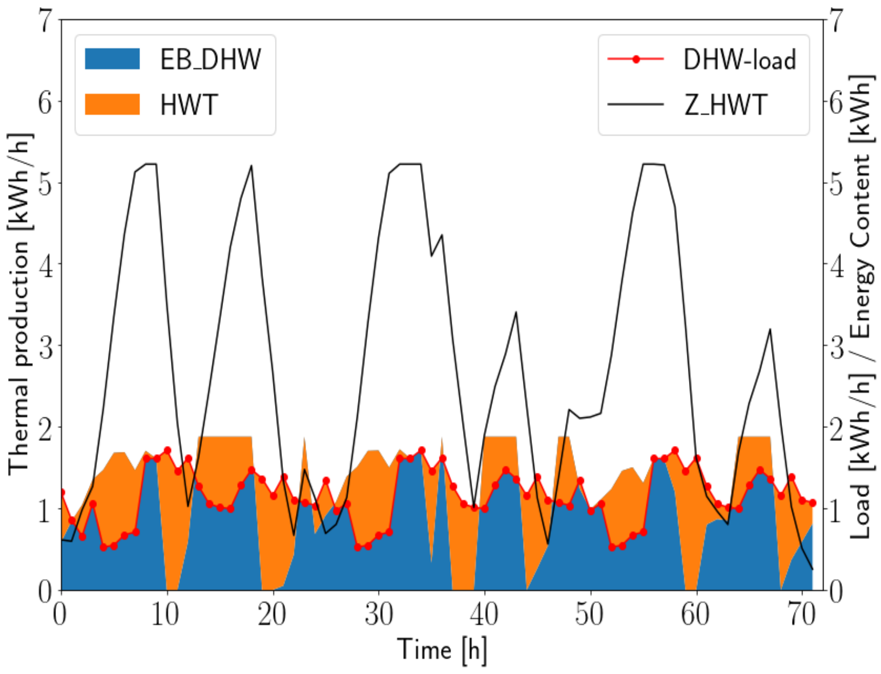

Figure 5.3.9: DHW-balance (PH_noZEB-b).

In figure 5.3.10, we see the operation of the technologies covering the SH-load. Just like in the PH_noZEB and PH_ZEB-cases, all available technologies are selected. The most significant difference with regard to dimensioning from the previous case is that the capacity of the electric radiators is at a capacity of 2.52 kW, compared to about 3.52 and 3.38 from PH_noZEB and PH_ZEB, respectively. The air-to-air heat pump and the fireplace are reduced to 1.17 and 1.24 kW, respectively. This means that the total space heating capacity of the system is reduced by over 1 kW. The reason for this being possible lies in the utilization of the building thermal mass. Assuming a heat capacity of 6.25 $\frac{kWh}{K}$ for the "shallow" storage 2.5.2, that is the outer parts of the walls and the air, furniture etc., and allowing a positive temperature deviation of 2 K (that is, overheating the building), we get a shallow storage of 12.5 kWh. As we can see from the plot, this storage is used to some degree in the servicing of the load. Starting at about the second hour, the building is overheated with the radiators and the air-to-air heat pump, reaching a state of just below 8 kWh at hour 8. Then, the storage is discharged until about hour 16, where the FP and A2A take cover the relatively modest load, with intermittent support from the PO. Since the SH-load is modest, a small charging of the SS takes place, followed by an immediate discharging. From hour 19, the A2A heat pump is shut off for an hour, since the heat from the SS discharging and the FP is sufficient to cover the load. From hour 22, the charging of the SS starts again, reaching a slightly higher energy content than the previous day. Thereafter, the same process repeats itself, with some minor deviations. On the last day shown, the SS does only reaches about 6 kWh, since more of the space heating system is required to cover the instantaneous SH-load, which is higher than in the previous two days. This higher load is brought on by lower outside temperatures.

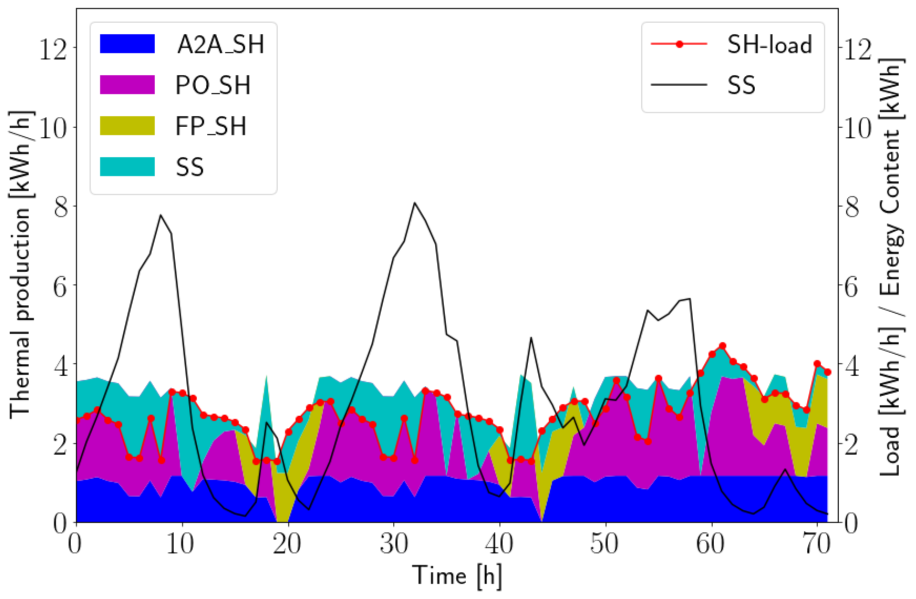

Figure 5.3.10: SH-balance (PH_noZEB-b).

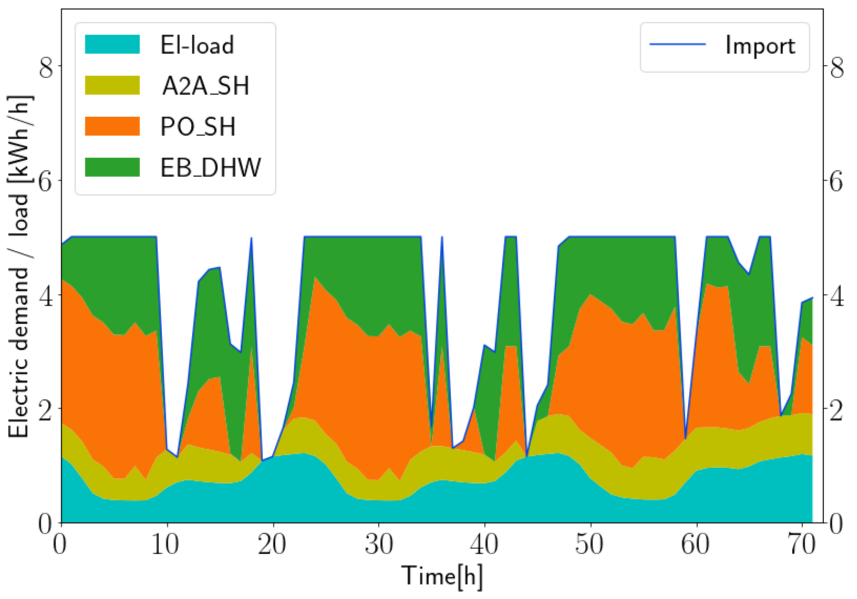

Figure 5.3.11: Total electric balance (PH_noZEB-b).

Figure 5.3.11 shows the total electric load. Comparing this plot to figure 5.3.3, we see that a different pattern emerges. First of all, the relatively curves with a sustained peak of 5 kW can be noticed. These curves correspond to the charging of the SS and HWT at nighttime, with contributions mostly coming from the A2A and PO. Then, after the SS is charged to 8 kWh, the import y_{imp} is effectively throttled to just above 1 kWh/h, since the SH- and DHW-loads are covered by the discharging of the SS and HWT, respectively. Then, since the HWT tank is charged again, the total electric load rises, to about 4.5 kWh/h, followed by a small dip to 3 kWh/h, and then another short peak of 5 kWh/h. This pattern of two sharp declines of the total electric load per day repeats itself the next two days, with the duration naturally being shorter when the load is higher. By considering figures 5.3.9 and 5.3.10, we see that these dips again correspond perfectly to the intervals in which the SS and HWT cover the loads.

with ZEB-restriction (PH_ZEB-b)

Now, the ZEB-constraint is added. Just like when adding the ZEB-restriction in the noBITES-cases, this necessitates an investment in PV-panels. In figure 5.3.12, the DHW-balance is shown. We see that the operational pattern is similar to the case in 5.3.9, with a combination of production from the EB and discharge from the HS covering the load. The major difference between the two cases is the size of the HS, 5.6 kWh to 9.6 kWh for the noZEB and ZEB cases respectively, and hence the charging/discharging patterns. In this case, the duration of the charging cycle is longer, since the HS is bigger, which means that the EB has to produce at full capacity for a longer time interval in order to fully charge the HS. To accommodate this, the size of the EB is increased as well, from 1.88 kW to 2.40 kW. Here, a trade-off between leveraging spot price differences and the added cost of a bigger HS is at play, and it is apparent that the model has found it optimal to invest in a larger buffer against higher spot prices. In addition, since the electricity pricing model used is power subscription (instead of energy pricing), there is a strong incentive to keep the import y_{imp} below 5 kW, as import above this limit will incur extra charges.

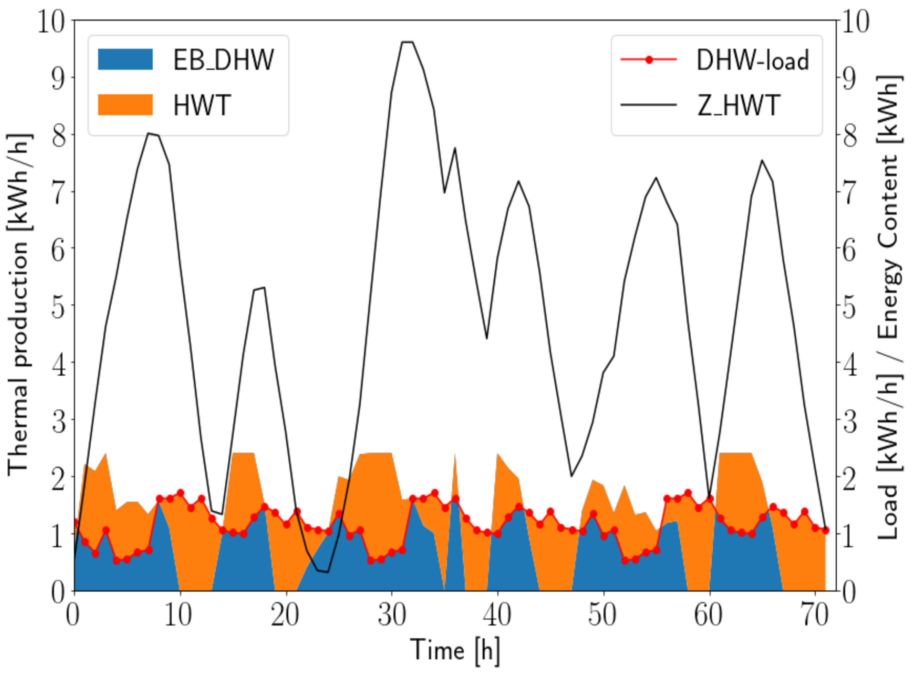

Figure 5.3.12: DHW-balance (PH_ZEB-b).

The SH-load is plotted in figure 5.3.14. Compared to the PH_noZEB-B-case, plotted in 5.3.10, the SS appears to be utilized less in these three days. The reason for this is that the FP has dropped out of the solution, which is a result of it having similar CO_2-factor (15.63 for wood logs vs 17 for electricity) and lower efficiency, in addition to the extra cost (fixed and specific) for the FP investment. Thus, in order to remain below the 5 kW limit PS limit, less heat is available for the SS, especially since the HWT is charged at a higher rate than in the previous case (EB at 2.40 kW). However, this does not mean that BITES is used less through the reduced year. In fact, as can be seen in figure 5.3.13, where the utilization of BITES is plotted for PH_noZEB-B and PH_ZEB-B, the utilization is more frequent for the ZEB-case in the first half of the year, although the energy content rarely goes above 4 kWh.

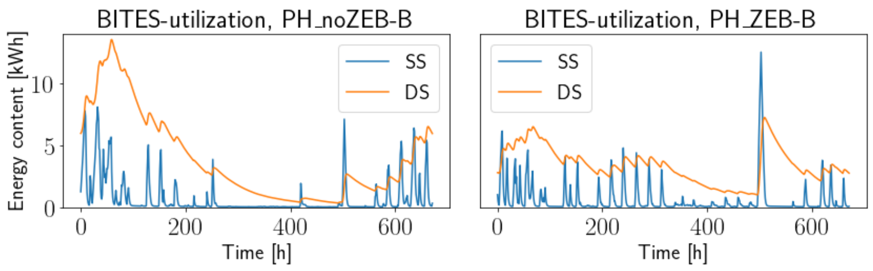

Figure 5.3.13: BITES utilization through the reduced scenario for PH_noZEB-B and PH_ZEB-B.

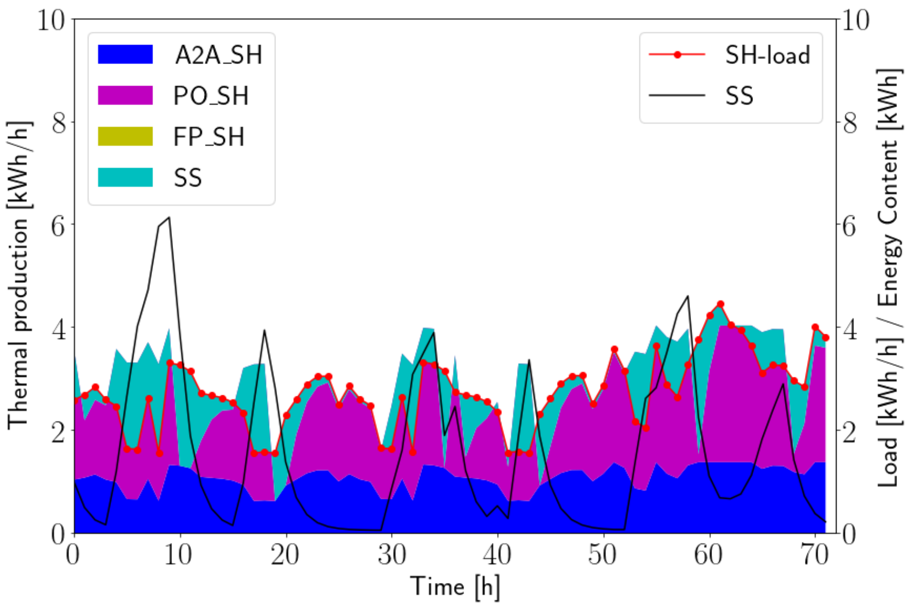

Figure 5.3.14: SH-balance (PH_ZEB-b).

Figure 5.3.11 shows the total electric load. It is similar to the situation in 5.3.11, with one notable exception: when the storages are discharged, at about hour 10 (08:00 Feb 1st), the total electric load does not drop further than just below 2 kWh. Again, this is because the FP has dropped out of the solution, and the A2A and PO still has to supply space heating most of the time. On the other hand, since the HWT is larger, there is a longer interval in which the EB is inactive, so the total electric load stays below 5 kWh/h longer. In addition the PV-production enforced by the ZEB-constraint enables the import y_{imp} to drop down partially or fully down to zero just after the storages (HWT and SS) are discharged.

By looking at table 5.3.1, we can see that the total import goes down from 1972.14 kWh to 1234.97 when adding the ZEB-constraint for the BITES-cases (from PH_noZEB-b to PH_ZEB-b). If we compare the figure for PH_ZEB-b to PH_ZEB, we see that the total import decreases by more than 100 kWh. This is in part because the PO is used less in peak load situations, as BITES functions as a peak load technology. As for the total system cost, it goes down when adding BITES in both the noZEB- and ZEB-cases. In the noZEB-cases, the reduction is 1735€, which equals 3.1 %. For the ZEB-cases, the reduction is 2736€, or 2.3 %. In both situations, the cost reduction achieved through a significant decrease in operational costs, as the investment costs are somewhat higher for both BITES-cases. This is because the reductions in the sizes of the space heating systems when adding BITES is offset by relatively large investments in HWTs. In conclusion, it can be said that adding BITES has a positive effect on the point-source system, as it functions as ”peak load shaver”, reducing the need for PO production when the SH-load is high.

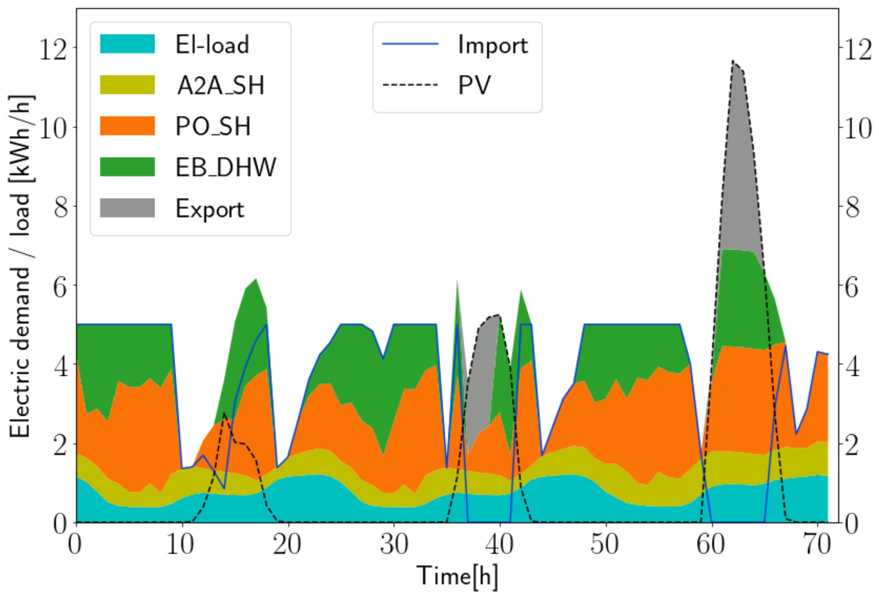

Figure 5.3.15: Total electric balance (PH_ZEB-b).

5.3.4 Waterborne system, with Building Internal Thermal Energy Storage (BITES)

Now, the waterborne system when adding BITES will be investigated. The same procedure as in the previous cases will be followed; first, the operation without the ZEB-constraint will be considered. Thereafter, the ZEB-constraint is added, and eventual changes will be commented and discussed. It is to be expected that adding BITES as an available technology should have a similar effect as with the point-source system. One point of interest is the economical value (that is, how large the cost reduction is) of BITES for the waterborne system. Since a system with heat pumps using water as an energy bearer will be less sensitive to spot prices (through the COP), we can already suspect that this cost reduction will be less than for the point-source cases.

without ZEB-constraint (WB_noZEB-b)

Figure 5.3.16 shows the DHW-load for WB_noZEB-B. Again, we see that the HWT is invested in when BITES is added. Thus, the EB drops out of the solution, and the DHW-load is covered by a combination of ASHP production and discharging of the HWT. However, the capacity of the HWT is smaller than for PH_noZEB-B and PH_ZEB-B, at 2.42 kWh. While it resembles the operation in 5.3.9 and 5.3.12, it has one notable distinction that separates it from the point-source cases. In these cases, the charging of the HWT is usually followed by an immediate discharging, or at least almost immediate (maximum latency of 2-3 hours). In this case, we see on three occasions that the HWT is fully charged, followed by a latency period of some 10 hours. In this latency period, which coincides with a low DHW-load, the SS is charged (see 5.3.17, which then gives way to a simultaneous discharge of both storages. One could say that HWT control has to ”look ahead” to a greater degree than for the point-source cases. The reason for this behaviour is that the ASHP, the selected base load technology, operates on both the SH- and DHW-loads. Since the model is incentivized to keep this capacity as low as possible (through the objective function), this pattern emerges, where the ASHP takes turns charging the SS and the HWT. This, in contrary to the point-source cases, is not a result of a need to limit the import to 5 kW or below (since the high efficiency/COP makes this possible in any case), but rather because the ASHP is limited to producing 4.49 kW at any given time, which is about 1 kW below the total peak heat load (the maximum of the sum of SH and DHW). Hence, the SS and HWT can seen as a direct replacement of the EB, reducing both the operational cost and the total import of electricity, since all of the heat is generated by the ASHP, which has a significantly higher efficiency than the EB. Additionally, a small contribution from the HS (Accumulator tank) can be seen in figure 5.3.17, which is a low-temperature storage tank for space heating purposes. It is dimensioned at 1 kWh, the lower bound set for the water tanks in all cases. From the plot, it can be seen that the HS is at 1 kWh most of time, with a quick discharging followed by an immediate charging when it is needed for peak load support.

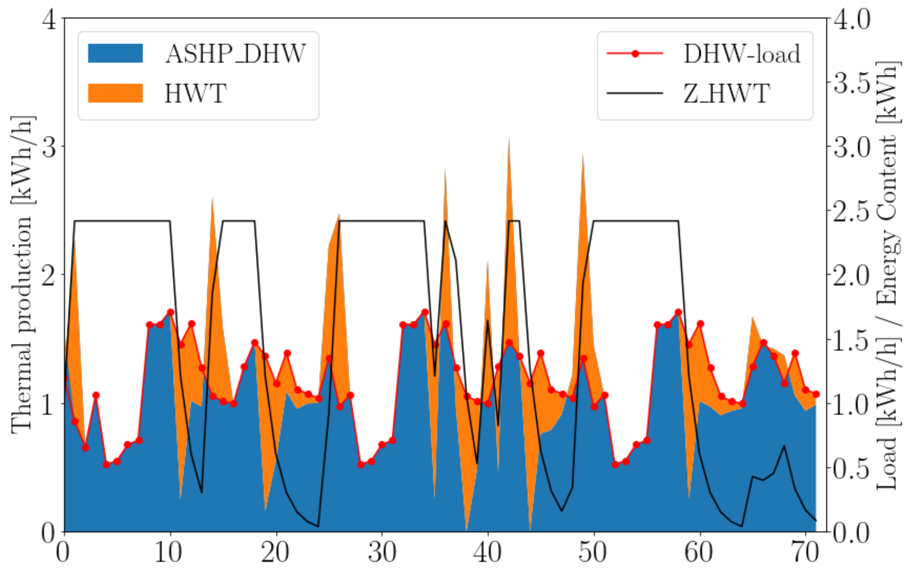

Figure 5.3.16: DHW-balance (WB_noZEB-b).

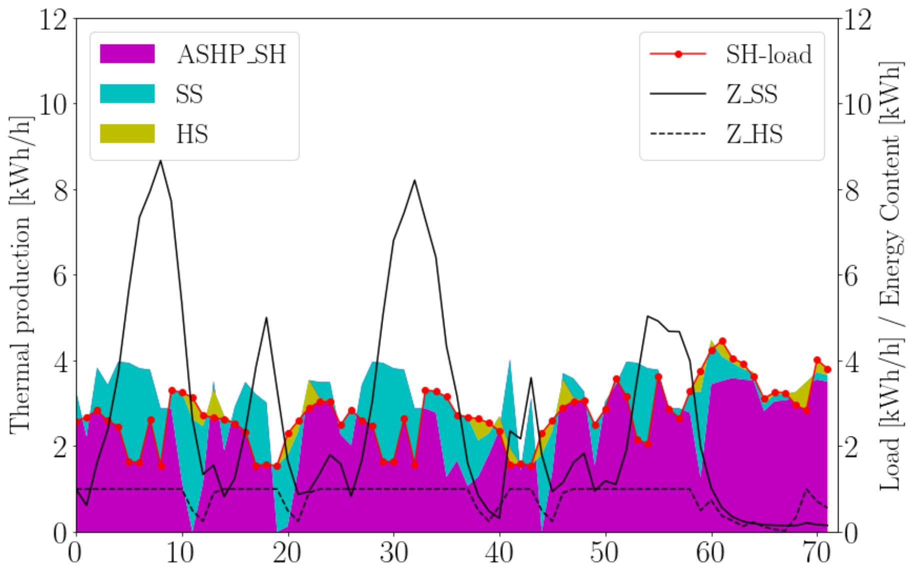

Figure 5.3.17: SH-balance (WB_noZEB-b).

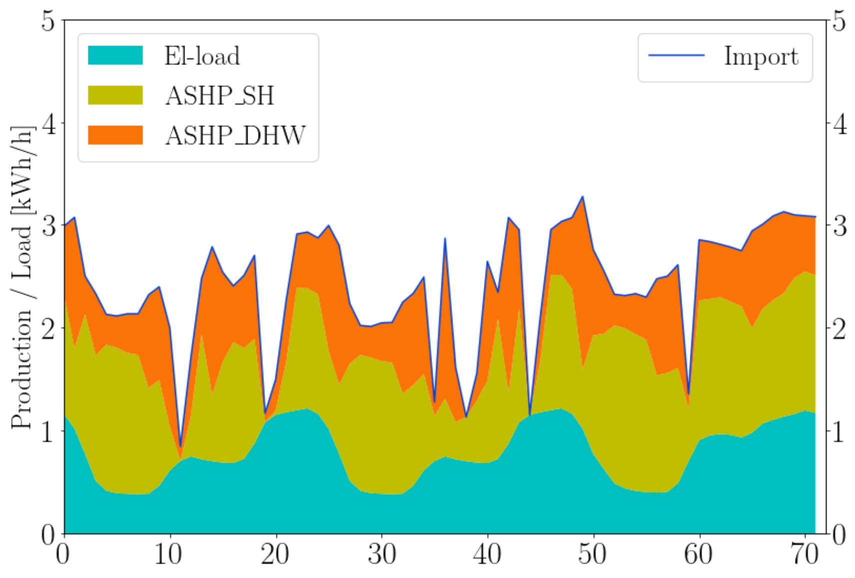

Figure 5.3.18: Total electric balance (WB_noZEB-b).

Figure 5.3.7 shows the total electric load. It differs noticeably from the WB_noZEB-case, plotted in 5.3.7, with the peaks being significantly reduced, since the EB is not present. In addition, the total load drops down to a lower level than in 5.3.7. For instance, it is as low as 1 kWh/h at hour 11, which coincides with the discharging of the HWT and SS. In general, the curve is much "sharper"; a result of the charging and discharging of these storages. Since there are no PV-panels, all of the electricity has to be imported.

with ZEB-constraint (WB_ZEB-b)

Now, the ZEB-restriction is added again. Figure 5.3.19 shows the DHW-load for this case. Just like in the case of WB_ZEB, the base load technology is switched from ASHP to GSHP. In addition, the HWT is bigger; at 5.66 kWh, compared to 2.42 kWh for the previous case. This means that a longer charging period is needed in order to fully charge the storage, so in these intervals, less GSHP capacity is available for charging the SS. Wee see this from hour 0 to 10 in figure 5.3.20; where the SS in the previous case reached a state of 8 kWh, it now stalls at just above 5 kWh in this period. On the other hand, the energy content of the SS rarely goes below 2 kWh in the three days plotted, whereas this happens five times in the WB_noZEB-case (figure 5.3.17). Furthermore, we see a more incomplete discharging of the SS after hour 10, with the storage actually reaching a content of above 10 kWh on one occasion, at approximately hour 45. In the plots shown, it appears that a new pattern emerges, where the SS has abrupt and incomplete charging cycles, while the full charging and discharging of the HWT takes precedence.

By considering figure 5.3.21, where the HWT- and BITES-utilization is plotted for the winter and spring weeks, we see that this holds on a bigger scale as well. In the winter days, the charging of the HWT is shifted to the left somewhat, since the SH-load is higher; thus requiring more free GSHP capacity for space heating. During the spring week, or more generally periods with a lower SH-load, the charging of the storages can take place simultaneously, as can be seen from the clear patterns from ca. hour 200 to 372. Here, we also see the longer latency period for HWT in the noZEB-case. In addition, in the tendency of the WB_noZEB-B case is two HWT cycles for every SS cycle, while the HWT follows the SS closely in WB_ZEB-B. The BITES-utilization seems to be somewhat higher for the ZEB-case, in part due to the extended usage in the first 72 hours.

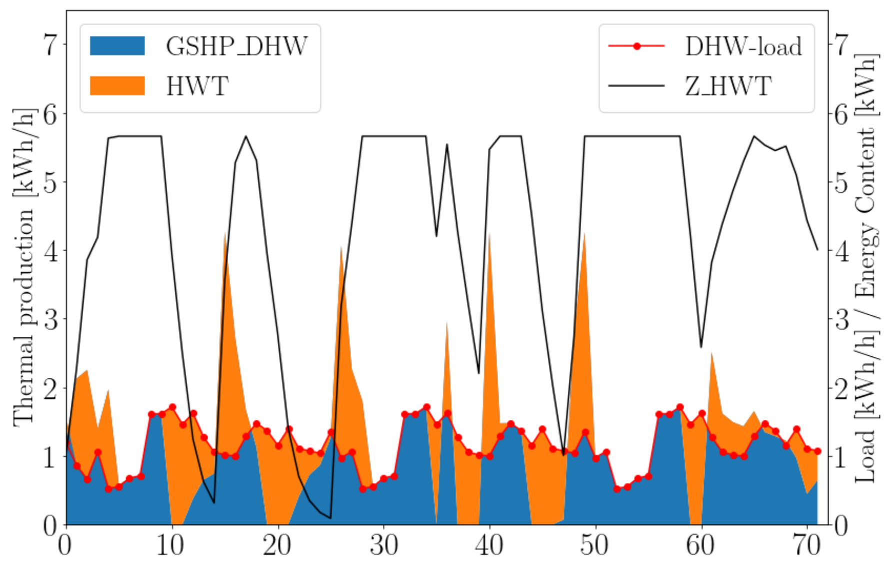

Figure 5.3.19: DHW-balance (WB_ZEB-b).

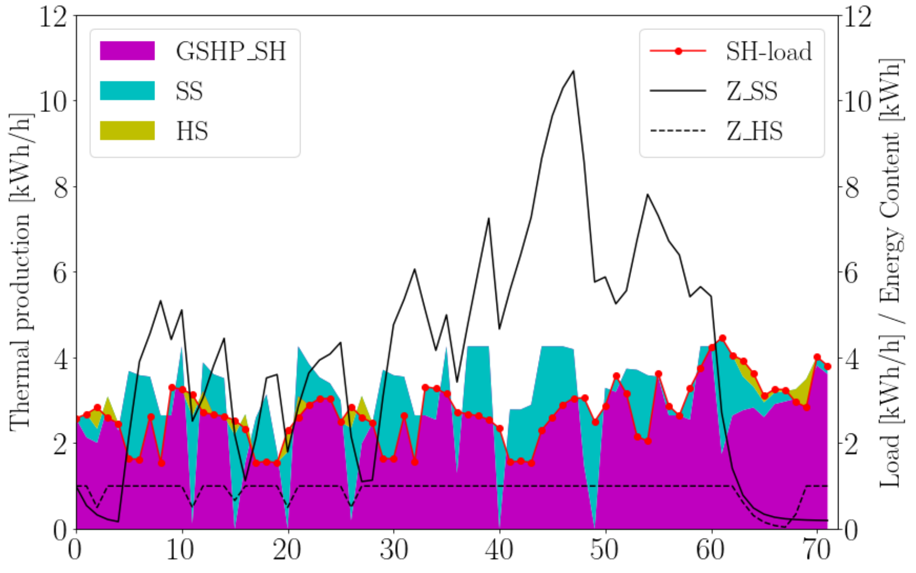

Figure 5.3.20: SH-balance (WB_ZEB-b).

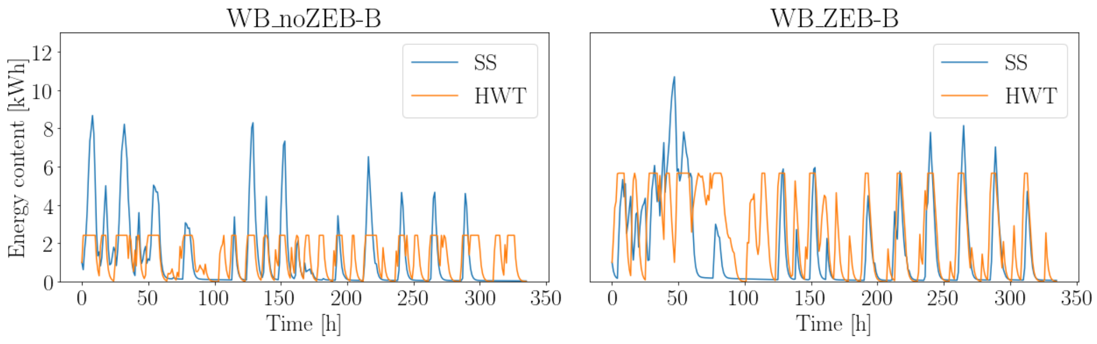

Figure 5.3.21: BITES- and HWT-utilization for winter and spring weeks.

In figure 5.3.22, the total electric load is plotted. In terms of grid impact, this case has the most optimal electric load profile. This is due to both the switch of base load technology from ASHP to GSHP and the addition of BITES. For the three days shown, the import y_{imp} hovers around 2 kWh/h most of the time, going above 2 kWh/h on two occasions. The peak import is is 2.73 kW, a reduction of 0.55 kW from WB_noZEB-B, or 16.8 %. Compared with the WB_ZEB-case, the reduction is 0.06 kW. This reduction is marginal. The total import, however, goes down from 630.06 kWh (WB_ZEB) to 576.45 kWh (WB_ZEB-B) when adding BITES, which is a more significant reduction. A possible explanation for the total import decreasing significantly more than the peak import could be that the EB use in the WB_ZEB-case usually coincides with the time when the PV-producing, thus enabling the EB to run on self-generated electricity, effectively keeping the import low.

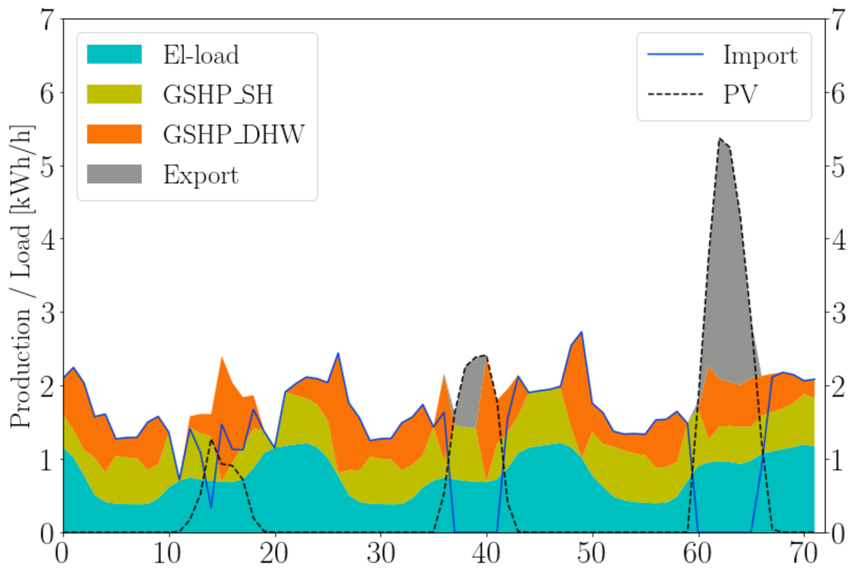

Figure 5.3.22: Total electric balance (WB_ZEB-b).

Discussion

In conclusion, it is apparent that the addition of BITES has a measurable positive effect on the waterborne system as well. In both the noZEB- and ZEB-cases, the total discounted system cost is reduced. For the noZEB-case (WB_noZEB to WB_noZEB-B), the reduction is from 39,666€ to 38,559€, or 1107 € (2.8 %). For the ZEB-case (WB_ZEB to WB_ZEB-B), it is 75,808€ to 74,080€, or 1728 € (2.28 %). In both cases, just like in the point-source cases, the reduction is brought about by a reduction in the operational costs. This reduction more than compensates for the increase in investment costs, which comes as a result of the investment in the storage technologies.

Another positive aspect of BITES utilization mentioned earlier that should be stressed again is the impact on the grid. By eliminating the need for a peak load production technology in the form of EB, the electricity is used more efficiently, since all of the heat is generated by the heat pumps. As already mentioned, this causes the peak and total import to decrease in both the WB_noZEB-B- and WB_ZEB-B-cases. When applied to a larger scale, the utilization of BITES could have a measurable impact on the grid burden of renewables, with the electricity produced from these technologies being used to operate heat pumps that pre-heat buildings in accordance with control strategies based on the principles mentioned in the previous section. This would lead to a trimming of the duration curves for imported and exported electricity, an advantageous situation for the grid operators, as the use of the grid would be more evenly distributed throughout the day.

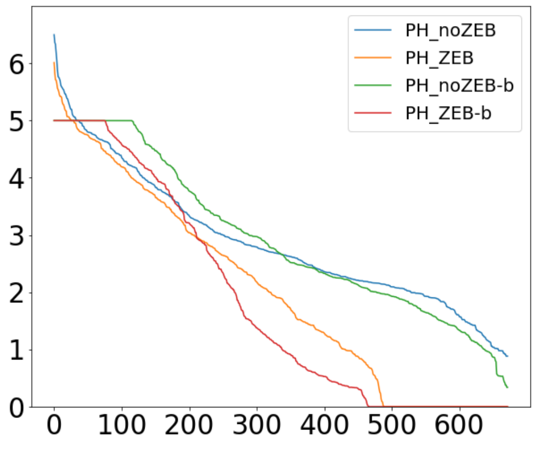

Figure 5.3.23: Duration curves PH-cases.

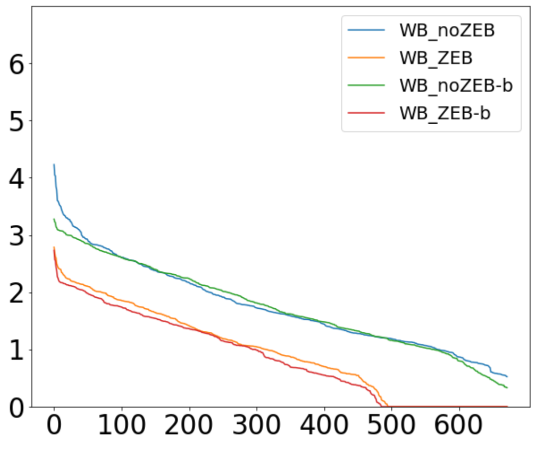

Figure 5.3.24: Duration curves WB-cases.

Figures 5.3.23 and 5.3.24 show the duration curves for electricity for the point-source cases and the waterborne cases, respectively. As for reducing the grid impact for ZEB-operation, it seems that adding BITES has a larger effect on the point-source cases, although a visible reduction can be seen as BITES is added for both WB_noZEB and WB_ZEB. When it comes to the flattening of the duration curves, it appears that the most important step towards this end is to select the waterborne heating system, seeing as how their duration curves are much less steep than the curves for the point-source cases. The most ideal case in terms of grid impact is WB_ZEB-b, with a peak total electric load of 2.73 kW, and almost 200 hours of self-sufficiency (no electricity import) through the reduced scenario.

Table 5.3.2: Summary of BITES-value. All costs in euro [€].

	0b		b		% decrease
PH_noZEB	tot.	56,386	tot.	54,651	3.08 %
	op.	52,590	op.	50,728	3.54 %
PH_ZEB	tot.	119,937	tot.	117,201	2.28 %
	op.	41,399	op.	37,851	8.6 %
WB_noZEB	tot.	39,666	tot.	38,559	2.79 %
	op.	27,111	op.	25,952	4.28 %
WB_ZEB	tot.	75,808	tot.	74,080	2.28 %
	op.	17,349	op.	15,997	7.79 %

In table 5.3.2, a summary of the cost reduction when adding BITES is presented. As was suspected in the start of this section, the total cost reduction when BITES is added is higher for the point-source system across the board. This makes sense, since the heating technologies for this system are more sensitive to spot prices, due to their lower efficiences. The total cost reduction in percent, however, is in the same range for all cases when adding BITES, 2.3-3.1 %. The largest percentage-wise decreases can be found in the operational costs for the ZEB-cases. For PH_ZEB-B and WB_ZEB-B, this decrease is at 8.6 % and 7.79 %, respectively. This could point to the combination of self-generation from the PV-panels being a particularly favorable combination. Consider the following: the PV-panels are producing for some 4-5 hours in the middle of the day. The production, even in winter, is significantly higher than the total electric load. Either all of this surplus electricity can be exported at the current spot price (generally lower in the middle of the day, when the PVs are producing), or some of it can be used to power the heating technologies, pre-heating the building thermal mass (and the HWT, as has been shown), curbing some of the import y_{imp} needed for heating purposes later in the evening, when the spot price is higher. Clearly, in such an idealized scenario, the latter is optimal, but a clear tendency towards this operational strategy can be seen in the plots in this section. The SS is generally charged in the middle of the day or at night. Naturally, there is no PV-production at night, but the spot price is generally lower at this time of the day as well, since the demand is low.

5.4 Sensitivity Analysis of two-node BITES model

In this section, a sensitivity analysis of the parameters for the two-node BITES-model will be undertaken. Since there is some uncertainty with regard to choice of parameters, it is advantageous to investigate how the size of these parameters influence the total cost of the system, in addition to the utilization of the storage. Below is a table showing the list of the relevant parameters and a brief description of each. To simplify the procedure, only one case (WB_ZEB-b) is selected for this sensitivity analysis.

Table 5.4.1: Key BITES-parameters for sensitivity analysis.

Parameter	Comment	Value in main cases
x_{ss}	"Installed" shallow storage size	12.5 kWh
x_{ds}	"Installed" deep storage size	90 kWh
τ_{ss}	Time constant for SS	115 h
τ_{ds}	Time constant for DS	267 h
K_{flow}	Flow factor (from SS to DS)	1.0 kWh/h

A physical interpretation is provided in [31] and 2.5.2. A natural starting point in the sensitivity analysis is to look at the cost reduction when the size of x_{ss} is increased, that is, the degree to which the house is allowed to overheat. Since it has already been shown that it is possible to reduce the total system cost by allowing the house to overheat, it would be reasonable to expect that raising this limit even further allows for greater reductions. However, it is to be expected that this "curbing" effect will drop off at some point. Besides, the question of comfort for the residents arises in such a situation. When it comes to the size of x_{ds}, it will be investigated whether it has anything more than an indirect effect on the utilization and system cost. In [54], which admittedly has a different scope than this work, it is suggested that the thermal mass of the building has a secondary influence on the flexibility potential of low-energy buildings (which ultimately is the topic in this work), with the heat losses being the primary driver for flexibility. In the two-node model used here, the heat losses are defined through the time constants, which yield the loss factors through equation 3.4.25. A possible deficiency of the model that arises out of defining the losses in this way is that the storage losses are independent of the ambient temperature.

The flow factor is the parameter that has the least intuitive interpretation of those in question. To put it simply, one could say that it is the tendency of the thermal energy to flow from the shallow storage to the deep storage. Using [31] and doing a simple area-adjusted calculation, we arrive at a preliminary value of 7.88 kWh/h. However, this value leads to a sparse utilization of the SS, as the majority of the thermal energy will flow to the DS before it can used for load covering. Setting this value to be too small, on the other hand, leads to the an almost constant DS, so in that case, one might just as well have used a simpler (one-node) model. In the main cases, a value of 1 kWh/h has been used. In the sensitivity analysis, the effect of K_{flow} on the cost reduction and BITES-utilization will be examined.

5.4.1 Effect of x_{ss} and x_{ds} on cost function

First, the effect of the sizes of DS and SS on the total objective cost is investigated. The purpose is to see to what degree overheating the house reduces the cost (for different sizes of DS, the "heaviness" of the building structure). The rest of the model parameters are set to their standard value.

The cost reduction for different sizes of SS and DS is plotted in figure 5.4.1. It is important to make a distinction between the nature of the variation in these parameters. For the SS, the extensive heat capacity is assumed to be fixed at

6.25 kWh/K, so the increase in x_{ss} is simply due to a larger degree of overheating allowed. For the DS, we assume the ”size” of the building construction to the same, but the extensive heat capacity to fluctuate. Hence, x_{ds} = 45 kWh implies an extensive heat capacity of 22.5 kWh/h, and 90 kWh twice that and so forth. Thus, keeping the time constant t_{ds} at the same value makes more sense. The difference in cost reduction is largest at x_{ss} = 3.125, at approximately 200 €, with the configuration having the smallest DS showing the highest cost reduction. An explanation for this could be that a larger DS absorbs more of the heat added the SS. Taking equation 2.5.10 into consideration, this seems likely, since a lower x_{ds} will decrease the flow from SS to DS, which enables the SS to retain more of the heat added to it. The cost reduction increases monotonically when x_{ss} is increased. For x_{ss} = 31.25, equivalent to a temperature increase of 5 K, the difference is at about 150 €. This shows that as the allowed temperature deviation is increased, the importance of the ”heaviness” of the building structure decreases.

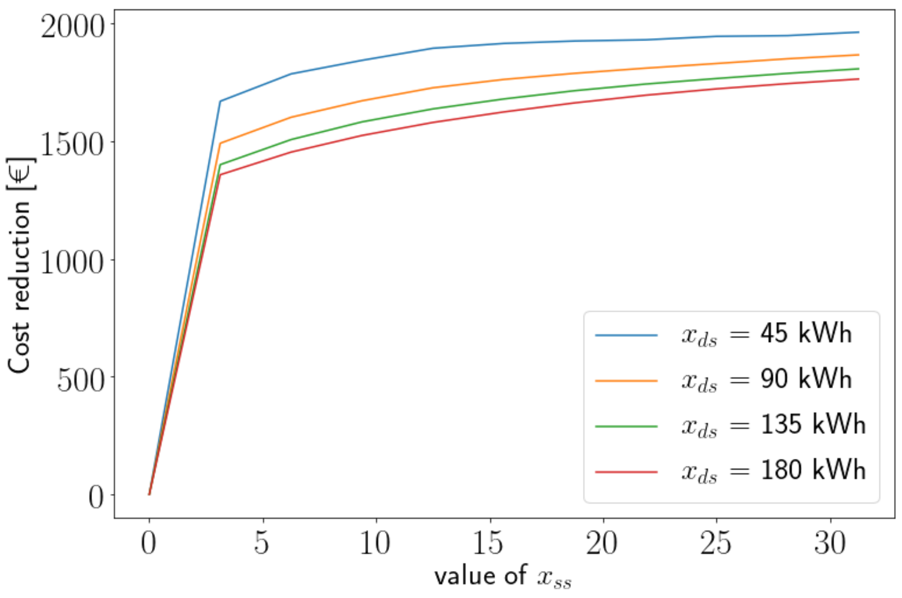

Figure 5.4.1: Effect of x_{ss} x_{ds} on total system cost.

When taking the magnitude of the changes made to the SS and DS into account, the model seems surprisingly insensitive to the parameters. In fact, it seems that the size of the DS has a greater effect on the cost reduction than the size of SS, since the difference between the DS at 45 kWh and 180 kWh is at least as big as the difference between any of the end points for a given size of DS, even though the change in magnitude for SS is greater (31.25 vs 3.125, 1000 %, 180 vs. 45, 400 %). According to the model, increasing the temperature/overheating the house by more than 0.5 °C (SS of 3.125 kWh) does not yield any significant reductions in total system cost; at most, for x_{ss} = 31.25, just below 400 € compared to x_{ss} = 3.125. This is an advantageous for the comfort-level of the inhabitants of the building, but it would be illuminating to venture an explanation for why this is the case. If there are significant differences in the utilization, we can conclude that there are a few key events (with high spot prices) that enable most of the cost reduction.

As can be seen from figure 5.4.2, BITES is used primarily in three time intervals through the reduced scenario: from hour 0 to hour 50, ca. hour 200 to 300, and from 450 to 500. In all of these intervals, it reaches the maximum overheating (3.125 kWh / 0.5 °C) at least once. The energy content in DS increases more when the peaks in SS last longer (functioning as an integrator of the heat added to SS), and decreases gradually in accordance with the time constant τ_{ds} = 267 h. It never gets close to its max capacity of 90 kWh. This case has a total cost of 74,317 €, and the GHSP is dimensioned at 4.36 kW.

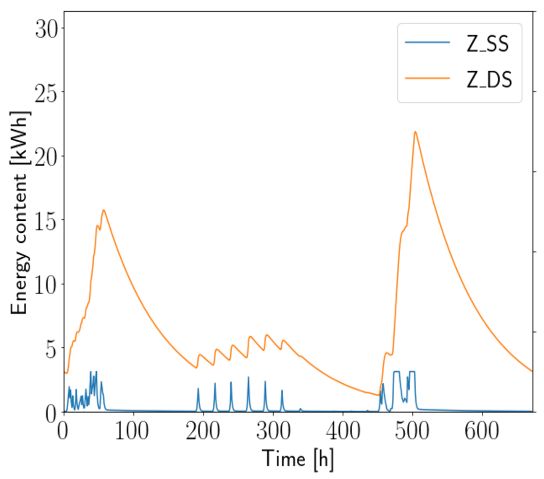

Figure 5.4.2: BITES-utilization, x_{ss} = 3.125.

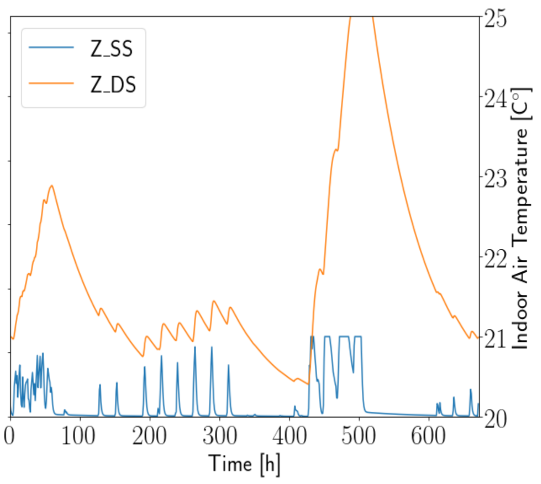

Figure 5.4.3: BITES-utilization, x_{ss} = 6.25.

Looking at figure 5.4.3, we see that the BITES-utilization increases when the building is allowed to overheat by 1 °C. Between hour 100 and 200, and towards to end of the scenario, a couple of additional incomplete charging cycles emerge, where the SS is charged to approximately 3 kWh. In addition, the charging cycles already present in the previous case increase in magnitude, with the SS reaching a state of at least 5 kWh in most of these. This case has a total cost of 74,207 €, and the GSHP is dimensioned at 4.32 kW. Figure 5.4.4 shows the BITES-utilization for the case of SS=12.5 and DS=90, the values used for the optimization in the main cases. As we can see, the utilization increases yet again, with the spikes becoming higher and more frequent. However, no new charging intervals appear, rather, the intervals already present are somewhat extended. With the exception of hour 50, and the interval between 420 and 500, the SS energy content does not exceed 6.25 kWh, which indicates that the economical value of BITES has reached a sort of saturation point at this stage. The total cost for this case is 74,081 €, and the GSHP is at 4.26 kW.

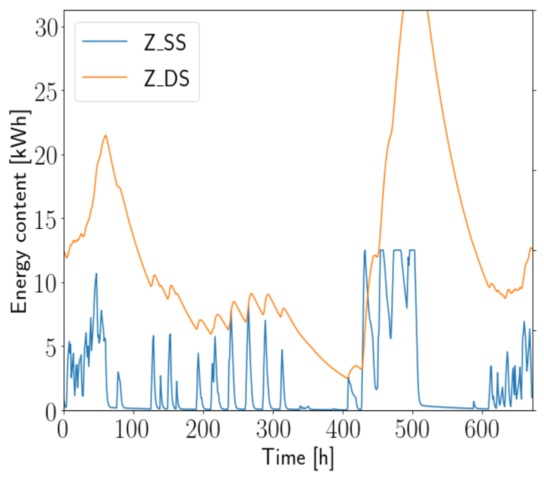

Figure 5.4.4: BITES-utilization, x_{ss} = 12.5.

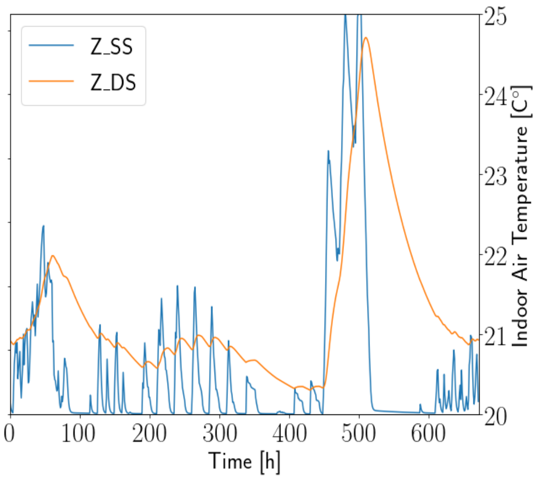

Figure 5.4.5: BITES-utilization, x_{ss} = 31.25.

The case with SS=31.25 is plotted in figure 5.4.5. This is the most extreme case, as SS at 31.25 kWh corresponds to overheating by 5 °C. The SS reaches this state on two occasions, around hour 500. Otherwise, the energy content stays below 20 kWh (ca. 3 °C). (The total cost of this case is 73,942 €, and the GSHP is at 4.24 kW.)

Table 5.4.2: Key values for sensitivity analysis of parameter x_{ss}.

x_{ss} [kWh]	GSHP [kW]	Avg. temp [°C]	Total cost [€]
0	4.71	20.00	75,808
3.125	4.36	20.05	74,317
6.25	4.32	20.16	74,207
12.5	4.26	20.36	74,081
31.25	4.24	20.59	73,942

Table 5.4.2 shows the key values for the scenarios plotted above. The average temperature is calculated by defining the set-point temperature to be 20 °C, and integrating the curves for z_{ss}, with the assumption stating that 6.25 kWh is equal to 1 °C being essential. As can be seen from the table, the ability to underdimension the GSHP compared with the 0B-case does not increase significantly beyond x_{ss}, as the difference is only 0.12 kW between x_{ss} = 3.125 and x_{ss} = 31.25. In addition, most of the cost decrease comes when increasing x_{ss} from 0 to 3.125, with difference in total cost between x_{ss} = 3.125 and x_{ss} = 31.25 only being 375 €. Thus, we can conclude that the main driver for the cost reduction the ability of the model to overheat the building *slightly* (i.e. 0.5-2 °C) with heat generated by a heat pump (in contrast to the 0b-case, where the EB is used) in peak load situations. The savings resulting from a reduction in the GSHP capacity are offset by investment in hot water tanks (see table 5.3.1). Another reason to limit the overheating is the average temperature, as it sees a significant increase as x_{ss} is increased. From the set point temperature of 20 °C in the 0B-case, it rises to 20.59 °C for x_{ss}. This in and of itself may not seem that deterrent to the comfort-level of the inhabitants, but if one considers that there are sizable intervals in which BITES is not utilized, it is far from optimal. For instance, between hour 450 and 520, i.e. close to three days, the average temperature is above 23 °C. Further research must done in order to determine whether this is acceptable for the building residents.

5.4.2 Effect of K_{flow}

In this section, the sensitivity of the results with respect to K_{flow} will be examined. First, the impact of this parameter on the total objective cost will be considered.

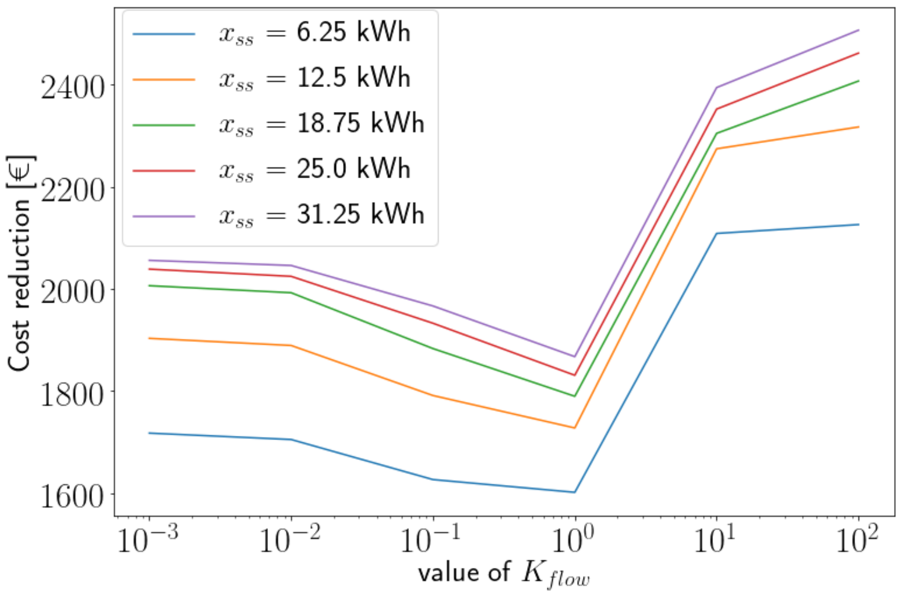

Figure 5.4.6: Effect of K_{flow} on cost reduction for different sizes of x_{ss}.

In figure 5.4.6, the cost reduction is plotted with increasing values of K_{flow}, for different values of x_{ss}. It can be seen that for each 1 °C of overheating (6.25 kWh) added to x_{ss}, the cost decrease drops in value. This is in line with the findings from 5.4.1. Furthermore, the cost decrease is at its highest value for K_{flow} = 0.001, and decreases monotonically as K_{flow} approaches 1. This makes sense, since a low K_{flow}-value implies that most of the thermal energy (except for the losses, given by the loss factor $K_{loss}^{shallow}$) stored in the SS will remain there (see equation 2.5.14), and can be used directly for load coverage. When K_{flow} increases from 1 to 100, however, the cost reduction increases again, and becomes larger than for $K_{flow} \leq 1$ The reason for this is presumably that since more energy enters the DS, this storage reaches a higher relative value than in for $K_{flow} \leq 1$. Since this storage has a high time constant (267 hours), some of this energy will flow back to the SS, and can then be used for load coverage.

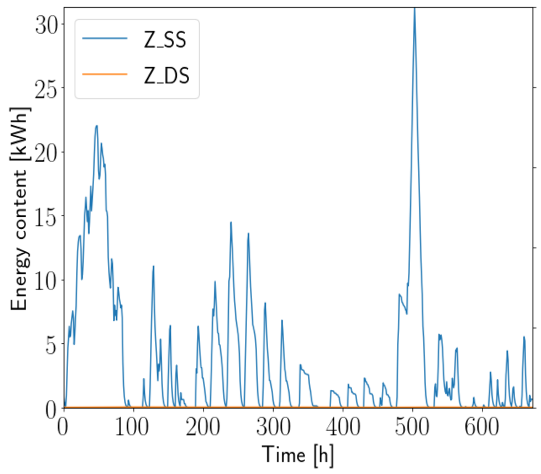

Figure 5.4.7: BITES-utilization, $K_{flow} = 0.001$.

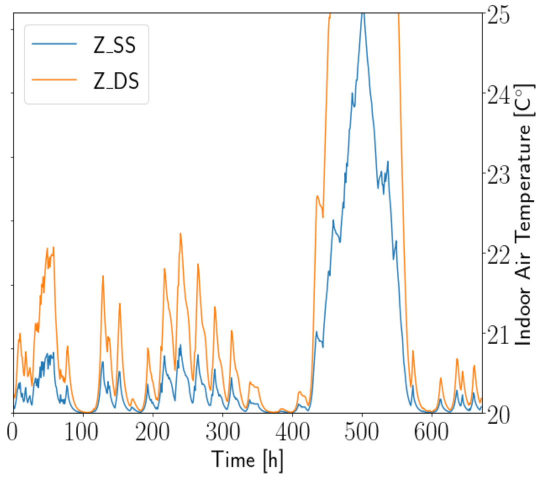

Figure 5.4.8: BITES-utilization for $K_{flow} = 100$.

Figure 5.4.7 shows the most extreme case. The building temperature, indirectly represented by x_{ss}, fluctuates to a large degree, and is at its set-point value of 20 only on a few occasions through the reduced year. Again, this might present a comfort issue for the inhabitants of the building. (In addition, the behaviour is unrealistic from a modelling point-of-view, as it takes a long time for the temperature to decrease after an initial increase). The DS is constant, so in this scenario, the two-node BITES model collapses to a one-node model (only SS). This case has a total cost of 73,752 €.

In figure 5.4.8, K_{flow} is increased to 100. We see that this has a large effect on how the BITES is utilized. Firstly, it can be seen that the energy content of the SS rarely goes above 5 kWh. As was mentioned above, this is because a large part of the energy added to the SS tends to flow to the DS, as a result of the large K_{flow}. In a sense, this means that the SS/BITES is not fully utilized (with respect to overheating), considering that the SS only goes above 5 kWh once during the whole scenario. On the other hand, we see that the curve for z_{ss} is ”flatter” than what is the case in figure 5.4.7. This implies that some of the energy in the DS flows back to the SS, since there is only two ways for this energy to flow, either to the ambient (implicit via the loss factor), or back to the SS (via flow, see equation 2.5.10). From ca. hour 420 to 580, there is a large interval (almost a week) in which the SS is charged to some degree for the duration. The average temperature in this sequence is 22.21 °C. This case has a total cost of 73,301 €, which is a 450 € decrease from the previous case.

Table 5.4.3: Key values for sensitivity analysis of parameter K_{flow}.

x_{ss} [kWh]	K_{flow}	GSHP [kW]	ASHP [kW]	Avg. temp [°C]	Cost reduction [€]
31.25	0.001	4.14	0	20.60	2056
31.25	1.0	4.24	0	20.59	1867
31.25	100	0	4.40	20.66	2507
6.25	0.001	4.37	0	20.19	1718
6.25	1.0	4.32	0	20.16	1602
6.25	100	0	4.34	20.17	2126

Some key values for the sensitivity analysis with respect to K_{flow} are tabulated in table 5.4.3. For $K_{flow} = 100$, the optimal heating technology switches from the GSHP to the ASHP. This, along to the return flow from the DS to the SS, will also contribute to the cost reduction.

5.4.3 Effect of time constants τ_{ss} and τ_{ds}

In this section, the effect of the time constants τ_{ss} and τ_{ds} on the cost reduction will be investigated. For the main results, values of 115 h and 267 h were used, respectively. These values were taken directly from [31], where an analysis of the cost reduction potential of BITES and HWTs in the Göteborg-area district heating system was undertaken. Thus, these time constants are meant to reflect the first-order response of several thousand houses operating in conjunction, and the time constants might be unsuitable for use in building-level framework.

Figure 5.4.9 shows the cost increase as the size of x_{ss} is increased, for six different values of τ_{ss}. The step size for x_{ss} is 6.25 kWh. Again, a clear trend can be seen, where the most significant change in cost reduction comes when x_{ss} goes from zero to the second smallest size, in this case 6.25 kWh. For x_{ss} = 6.25 kWh, the difference in cost reduction between the smallest time constant, 20 h, and the biggest time constant, 120 h, is about 250 €. This is a natural consequence of the way the losses are defined in the two-node BITES-model; since the time constant t_{ss} is smaller, more heat is lost relative to the current state (i.e. lower loss factor). For τ_{ss} = 120 h, 5 h higher than in the standard case, the cost reduction shows the same tendency as we have seen in the previous analyses, with a slight increase as x_{ss} is increased. As we downwards, however, we see that this slight increase flattens out, until it almost appears to vanish completely. For τ_{ss} = 20, it seems that the cost reduction is completely independent of the degree of overheating allowed. This warrants further investigation.

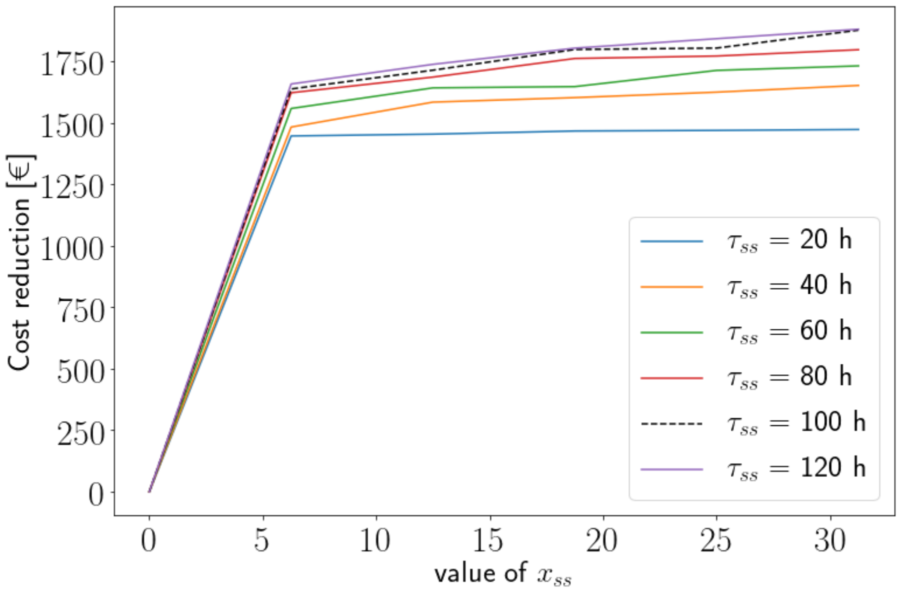

Figure 5.4.9: Effect of τ_{ss} on cost reduction.

Figure 5.4.10 shows the BITES-utilization for τ_{ss} = 20 h and x_{ss} = 6.25 kWh. We see that the utilization is sparse, with the SS reaching 6.25 kWh only twice during the reduced scenario, around hour 500. Otherwise, it barely gets up to 3 kWh. For x_{ss} = 31.25 kWh, a similar pattern emerges, with the two peaks of 6.25 kWh are replaced by one of approximately 16 kWh, and the energy content of the SS otherwise rarely exceeding 5 kWh. In other words, with a time constant τ_{ss} of 20 h, the heat escapes the SS so quickly that there are few situations in which it is optimal to charge it fully. Since the cost reduction is almost exactly the same for x_{ss} = 6.25 and x_{ss} = 31.25, we can say that the intervals in which BITES is utilized in the former case constitute the key events mentioned earlier, which contribute the most towards the cost reduction.

Another conspicuous feature can be noticed in figure 5.4.11, namely that the energy content of the DS significantly lower than in all previous cases. This is a result of the small time constant τ_{ss} and the way the cross-node flow is defined. Since it is expressed in relative terms, and the energy content of SS rarely exceeds 5 kWh, the flow from SS to DS will be quite small during the whole scenario. This is an obvious error in the model, arising from the non-physical nature of the parameter K_{flow}. It should be clear that the energy content in this case should be at least as high as the case with τ_{ss} = 20 h and x_{ss} = 6.25 kWh.

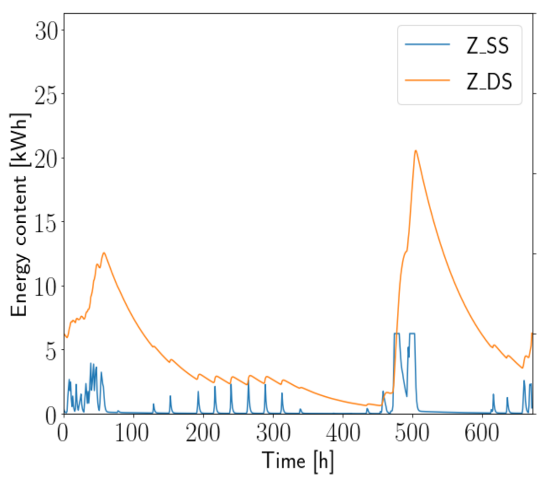

Figure 5.4.10: BITES-util, τ_{ss} = 20 h, x_{ss} = 6.25 kWh.

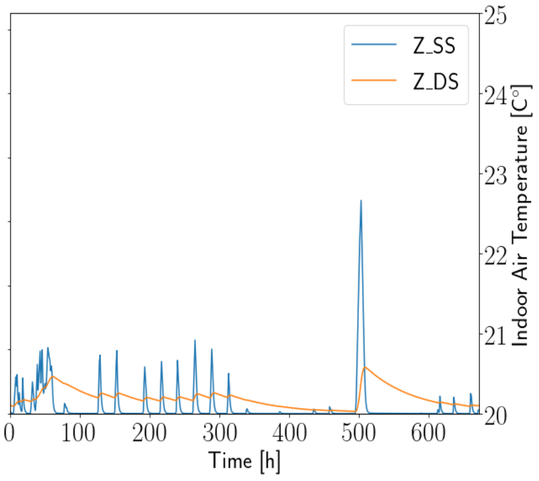

Figure 5.4.11: BITES-util, τ_{ss} = 20 h, x_{ss} = 31.25 kWh.

Figure 5.4.12 shows the effect of the deep storage time constant, τ_{ds}, on the cost reduction. The value of this parameter in the main section is 267 hours. Here, values from 50 h to 300 h are considered, with a step size of 50 h. The same trend as for τ_{ss} can be observed, with the cost reduction increasing as the time constant τ_{ds} is increased. However, the increase in cost reduction as x_{ss} is increased persists for all time constants considered, in contrast to what was the case for τ_{ss}. This can be seen by noting the difference in cost reduction for x_{ss} = 6.25 kWh and x_{ss} = 31.25 kWh. However, the difference in cost reduction when x_{ss} is 6.25 kWh for the smallest and biggest time constant is similar, around 250 €.

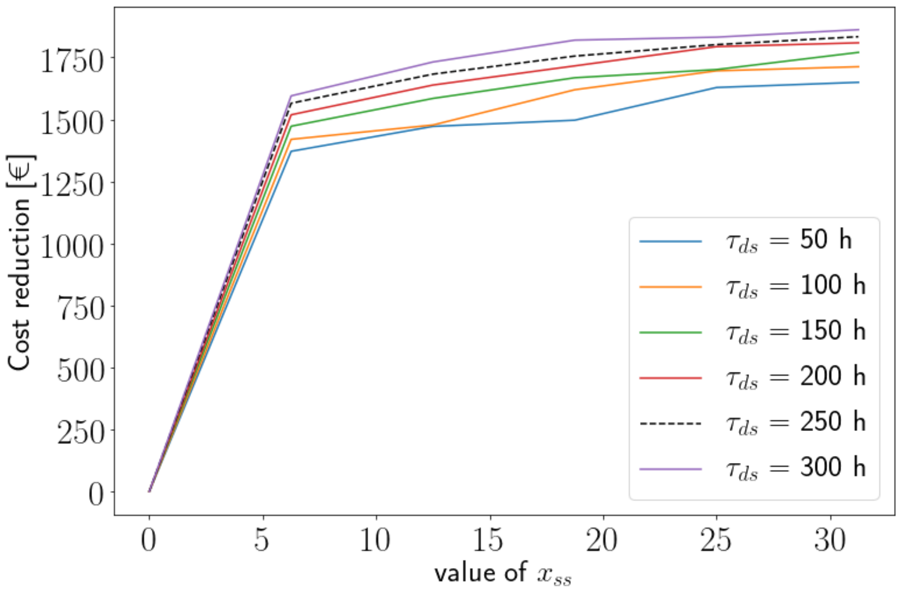

Figure 5.4.12: Effect of τ_{ds} on cost reduction.

5.4.4 Discussion

The sensitivity analysis shows that the implemented two-node BITES model exhibits some sensitivity to changes in its parameters. Increasing the degree to which the building is allowed to overheat (x_{ss}) from 0.5 °C to 5 °C gives an increase in cost reduction of 500€ at the most (x_{ds} = 90). This relatively small decrease seems to stem from a combination of removing the EB, the selected peak load heating technology for most of the cases, and reduced operational costs, i.e. overheating when the spot price is low.

For the parameter K_{flow}, the tendency of the thermal energy added to SS to flow to DS, another picture emerges. For $K_{flow} < 1$, the difference in cost reduction is at most ca. 400 €, but as can be seen in figure 5.4.7, this represents a scenario in which the energy content in DS is almost constant, which means that practically no energy flows from the SS to the DS (one-node case). For $K_{flow} = 1$, the value used in the main cases, the difference in cost reduction is at minimum; ca. 300 €. For $K_{flow} \geq 1$, the difference in cost reduction rises again, until it reaches 400 € again for $K_{flow} = 100$. For this value of K_{flow}, the cost reduction is at a maximum for all sizes of x_{ss}. This suggests that poor insulation between the SS and DS gives the highest value for BITES. However, due to the abstracted nature of the two-node model, it is not strictly correct to interpret K_{flow} as a physical parameter (unit is kWh/h).

The question then becomes: does the implemented two-node BITES model capture the essential elements of the building's thermal mass? For instance, it seems unlikely that the value of BITES as a thermal energy storage is large independent of the thermal properties of the building 2.5.4. Consider the values for K_{flow}. This parameter, although it has the unit kWh/h, can in a sense be interpreted as the inverse of the thermal resistance. According to [36], the difference between a thermal insulator and a thermal conductor is about three orders of magnitude. The values considered for K_{flow} in the previous section is five orders of magnitude. Hence, according to the model, the choice of material for thermal insulation has little effect on the usefulness of BITES, which clearly is an unrealistic situation. In addition, situations emerge where heat flows from the deep storage to the shallow storage, despite the ambient temperature being lower than the inside air temperature. On the other hand, we might consider all of the material data to be reflected in the time constants, which yield the losses to the ambient. In that case, the flaw of the model is more in its lack of an intuitive interpretation, rather than a strict modelling error. This may be explained by its original use, techno-economical analysis of a larger DH network.

The sensitivity analyses with respect to the time constants τ_{ss} and τ_{ds} show that the value of BITES increases with higher time constants. If the time constant for SS, τ_{ss}, is decreased enough, increasing the size of x_{ss} beyond 6.25 kWh does not yield any further cost reduction. For lower values of τ_{ds}, the cost reduction increases at about the same rate as for higher values when x_{ss} is increased. The time constants used in the main results may be too high for a single building, which leads to an overestimation of the economical value of BITES. With this in mind, some remarks about the time constants can be made. As mentioned above, the time constants can be considered to reflect the material properties of the building. This is evident when one considers equation 2.5.24, where the conductance is divided into two elements, one for conduction (materials) and one for convection (ventilation). These elements correspond almost perfectly to the deep/shallow-separation in the two-node model. According to this equation, it is possible to increase the time constant not only by increasing the building thermal mass, but also by decreasing the thermal losses (that is, increasing the thermal resistance). Thus, the shallow storage time constant τ_{ss} can be increased simply by decreasing the ventilation rate, although this may have detrimental effects in other aspects. The deep storage time constant τ_{ds}, on the other hand, is fixed by the construction of the building. With reference to 2.5.4, we can say that materials with high thermal resistivity and high heat capacity are favorable for BITES use.

To find out whether the lack of sensitivity in the two-node BITES model resides in the two-node model itself, or if it is due to the lack of technical detail in the optimization framework, the 3R2C model from 2.5.3 can be implemented. Then, a similar sensitivity analysis can be performed, to see if the conclusions from the initial analysis hold. This is left for further work. However, as long as the hourly time step is kept, it is questionable whether any thermal mass representation can improve upon the accuracy of the two-node BITES model.

Chapter 6

Conclusion and Further Work

6.1 Conclusion

A deterministic MILP optimization model, formulated in Python using the modelling extension library Pyomo, has been studied. The model is used to find optimal operation strategies and technology investments for a passive house building, based on a wide array of input data, in order to reduce the operational net emissions of the building during its lifetime, in accordance with the ZEB-constraint. Using the work in [2] as a starting point, a new framework was developed, with the Space Heating and Domestic Hot Water Loads separated. Furthermore, two versions of the model were synthesized, one with a point-source heating system, and another with waterborne heating.

Several cases were investigated, with and without the ZEB-constraint. This was done for both the point-source and waterborne models. With the given input data, it was shown that the waterborne heating system was the most optimal for a low-energy house, as it had a lower total cost both with and without the ZEB-constraint. Additionally, owing to the to higher efficiences (COPs) of the technologies available for the waterborne system, the grid impact/burden was also more favorable for this system, since less import is required in order to supply to necessary heat, which in turn leads a reduced need to export electricity as the ZEB-restriction is added.

To investigate the flexibility potential of low-energy buildings, the two-node representation of the building internal thermal energy storage (BITES) developed in [31] was implemented in both versions of the model. The addition of BITES as an available storage technology had a favorable impact on both systems. It was found that the BITES could replace the parts of the heating systems that were intended for peak load use, essentially functioning as a short-term buffer for the space heating load. For both system configurations, this was done in combination with significant investments in Hot Water Tanks (HWT) (above the peak DHW-load), which served as a buffer for the domestic hot water (DHW) load. The most promising BITES-cases, compared to their noBITES counterparts, were PH_ZEB-b and WB_ZEB-b; showing decreases in the operational costs of 8.60 % and 7.79 %, respectively.

Finally, a sensitivity analysis of the two-node BITES model was undertaken, in order to find a range of values for the cost reduction when BITES is added, in addition to the suitability of this thermal mass representation in a MILP optimization framework with an hourly time step. It was found that the economical value of BITES was at least as dependant on the "heaviness" of the building structure as the degree to which the building was allowed to overheat. The effect of the time constants on the solution was also examined, which suggested that larger time constants yield higher cost reductions. Furthermore, the suitability in using the non-physical parameter K_{flow} on a building-level was called into question, with an alternative model found in literature, based on a circuit analogy, suggested as an alternative representation of the building thermal mass.

6.2 Further Work

Extension of scope to neighborhood

In this work, the topic is the optimization of an energy system for a single building of passive house standard, with a focus on load flexibility by using the building thermal mass as a short-term energy storage. Using a more object-oriented approach in the programming of the model, the scope of the model can be extended to e.g. neighborhood level. Then, several buildings operating in conjunction can be studied. This has already been done in [55], for the ZEN-pilot project in Evenstad, Norway. A point of interest in this context is to see the effect of BITES on a larger scale. For instance, will the load-shifting effect of BITES persist when the scope is increased, or will the peak loads become too high when the buildings charge the thermal mass and hot water tanks at the same time?

Alternative representation of building thermal mass

As was shown by the sensitivity analysis, there are some problems with the two-node representation of the building thermal mass used. For instance, there is the non-physical nature of the parameter K_{flow}. In addition, the losses are defined through the time constants, which makes them independent of the outside temperature. A way to remedy this is to implement a thermal network model based on a circuit analogy. Then, the losses will depend on the outside temperature, and the cross-flow resistance/conductance will be defined in terms of K/W. The problem now becomes to find reasonable values for these parameters. A first approximation could be to use the time constants and the assumed heat capacities for the SS and DS from [31] to calculate the thermal resistances, using basic circuit theory. This will yield a certain range of values that can be considered. It is important to keep in mind that although the units of the thermal resistance suggest that the parameters have physical meaning, they must be considered to be equivalent or aggregated parameters. That is, they do not represent a certain wall or part of the roof, but rather a mathematical abstraction of the building as a thermal energy storage.

Control

In order to actualize the savings potential of BITES, clever control is required. In the optimization model used in this work, perfect foresight is assumed with respect to loads, weather data and spot prices. In a real-world application, this lack of determinism could be alleviated by using time-series prediction techniques and Model Predictive Control [56]. Using weather forecasts and historical data as input to the prediction (spot price and heat load, most importantly), these can be used to determine the optimal action in the current time step. For a more detailed model of BITES, a 3R2C model such as the one studied in [57] could be implemented. Another alternative approach is to build a model from measurement data, an approach known as inverse modelling [58]. The downside here is the access to measurement data.

Improved modelling of COP of heat pumps

In figure 4.3.5, the COPs of the different heat pumps for space heating and domestic hot water purposes is plotted. It is obvious that a COP of close to 10, seen in the summer week for the GSHP for space heating, is an unrealistic number. An effort should be made to improve the way the COP is modelled. However, some alleviating remarks can be made here. First of all, as was mentioned in the chapter on input data, the space heating load is usually very low in the intervals where the COP is this high. As a result, the amount of heat produced with this COP is actually quite low. Secondly, according to [23], a review paper on domestic heat pumps, the interval in which the polynomials used to calculate the COPs are valid is for ΔT between 20 and 60 °C. Hence, for a low ΔT, that is, a low difference between the source and sink temperatures, the polynomials are invalid, and a new method must be found to approximate the COPs.

Bibliography

[1] Intergovernmental Panel on Climate Change, *Summary for Policymakers*. Cambridge University Press, 2014, p. 1–30.

[2] I. M. Andersen, "Stochastic Optimization of Zero Emission Buildings," Master's thesis, Institute for Electrical Power Engineering, Norwegian University of Science and Technology, 2018.

[3] S. Fufa, R. Dahl Schlanbusch, I. Andresen, M. Kjendseth Wiik, and K. Sørnes, "A Norwegian ZEB Definition Guideline," January 2016.

[4] K. B. Lindberg, "Zero Energy Buildings on the Power System: A study of load profiles, flexibility and system investments," Ph.D. dissertation, Department of Electrical Power Engineering, Norwegian University of Science and Technology, 2017. [Online]. Available: https://brage.bibsys.no/xmlui/handle/11250/2450566

[5] K. B. Lindberg and G. Doorman, "Hourly load modelling of non-residential building stock," in *2013 IEEE Grenoble Conference*. IEEE, June 2013, pp. 1–6.

[6] K. B. Lindberg, G. Doorman, J. E. Chacon, and D. Fischer, "Hourly electricity load modelling of non-residential passive buildings in a nordic climate," in *2015 IEEE Eindhoven PowerTech,* 29.6 – 2.7 2015. IEEE.

[7] European Environment Agency. (2019) Global and European temperature. Last visited on 19-06-2019. [Online]. Available: https://www.eea.europa.eu/data-and-maps/indicators/global-and-european-temperature-9/assessment

[8] T. Boermans, A. Hermelink, S. Schimschar, J. Grozinger, and M. Offerman, "Principles for Nearly Zero-Energy Buildings," 2011. [Online]. Available: http://bpie.eu/wp-content/uploads/2015/10/HR_nZEB-study.pdf

[9] Zero Emission Building project. (2017) ZEB Final Report - 2009-2017. Last visited on 2019-06-20. [Online]. Available: https://www.zeb.no/index.php/en/final-report

[10] Thomas Stetz, Manoel Rekinger, Ioannis Theologitis. (2014) Transition from uni-directional to bi-directional distribution grids. Last visited on 2019-06-20. [Online]. Available: https://smartgrids.no/wp-content/uploads/sites/4/2014/11/Report-IEA-PVPS-T14-03_2014.pdf

[11] Norges vassdrags- og energidirektorat (NVE). (2018) Nasjonal varedeklarasjon 2016. Last visited on 2019-06-20. [Online]. Available: https://www.nve.no/reguleringsmyndigheten-for-energi-rme-marked-og-monopol/varedeklarasjon/nasjonal-varedeklarasjon-2016/

[12] ——. (2015) Energibruk i norge. Last visited on 2019-06-20. [Online]. Available: https://www.nve.no/energibruk-og-effektivisering/energibruk-i-norge/

[13] S. Merlet and B. Thorud. (2015) Solenergi i Norge: Status og fremtidsutsikter. Last visited on 2019-06-20. [Online]. Available: http://energiogklima.no/kommentar/solenergi-i-norge-status-og-framtidsutsikter/

[14] I. M. Andersen, "Investment opportunities for batteries in zero emission buildings," Project work fall 2017 (Unpublished), Department of Electrical Power Engineering, Norwegian University of Science and Technology.

[15] Gurobi. (2018) Mixed-Integer Programming (MIP) Basics — Gurobi. Last visited on 2018-10-30. [Online]. Available: http://www.gurobi.com/resources/getting-started/mip-basics

[16] M. Bagle, "Integration opportunities for wind turbines in Zero Emission Neighborhoods," Project work fall 2018 (Unpublished), Department of Electrical Engineering, Norwegian University of Science and Technology.

[17] M. Dhar. (2017) You'd also like. Last visited on 2019-05-24. [Online]. Available: https://www.livescience.com/41995-how-do-solar-panels-work.html

[18] samlexsolar.com. (2019) Learn about the basics of off grid solar systems — samlex solar. Last visited on 2019-05-24. [Online]. Available: http://samlexsolar.com/learning-center/solar-systems-basics.aspx

[19] Narasimhan. (2019) Solar cell - the basic unit of a solar power unit. Last visited on 26-05-2019. [Online]. Available: http://www.suncyclopedia.com/en/solar-cell-the-basic-unit-of-a-solar-power-plant/

[20] G. M. Master, *Renewable and Efficient Electric Power Systems*, 2004. [Online]. Available: http://doi.wiley.com/10.1002/0471668826

[21] T. Huld, R. Gottschalg, H. G. Beyer, and M. Topic, "Mapping the performance of pv modules, effects of module type and data averaging," *Solar Energy*, vol. 84, no. 2, pp. 324 – 338, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0038092X0900293X

[22] S. Nordahl, "Design of Roof PV Installation in Oslo," Master's thesis, Institute for Electrical Power Engineering, Norwegian University of Science and Technology, 2012.

[23] I. Staffell, D. Brett, N. Brandon, and A. Hawkes, "A review of domestic heat pumps," *Energy Environ. Sci.*, vol. 5, pp. 9291–9306, 10 2012.

[24] Y. A. Cengel and M. A. Boles, *Thermodynamics: An Engineering Approach*, 6th ed. McGraw-Hill, 2008.

[25] K. B. Lindberg, G. Doorman, D. Fischer, M. Korpås, A. Ånestad, and I. Sartori, "Methodology for optimal energy system design of zero energy buildings using mixed-integer linear programming," *Energy and Buildings*, vol. 127, pp. 194 – 205, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S037877881630408X

[26] The Green Age. (2019) Heaters - electric boilers. Last visited on 2019-06-04. [Online]. Available: https://www.thegreenage.co.uk/tech/electric-boilers/

[27] I. Dryden, "Chapter 7 - electrical heating fundamentals," in *The Efficient Use of Energy*, 2nd ed. Butterworth-Heinemann, 1982, pp. 94 – 114. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780408012508500167

[28] R. Saidur, E. Abdelaziz, A. Demirbas, M. Hossain, and S. Mekhilef, "A review on biomass as a fuel for boilers," *Renewable and Sustainable Energy Reviews*, vol. 15, no. 5, pp. 2262 – 2289, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1364032111000578

[29] M. Fossdal, E. Arnstad, K. Mathiesen, and B. Eriksen, "Fornybar energi 2007," *NVE, Enova, NFR, Innovasjon Norge*, p. 182, 2007. [Online]. Available: www.fornybar.no

[30] J. A. Duffle and W. A. Beckmann, *Solar Engineering of Thermal Processes*. John Wiley and Sons, Inc., 2013.

[31] D. Romanchenko, J. Kensby, M. Odenberger, and F. Johnsson, "Thermal energy storage in district heating: Centralised storage vs. storage in thermal inertia of buildings," *Energy Conversion and Management*, vol. 162, pp. 26 – 38, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0196890418300803

[32] J. Kensby, A. Truschel, and J.-O. Dalenback, "Potential of residential buildings as thermal energy storage in district heating systems - Results from a pilot test," *Applied Energy*, vol. 137, pp. 773 – 781, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261914007077

[33] J. Carlsson, "Marginal Price Control of Buildings Utilised as Thermal Energy Storage - Optimising the heating cost of a modelled residential building," Master's thesis, 2016.

[34] J. Hedbrant, "On the thermal inertia and time constant of single-family houses," Licentiate Thesis, Linköping University, Energy Systems, The Institute of Technology, 2001.

[35] R. Elliott. (2013) ESP - Heatsink design and transistor mounting. Last visited on 2019-06-02. [Online]. Available: http://sound.whsites.net/heatsinks.htm

[36] C. J. M. Lasance, "Ten Years of Boundary-Condition- Independent Compact Thermal Modeling of Electronic Parts: A Review," *Heat Transfer Engineering*, vol. 29, no. 2, pp. 149–168, 2008. [Online]. Available: https://doi.org/10.1080/01457630701673188

[37] Frank P., David P. DeWitt, Theodore L. Bergman, Adrienne S. Lavine, *Fundamentals of Heat and Mass Transfer*, 6th ed. John Wiley and Sons, 2007, p. 260–261.

[38] O. T. Ogunsola, L. Song, and G. Wang, "Development and validation of a time-series model for real-time thermal load estimation," *Energy and Buildings*, vol. 76, pp. 440–449, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.enbuild.2014.02.075

[39] C. Alexander and M. Sadiku, *Fundamentals of Electric Circuits*, 4th ed. McGraw Hill Higher Education, 2008.

[40] S. N. Bjarghov, "Utilizing EV Batteries as a Flexible Resource at End-user Level," Master's thesis, Institute for Electrical Power Engineering, Norwegian University of Science and Technology, 2017.

[41] J. S. Erdal, "Stochastic Optimisation of Battery System Operation Strategy under different Utility Tariff Structures," Master's thesis, Institute for Electrical Power Engineering, Norwegian University of Science and Technology, 2017.

[42] Nordpoolspot. (2019) Historical Market Data — Nord Pool. Last visited on 2019-05-30. [Online]. Available: https://www.nordpoolgroup.com/historical-market-data/

[43] Bioforsk. (2019) Landbruksmeteorologisk tjeneste. Last visited on 2019-05-30. [Online]. Available: https://lmt.nibio.no/

[44] Defa. (2019) Nobø Top panelovn - Effektiv panelovn til hytta - DEFA Hyttestyring. Last visited on 2019-06-20. [Online]. Available: https://www.defa.com/no/produkt/nobo-top-panelovn-20cm/

[45] Byggmaker. (2019) Vedovn 40 Cbs Sort Lakk - Dovre. [Online]. Available: https://www.byggmakker.no/varme-og-ventilasjon/peis-og-vedovn/vedovn-dovre-40cbs--sort-lakk/

[46] Mitsubishi Electric. (2019) Kaiteki 6300. Last visited on 2019-06-20. [Online]. Available: https://www.miba.no/butikk/aircondition123/kaiteki-6300-hvit/

[47] Byggmaker. (2019) Vedovn 40 cbs sort lakk - dovre - byggmakker aasen five. Last visited on 2019-06-18. [Online]. Available: https://www.byggmakker.no/varme-og-ventilasjon/peis-og-vedovn/vedovn-dovre-40cbs--sort-lakk/

[48] kvalitetsved.no, "Vedfakta," 2019, last visited on 2019-06-23. [Online]. Available: http://www.kvalitetsved.no/ved-fakta-riktig-pris-pa-ved

[49] I. Andresen, K. Buvik, C. Grini, K. Sjøstrand, M. Thyholt, and T. Wigenstad, "Miljøvennlig varmeforsyning til lavenergi- og passivhus," SINTEF Byggforsk, Tech. Rep., September 2010. [Online]. Available: https://www.sintefbok.no/book/download/655/vinfopubutgivelserprosjektrapportsintef_byggforsk_prosjektrapportersb_prrapp_59sb_prrapp_59pdf

[50] UK Government. (2019) Greenhouse gas reporting: conversion factors 2019. Last visited on 2019-06-07. [Online]. Available: https://www.gov.uk/government/publications/greenhouse-gas-reporting-conversion-factors-2019

[51] Akershus Enøk og Inneklima AS. (2019) Enøk i norge. Last visited on 2019-05-31. [Online]. Available: https://www.enok.no/felleskalkf6b9.html?f=sparedusj

[52] Stiebel Eltron, "Planung und Installation Warmepumpen," *Holzminden: Stiebel Eltron GmbH*, 2013.

[53] Encore Geothermal. (2013) Mitsubishi zuba cold climate air source heat pumps. Last visited on 2019-06-21. [Online]. Available: https://web.archive.org/web/20141021195035/http://encore-geothermal.ca/sustainable-solutions/air-source-heat-pumps/

[54] K. Foteinaki, R. Li, A. Heller, and C. Rode, "Heating system energy flexibility of low-energy residential buildings," *Energy and Buildings*, vol. 180, pp. 95 – 108, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378778818314555

[55] D. Pinel, M. Korpås, and K. Lindberg, "Cost Optimal Design of Zero Emission Neighborhoods' (ZENs) Energy System: Model Presentation and Case Study on Evenstad," March 2019.

[56] J. Reynolds, M. W. Ahmad, Y. Rezgui, and J.-L. Hippolyte, “Operational supply and demand optimisation of a multi-vector district energy system using artificial neural networks and a genetic algorithm,” *Applied Energy*, vol. 235, pp. 699 – 713, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261918317070

[57] M. A. Fayazbakhsh, F. Bagheri, and M. Bahrami, “A Resistance-Capacitance Model for Real-Time Calculation of Cooling Load in HVAC-R Systems,” *Journal of Thermal Science and Engineering Applications*, vol. 7, December 2015.

[58] R. Kramer, J. van Schijndel, and H. Schellen, “Simplified thermal and hygric building models: A literature review,” *Frontiers of Architectural Research*, vol. 1, no. 4, pp. 318 – 325, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2095263512000647

Appendix A

This appendix contains the code for the waterborne model. The following python script, MODEL_wb_SH.py, defines the waterborne model:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 12 20:17:29 2019

@author: marius
"""
import pyomo.environ as pyo


class ZEBModel():

    def __init__(self, M_const = 1000):
        """Create Abstract Pyomo model for ZEB
        """
        # Determenistic two-stage model
        self.abstractmodel = self.createTWOSTAGEMODEL()

        self.M_const = M_const

    def disco(self, n, r):
        '''Discounting factor'''
        return 1/((1+r)**n)

    def annui(self, n, r):
        '''Annuity factor'''
        return r/(1-(1+r)**(-n))

    def capit(self, n, r): # Capitalization factor
        '''Capitalization factor'''
        return (1-(1+r)**(-n))/r

    def cost(self, Yn, cost, l, r):
        '''For the two-stage model: calculating forced reinvestment costs'''
        Kn = pyo.floor(Yn/(l*1))
        n = Yn-l*Kn
        Tn = Yn-n
        return cost*(self.annui(l,r)*self.capit(n,r)*self.disco(Tn, r) \
                 + sum(self.disco(k*l,r) for k in range(0,Kn)))


    def npv_cost_Investments(self, m, st):
```

```python
        '''Investment function for two-stage/deterministic, including o&m costs'''
        investments = 0
        if st==1:
            for i in m.I:
                investments +=  (m.C_spe[i]*m.x[i]+ m.C_fxd[i]*m.a_i[i])
                '''Operations and maintenance costs included in investment costs:'''
                #investments +=  (m.C_run[i]*m.C_spe_0[i]*m.x[i])*self.capit(m.YRN,
                 ↪  m.R)*self.disco(1, m.R)
        else:
            investments = 0
        return investments

    def npv_cost_Operations(self, m, st):
        '''Operational costs for  two-stage/deterministic
        Summation of yearly costs for all years in YRN'''
        techrun = 0
        runcosts = 0
        gridtariff = 0
        operations = 0
        if m.lastT == 4367:
            f = 2
        elif m.lastT == 23:
            f = 8736/24
        elif m.lastT == 287:
            f = 8736/288
        elif m.lastT == 671:
            '''multiplicationfactor will be 13 if reduced model'''
            f = 13
        elif m.lastT == 727:
            f = 12
        elif m.lastT == 8735:
            f = 1

        if st == 2:
            for i in m.I:
                techrun +=  m.C_run[i]*m.C_spe_0[i]*m.x[i]
            if m.A_ep: #Grid tariff model (includes VAT): Energy pricing
                ''' Nettleie: fastledd=8.61 EUR/mnd, energiledd = 0.05 EUR/kWh (inkluderer
                 ↪  enova-avgift samt moms)'''
                gridtariff = 12*m.C_fxd_ep + m.C_spe_ep*sum(f*m.y_imp[t] for t in m.T)
            elif m.A_ps == 1: #Grid tariff model(includes VAT): Power subscription

                gridtariff =  12*(m.C_fxd_ps*(1+m.Y_max)) + m.C_pty_ps*sum(f*m.y_pty[t] for
                 ↪  t in m.T) + m.C_spe_ps*sum(f*m.y_imp[t] for t in m.T)



            VAT = 1.25
            runcosts  = sum(VAT*m.y_imp[t]*m.P_spot[t] + (m.bf_sh[t] + m.bf_dhw[t])*m.C_bf -
             ↪  m.y_exp[t]*m.P_spot[t]*m.A_exp for t in m.T)
            operations = (f*runcosts + gridtariff + techrun)*self.capit(m.YRN,
             ↪  m.R)*self.disco(1, m.R)
        else:
            operations = 0
        return operations


    def createTWOSTAGEMODEL(self):
        m = pyo.AbstractModel()
        m.name = 'ZEB stochastic two-stage model'

    # SETS ##########################################################
        m.T  = pyo.Set(doc = 'Set of all hours, full model: 8736, reduced model: 672')
        m.M = pyo.Set(doc = 'Set of all months')
        m.I = pyo.Set(doc = 'Set of all technologies')
        m.ST = pyo.Set(initialize = [1, 2], doc='STAGE')

    # PARAMETERS ####################################################

        m.lastT = pyo.Param(within=m.T, doc="Last time step")

    #---Technology costs
        m.C_fxd_0 = pyo.Param(m.I,within=pyo.NonNegativeReals, default = 0, doc='Fixed
         ↪  investment cost for all techs, EUR in year t = 0')

        m.C_spe_0 = pyo.Param(m.I,within=pyo.NonNegativeReals, default = 0,doc='Investment
         ↪  costs dependent on installed capacity, EUR/kW (EUR/kWh) in t= 0')
        m.C_run = pyo.Param(m.I,within=pyo.NonNegativeReals,default = 0,doc='Yearly running
         ↪  cost of each tech, given from investment costs EUR/kW installed')

    #---Grid Tariff pricing
        #Energy pricing
        m.A_ep = pyo.Param(within=pyo.Binary,default = 0,doc = 'Activation of energy
         ↪  pricing')
        m.C_fxd_ep = pyo.Param(within=pyo.NonNegativeReals,default = 0,doc='Fixed charge
         ↪  part of grid tariff for ep')
        m.C_spe_ep = pyo.Param(within=pyo.NonNegativeReals,default = 0,doc='Specific energy
         ↪  charge part of grid tariff for ep')

        #Power Subscription pricing
        m.A_ps = pyo.Param(within = pyo.Binary, default = 0,doc = 'Activation of power
         ↪  subscription pricing')
        m.C_fxd_ps = pyo.Param(within=pyo.NonNegativeReals,default = 0, doc='Subscriptopn
         ↪  charge for pp')
        m.C_pty_ps = pyo.Param(within=pyo.NonNegativeReals,default = 0,doc='Penalty charge
         ↪  for pp')
        m.C_spe_ps = pyo.Param(within=pyo.NonNegativeReals,default = 0,doc='Energy charge
         ↪  charge for pp')
        m.Y_max = pyo.Param(within=pyo.NonNegativeReals,default = 0,doc = 'Subscription
         ↪  limit')

        #Peak Power Pricing
        m.A_pp = pyo.Param(within=pyo.Binary)
        m.C_pp = pyo.Param(within=pyo.NonNegativeReals)
        m.C_fxd_pp = pyo.Param(within=pyo.NonNegativeReals)
        m.C_spe_pp = pyo.Param(within=pyo.NonNegativeReals)

    #---#Reference System

        #CO2-Factors
        m.A_co2 = pyo.Param(within=pyo.Binary,doc = 'Activation of co2 crediting system')
        m.G_ref = pyo.Param(within=pyo.NonNegativeReals,doc='CO2 reference emissions')
        m.G_el = pyo.Param(within=pyo.NonNegativeReals,doc='gCO2 eq. per kWh
        ↪ imported/exported')
        m.G_bf = pyo.Param(within=pyo.NonNegativeReals,doc='gCO2 eq. per kWh for technology
        ↪ i, i.e BB')

        #Primary Energy Factors
        m.A_pe = pyo.Param(within=pyo.Binary,doc = 'Activation of primary energy crediting
        ↪ system')
        m.PE_ref = pyo.Param(within=pyo.NonNegativeReals,doc='CO2 reference emissions')
        m.PE_imp = pyo.Param(within=pyo.NonNegativeReals,doc='PE per kWh imported
        ↪ electricity')
        m.PE_exp = pyo.Param(within=pyo.NonNegativeReals,doc='PE per kWh exported
        ↪ electricity')
        m.PE_bf = pyo.Param(within=pyo.NonNegativeReals,doc='PE per kWh for technology i,
        ↪ i.e BB')

    #---Technologies
        m.A_i = pyo.Param(m.I,  within=pyo.Binary,doc='Pre-activation of each tech')

        m.Eff = pyo.Param(m.I,within=pyo.NonNegativeReals, doc='Technology efficiency')
        m.Eff_ba_ch = pyo.Param(within=pyo.NonNegativeReals,doc='Battery charging
        ↪ efficiency')
        m.Eff_ba_dch = pyo.Param(initialize = 1, doc='Battery discharge efficiency')
        m.Beta_ba = pyo.Param(within=pyo.NonNegativeReals,doc='Charging/discharging rate')
        m.Beta_hs = pyo.Param(within=pyo.NonNegativeReals, doc='identical charging rate for
        ↪ heat storage')

        m.L = pyo.Param(m.I,  within=pyo.NonNegativeIntegers, doc='Lifetime of technology
        ↪ i')
        m.X_min = pyo.Param(m.I,  within=pyo.NonNegativeReals, doc='Max possible installed
        ↪ capacity of technology ')
        m.X_max = pyo.Param(m.I, within=pyo.NonNegativeReals, doc='Min possible installed
        ↪ capacity of technology ')

        m.Temp = pyo.Param(m.T, within=pyo.Reals, doc='Ambient temperature of certain hour')
        m.Y_pv = pyo.Param(m.T, within=pyo.NonNegativeReals,doc='Possible PV output at time
        ↪ t')
        m.COP_ashp_dhw = pyo.Param(m.T, within=pyo.NonNegativeReals, doc='Heat pump
        ↪ performance at time t w/r DHW')
        m.COP_gshp_dhw = pyo.Param(m.T, within=pyo.NonNegativeReals, doc='Heat pump
        ↪ performance at time t w/r DHW')
        m.COP_ashp_sh = pyo.Param(m.T, within=pyo.NonNegativeReals, doc='Heat pump
        ↪ performance at time t w/r SH')
        m.COP_gshp_sh = pyo.Param(m.T, within=pyo.NonNegativeReals, doc='Heat pump
        ↪ performance at time t w/r SH')

        #BITES-params
        m.SS_cap = pyo.Param(within=pyo.NonNegativeReals, doc='Shallow storage capacity of
        ↪ house, fully determined by area, thus a parameter instead of a variable')
        m.DS_cap = pyo.Param(within=pyo.NonNegativeReals, doc='Deep storage capacity of
        ↪ house, ---=---')
        m.K_shallow = pyo.Param(doc='loss factor for BITES, shallow part')
        m.K_deep = pyo.Param(doc='loss factor for BITES, deep part')
        m.K_flow = pyo.Param(doc='flow factor for BITES, cross-node flow')

    #---Energy Demand
```

```
        m.D_el = pyo.Param(m.T,  within=pyo.NonNegativeReals, doc='Hourly building
         ↪  electricity demand')
        m.D_sh = pyo.Param(m.T, within=pyo.NonNegativeReals, doc='Hourly building space
         ↪  heating demand')
        m.D_dhw = pyo.Param(m.T, within=pyo.NonNegativeReals, doc='Hourly building domestic
         ↪  hot water demand')

    #---Grid
        m.P_spot = pyo.Param(m.T,  within=pyo.NonNegativeReals, doc='Hourly price of
         ↪  imported electricity EUR/kWh including certificates')
        m.X_max_imp = pyo.Param(within=pyo.NonNegativeReals, doc='Maximum grid import')
        m.X_max_exp = pyo.Param( within=pyo.NonNegativeReals,doc='Maximum grid export')
        m.A_imp = pyo.Param( within=pyo.Binary, doc='Binary: Import is activated, 1/0')
        m.A_exp = pyo.Param( within=pyo.Binary,doc='Binary: Export is activated, 1/0')
        m.C_bf = pyo.Param(within=pyo.NonNegativeReals, doc='Constant price of biofuel')

    #---Control
        m.gamma = pyo.Param( within=pyo.NonNegativeReals, doc='=0 for strictly ZEB')
        m.R = pyo.Param(within=pyo.NonNegativeReals, doc='Chosen discount Rate')
        m.YRN = pyo.Param( within=pyo.NonNegativeIntegers, doc='Total years in modelling
         ↪  period')

        def npv_inv_spe(m, i):
            return self.cost(m.YRN, m.C_spe_0[i], m.L[i], m.R)
        m.C_spe = pyo.Param(m.I, rule = npv_inv_spe)

        def npv_inv_fxd(m, i):
            return self.cost(m.YRN, m.C_fxd_0[i], m.L[i], m.R)
        m.C_fxd = pyo.Param(m.I, rule = npv_inv_fxd)

    #VARIABLES ###########################################################

        # 1 STAGE : STRATEGIC VARIABLES
        m.x = pyo.Var(m.I,within = pyo.NonNegativeReals,
         ↪  doc='Optimal installed capacity (storage size), semi-continous, kW (kWh)')
        m.a_i = pyo.Var(m.I, within = pyo.Binary,
         ↪  doc='Activation binary decition for technology i, 1/0')


        #2 STAGE : OPERATIONAL VARIABLES
        m.q_hs = pyo.Var(m.T, domain = pyo.Reals,
         ↪  doc='Keeping track of HS discharge')
        m.q_eb_sh = pyo.Var(m.T,domain = pyo.NonNegativeReals,
         ↪  doc='Net heat supplied from (SH) electric boiler at time t, kWh/h')
        m.q_eb_dhw = pyo.Var(m.T,domain = pyo.NonNegativeReals,
         ↪  doc='Net heat supplied from (DHW) electric boiler at time t, kWh/h')
        m.q_bb_sh = pyo.Var(m.T,domain = pyo.NonNegativeReals,
         ↪  doc='Net heat supplied from (SH) bio boiler at time t, kWh/h')
        m.q_bb_dhw = pyo.Var(m.T,domain = pyo.NonNegativeReals,
         ↪  doc='Net heat supplied from (DHW) bio boiler at time t, kWh/h')
        m.bf_sh = pyo.Var(m.T,  domain= pyo.NonNegativeReals,
         ↪  doc='Biofuel input to bio boiler at time t for SH kWh/h')
        m.bf_dhw = pyo.Var(m.T,  domain= pyo.NonNegativeReals,
         ↪  doc='Biofuel input to bio boiler at time t for DHW kWh/h')
        m.q_ashp_sh = pyo.Var(m.T,domain=pyo.NonNegativeReals,
         ↪  doc='Net heat supplied from air source heat pump (SH) at time t, kWh/h')
        m.q_gshp_sh = pyo.Var(m.T,domain=pyo.NonNegativeReals,
          ↪  doc='Net heat supplied from ground source heat pump (SH) at time t, kWh/h')
        m.q_ashp_dhw = pyo.Var(m.T,domain=pyo.NonNegativeReals,
          ↪  doc='Net heat supplied from air source heat pump (DHW) at time t, kWh/h')
        m.q_gshp_dhw = pyo.Var(m.T,domain=pyo.NonNegativeReals,
          ↪  doc='Net heat supplied from ground source heat pump (DHW) at time t, kWh/h')

        m.z_hs = pyo.Var(m.T,  domain = pyo.NonNegativeReals,
          ↪  doc='Content in heat storage (HS) at the end of time t, kWh')
        m.z_hwt = pyo.Var(m.T,  domain = pyo.NonNegativeReals,
          ↪  doc='Content in hot water tank (HWT) at the end of time t, kWh')
        m.z_ss = pyo.Var(m.T,  domain = pyo.NonNegativeReals,
          ↪  doc='Content in shallow BITES at the end of time t, kWh')
        m.z_ds = pyo.Var(m.T,  domain = pyo.NonNegativeReals,
          ↪  doc='Content in deep BITES at the end of time t, kWh')
        m.z_ba = pyo.Var(m.T,  domain = pyo.NonNegativeReals,
          ↪  doc='Content of battery at the end of time t, kWh')
        m.Flow = pyo.Var(m.T,  domain = pyo.Reals,
          ↪  doc='Cross-node flow of two-node BITES model at time t, kWh')


        #charging variables for BITES, shallow part
        #m.z_ss_ch = pyo.Var(m.T,  domain = pyo.NonNegativeReals,
          ↪  doc='Maximum charging rate, shallow BITES at the end of time t, kWh')
        #m.z_ss_dch = pyo.Var(m.T,  domain = pyo.NonNegativeReals,
          ↪  doc='Maximum discharging rate, shallow BITES at the end of time t, kWh')
        m.q_ss = pyo.Var(m.T,  domain = pyo.Reals,                      doc='Energy released
          ↪  from shallow BITES at the end of time t, kWh')
        m.q_hwt = pyo.Var(m.T,  domain = pyo.Reals,                      doc='Energy
          ↪  released from HWT at the end of time t, kWh')


        m.y_imp = pyo.Var(m.T, domain = pyo.NonNegativeReals,
          ↪  doc='Electricity imported from grid at time t, kWh')
        m.y_exp = pyo.Var(m.T,  domain = pyo.NonNegativeReals,
          ↪  doc='Electricity exported to grid at time t, kWh')
        m.y_pv = pyo.Var(m.T, domain = pyo.NonNegativeReals,                     doc='PV
          ↪  production at time t, kWh/h')
        m.y_eb_sh = pyo.Var(m.T, domain = pyo.NonNegativeReals,
          ↪  doc='Electricity drawn from (SH) electric boiler at time t, kWh/h')
        m.y_eb_dhw = pyo.Var(m.T, domain = pyo.NonNegativeReals,
          ↪  doc='Electricity drawn from (DHW) electric boiler at time t, kWh/h')
        m.y_ashp_sh = pyo.Var(m.T, domain = pyo.NonNegativeReals,
          ↪  doc='Total electricity consumed by the AS heat pump (SH) at time t, kWh/h')
        m.y_gshp_sh = pyo.Var(m.T, domain = pyo.NonNegativeReals,
          ↪  doc='Total electricity consumed by the GS heat pump (SH) at time t, kWh/h')
        m.y_ashp_dhw = pyo.Var(m.T, domain = pyo.NonNegativeReals,
          ↪  doc='Total electricity consumed by the AS heat pump (DHW) at time t, kWh/h')
        m.y_gshp_dhw = pyo.Var(m.T, domain = pyo.NonNegativeReals,
          ↪  doc='Total electricity consumed by the GS heat pump (DHW) at time t, kWh/h')
        m.y_ch = pyo.Var(m.T, domain = pyo.NonNegativeReals,
          ↪  doc='Amount of electricity to battery (charging) at time t, kWh/h')
        m.y_dch = pyo.Var(m.T, domain = pyo.NonNegativeReals,
          ↪  doc='Amount of electricity discharge from battery at time t, kWh/h')
        m.y_pty = pyo.Var(m.T, domain = pyo.Reals,                              doc =
          ↪  'Penalty volume')


        m.y_max = pyo.Var(m.M, domain = pyo.NonNegativeReals,                              doc='max
         ↪  power for every month')

        #m.a_imp = pyo.Var(m.T,  domain = pyo.Binary,                                       doc
         ↪  ='Import activation inward time t, 1= activated')
        #m.a_exp = pyo.Var(m.T, domain= pyo.Binary,                                        doc
         ↪  ='Export actication inward time t, 1= activated')

        m.a_ch = pyo.Var(m.T,  domain = pyo.Binary,                                        doc
         ↪  ='Charging activation inward time t, 1 = activated')
        m.a_dch = pyo.Var(m.T,  domain= pyo.Binary,                                        doc
         ↪  ='Discharging activation inward time t, 1=activated')


    # CONSTRAINTS ##########################################################

    # 1 STAGE : INVESTMENTS

        #---Activation and boundary constraints
        def Tech_active(m, i, st):
            return m.x[i] <= m.a_i[i]*self.M_const
        m.Tech_active = pyo.Constraint(m.I, m.ST, rule = Tech_active)

        def Tech_Min(m, i, st):
            return m.X_min[i]*m.a_i[i] <= m.x[i]
        m.Tech_Min = pyo.Constraint(m.I, m.ST, rule= Tech_Min)

        def Tech_Max(m, i, st):
            return m.x[i] <= m.X_max[i]*m.A_i[i]
        m.Tech_Max = pyo.Constraint(m.I, m.ST, rule= Tech_Max)

    #2 STAGE : OPERATIONS

        #---Balacing constraints

        def El_Balance(m, t, st):
            return m.D_el[t] == m.y_imp[t] + m.y_pv[t] - m.y_exp[t] + m.y_dch[t] - m.y_ch[t]
             ↪  - m.y_ashp_sh[t] - m.y_gshp_sh[t] - m.y_ashp_dhw[t] - m.y_gshp_dhw[t] -
             ↪  m.y_eb_sh[t] - m.y_eb_dhw[t]
        m.El_Balance = pyo.Constraint(m.T, m.ST, rule = El_Balance)

        def SH_balance(m, t, st):
            if t == 0:
                return m.D_sh[t] + m.z_hs[t] == m.z_hs[m.lastT]*m.Eff['HS'] + m.q_ss[t] +
                 ↪  m.q_ashp_sh[t] + m.q_gshp_sh[t] + m.q_eb_sh[t] + m.q_bb_sh[t]
            else:
                return m.D_sh[t] + m.z_hs[t] == m.z_hs[t-1]*m.Eff['HS'] + m.q_ss[t] +
                 ↪  m.q_ashp_sh[t] + m.q_gshp_sh[t] + m.q_eb_sh[t] + m.q_bb_sh[t]
        m.SH_balance = pyo.Constraint(m.T, m.ST, rule = SH_balance)

        def DHW_balance(m, t, st):
            if t == 0:
                return m.D_dhw[t] + m.z_hwt[t] == m.z_hwt[m.lastT]*m.Eff['HWT'] +
                 ↪  m.q_ashp_dhw[t] + m.q_gshp_dhw[t] + m.q_eb_dhw[t] + m.q_bb_dhw[t]
            else:
                return m.D_dhw[t] + m.z_hwt[t] == m.z_hwt[t-1]*m.Eff['HWT'] +
                 ↪  m.q_ashp_dhw[t] + m.q_gshp_dhw[t] + m.q_eb_dhw[t] + m.q_bb_dhw[t]
        m.DHW_balance = pyo.Constraint(m.T, m.ST, rule = DHW_balance)

        #---Capacity
        #split heat pumps in two here as well
        #new rule, the sum of heat produced for SH and DHW must be less than/equal to
         ↪  capacity

        def ASHP_Restriction(m,t, st):
            return m.q_ashp_sh[t] + m.q_ashp_dhw[t] <= m.x['ASHP']
        m.ASHP_Restriction = pyo.Constraint(m.T, m.ST, rule = ASHP_Restriction)

        def GSHP_Restriction(m,t, st):
            return m.q_gshp_sh[t] + m.q_gshp_dhw[t] <= m.x['GSHP']
        m.GSHP_Restriction = pyo.Constraint(m.T, m.ST, rule = GSHP_Restriction)

        def EB_Restriction(m,t, st):
            return m.q_eb_sh[t] + m.q_eb_dhw[t] <= m.x['EB']
        m.EB_Restriction = pyo.Constraint(m.T, m.ST, rule = EB_Restriction)

        def BB_Restriction(m,t, st):
            return m.q_bb_sh[t] + m.q_bb_dhw[t] <= m.x['BB']
        m.BB_Restriction = pyo.Constraint(m.T, m.ST, rule = BB_Restriction)


        #---Grid equations

        def Grid_Import(m,t, st):
            return m.y_imp[t] <= m.X_max_imp
        m.Grid_Import = pyo.Constraint(m.T, m.ST,  rule=Grid_Import)

        def Grid_Export(m,t, st):
            return m.y_exp[t] <= m.X_max_exp
        m.Grid_Export = pyo.Constraint(m.T, m.ST,  rule=Grid_Export)
        '''
        def Prosumer_Balance(m,t, st):
            return m.a_imp[t] + m.a_exp[t] <= 1
        m.Prosumer_Balance = pyo.Constraint(m.T, m.ST, rule=Prosumer_Balance)
        '''


        #---Storage equations

        def HS_Restriction(m, t, st):
            return m.z_hs[t] <= m.x['HS']
        m.HS_Restriction = pyo.Constraint(m.T, m.ST, rule=HS_Restriction)

        def HWT_Restriction(m, t, st):
            return m.z_hwt[t] <= m.x['HWT']
        m.HWT_Restriction = pyo.Constraint(m.T, m.ST, rule=HWT_Restriction)

        def SS_Restriction(m, t, st):
            return m.z_ss[t] <= m.x['SS']*m.A_i['SS']
        m.SS_Restriction = pyo.Constraint(m.T, m.ST, rule=SS_Restriction)

        def DS_Restriction(m, t, st):
            return m.z_ds[t] <= m.x['DS']*m.A_i['DS']
        m.DS_Restriction = pyo.Constraint(m.T, m.ST, rule=DS_Restriction)

        def SS_charge_active(m,t, st):
            return m.q_ss[t] <= m.z_ss[t]
        m.SS_charge_active = pyo.Constraint(m.T, m.ST, rule=SS_charge_active)

        def SS_discharge_active(m,t, st):
            return - m.z_ss[t] <= m.q_ss[t]
        m.SS_discharge_active = pyo.Constraint(m.T, m.ST, rule=SS_discharge_active)

        def HS_charge_active(m,t, st):
            return m.q_hs[t] <= m.z_hs[t]
        m.HS_charge_active = pyo.Constraint(m.T, m.ST, rule=HS_charge_active)

        def HS_discharge_active(m,t, st):
            return - m.z_hs[t] <= m.q_hs[t]
        m.HS_discharge_active = pyo.Constraint(m.T, m.ST, rule=HS_discharge_active)

        def HWT_charge_active(m,t, st):
            return m.q_hwt[t] <= m.z_hwt[t]
        m.HWT_charge_active = pyo.Constraint(m.T, m.ST, rule=HWT_charge_active)

        def HWT_discharge_active(m,t, st):
            return - m.z_hwt[t] <= m.q_hwt[t]
        m.HWT_discharge_active = pyo.Constraint(m.T, m.ST, rule=HWT_discharge_active)

        '''
        def SS_charge_active(m, t, st):
            if m.Temp[t] >= 15:
                return m.q_ss[t] >= -m.x['SS']
            elif m.Temp[t] < 15 and m.Temp[t] > -15:
                return m.q_ss[t] <= m.x['SS'] * (1 - (15 - m.Temp[t])/30)
            else:
                return m.q_ss[t] >= 0
        m.SS_charge_active = pyo.Constraint(m.T, m.ST, rule = SS_charge_active)

        def SS_discharge_active(m, t, st):
            if m.Temp[t] >= 15:
                return m.q_ss[t] <= 0
            elif m.Temp[t] < 15 and m.Temp[t] > -15:
                return m.q_ss[t] >= -(m.x['SS'] * ((15 - m.Temp[t])/30))
            else:
                return m.q_ss[t] <= m.x['SS']
        m.SS_discharge_active = pyo.Constraint(m.T, m.ST, rule = SS_charge_active)
        '''

        def HS_Balance_ch(m,t,st):
            if t == 0:
                return m.q_hs[t] == m.z_hs[m.lastT] - m.z_hs[t]
            else:
                return m.q_hs[t] == m.z_hs[t-1] - m.z_hs[t]
        m.HS_Balance_ch = pyo.Constraint(m.T, m.ST, rule = HS_Balance_ch)

        def HWT_Balance_ch(m,t,st):
            if t == 0:
                return m.q_hwt[t] == m.z_hwt[m.lastT] - m.z_hwt[t]
            else:
                return m.q_hwt[t] == m.z_hwt[t-1] - m.z_hwt[t]
        m.HWT_Balance_ch = pyo.Constraint(m.T, m.ST, rule = HWT_Balance_ch)

        def Flow_Constraint(m,t,st):
            return m.Flow[t] == m.K_flow*((m.z_ss[t]/m.SS_cap) - (m.z_ds[t]/m.DS_cap))
        m.Flow_Constraint = pyo.Constraint(m.T, m.ST, rule = Flow_Constraint)

        def SS_balance_ch(m, t, st):
            if t == 0:
                return m.q_ss[t] == m.z_ss[m.lastT] - m.z_ss[t] - m.Flow[t] - m.z_ss[t]*(1 -
                 ↪ m.K_shallow)
            else:
                return m.q_ss[t] == m.z_ss[t-1] - m.z_ss[t] - m.Flow[t] - m.z_ss[t]*(1 -
                 ↪ m.K_shallow)
        m.SS_balance_ch = pyo.Constraint(m.T, m.ST, rule = SS_balance_ch)

        def DS_balance_ch(m, t, st):
            if t == 0:
                return m.z_ds[t] == m.z_ds[m.lastT] + m.Flow[t] - m.z_ds[t]*(1 - m.K_deep)
            else:
                return m.z_ds[t] == m.z_ds[t-1] + m.Flow[t] - m.z_ds[t]*(1 - m.K_deep)
        m.DS_balance_ch = pyo.Constraint(m.T, m.ST, rule = DS_balance_ch)

        def HS_discharge_rate_min(m,t, st):
               return -m.x['HS']*m.Beta_hs <= m.q_hs[t]
        m.HS_discharge_rate_min = pyo.Constraint(m.T, m.ST, rule=HS_discharge_rate_min)

        def HS_discharge_rate_max(m,t, s):
            return m.q_hs[t] <= m.x['HS']*m.Beta_hs
        m.HS_discharge_rate_max = pyo.Constraint(m.T, m.ST, rule=HS_discharge_rate_max)

        def HWT_discharge_rate_min(m,t, st):
               return -m.x['HWT']*m.Beta_hs <= m.q_hwt[t]
        m.HWT_discharge_rate_min = pyo.Constraint(m.T, m.ST, rule=HWT_discharge_rate_min)

        def HWT_discharge_rate_max(m,t, s):
            return m.q_hwt[t] <= m.x['HWT']*m.Beta_hs
        m.HWT_discharge_rate_max = pyo.Constraint(m.T, m.ST, rule=HWT_discharge_rate_max)

        def BA_Restriction(m,t, st):
            return m.z_ba[t] <= m.x['BA']
        m.BA_restriction = pyo.Constraint(m.T, m.ST, rule=BA_Restriction)

        def BA_Balance(m,t, st):
            if t == 0:
                return m.z_ba[t] == m.z_ba[m.lastT] - m.y_dch[t]*(1/m.Eff_ba_dch) +
                 ↪ m.y_ch[t]*m.Eff_ba_ch
            else:
                return m.z_ba[t] == m.z_ba[t-1] - m.y_dch[t]*(1/m.Eff_ba_dch) +
                 ↪ m.y_ch[t]*m.Eff_ba_ch
        m.BA_Balance = pyo.Constraint(m.T, m.ST, rule=BA_Balance)

        def BA_Charge_Balance(m, t, st):
```

```
        if t == 0:
            return m.y_ch[t] <= (m.x['BA'] -
             ↪  m.z_ba[m.lastT])*m.A_i['BA']*(1/m.Eff_ba_ch)
        else:
            return m.y_ch[t] <= (m.x['BA'] - m.z_ba[t-1])*m.A_i['BA']*(1/m.Eff_ba_ch)
    m.BA_Charge_Balance = pyo.Constraint(m.T, m.ST, rule=BA_Charge_Balance)

    def BA_Discharge_Balance(m,t, st):
        if t == 0:
            return m.y_dch[t] <= m.z_ba[m.lastT]*m.A_i['BA']*m.Eff_ba_dch
        else:
            return m.y_dch[t] <= m.z_ba[t-1]*m.A_i['BA']*m.Eff_ba_dch
    m.BA_Discharge_Balance = pyo.Constraint(m.T, m.ST, rule=BA_Discharge_Balance)

    def BA_charge_active(m,t, st):
        return m.y_ch[t] <= m.X_max_imp*m.a_ch[t]
    m.BA_charge_active = pyo.Constraint(m.T, m.ST, rule=BA_charge_active)

    def BA_discharge_active(m,t, st):
        return m.y_dch[t] <= m.X_max_imp*m.a_dch[t]
    m.BA_discharge_active = pyo.Constraint(m.T, m.ST, rule=BA_discharge_active)

    def Battery_Balance(m,t, st):
        return m.a_ch[t] + m.a_dch[t] <= 1
    m.Battery_Balance = pyo.Constraint(m.T, m.ST, rule=Battery_Balance)

    def BA_charge_rate(m,t, st):
        return m.y_ch[t] <= m.x['BA']*m.Beta_ba
    m.BA_charge_rate = pyo.Constraint(m.T, m.ST, rule=BA_charge_rate)

    def BA_discharge_rate(m,t, st):
        return m.y_dch[t] <= m.x['BA']*m.Beta_ba
    m.BA_discharge_rate = pyo.Constraint(m.T, m.ST, rule=BA_discharge_rate)

    #---Production constraints for generating technologies

    def PV_Balance(m,t, st):
        return m.y_pv[t] == m.x['PV']*m.Y_pv[t]
    m.PV_Balance = pyo.Constraint(m.T, m.ST, rule=PV_Balance)

    def ASHP_SH_Balance(m,t, st):
        return m.q_ashp_sh[t] == m.y_ashp_sh[t]*m.COP_ashp_sh[t]
    m.ASHP_SH_Balance = pyo.Constraint(m.T, m.ST, rule = ASHP_SH_Balance)

    def ASHP_DHW_Balance(m,t, st):
        return m.q_ashp_dhw[t] == m.y_ashp_dhw[t]*m.COP_ashp_dhw[t]
    m.ASHP_DHW_Balance = pyo.Constraint(m.T, m.ST, rule = ASHP_DHW_Balance)

    def GSHP_SH_Balance(m,t, st):
        return m.q_gshp_sh[t] == m.y_gshp_sh[t]*m.COP_gshp_sh[t]
    m.GSHP_SH_Balance = pyo.Constraint(m.T, m.ST, rule = GSHP_SH_Balance)

    def GSHP_DHW_Balance(m,t, st):
        return m.q_gshp_dhw[t] == m.y_gshp_dhw[t]*m.COP_gshp_dhw[t]
    m.GSHP_DHW_Balance = pyo.Constraint(m.T, m.ST, rule = GSHP_DHW_Balance)

```

```python
        def EB_SH_Balance(m,t, st):
            return m.q_eb_sh[t] == m.y_eb_sh[t]*m.Eff['EB']
        m.EB_SH_Balance = pyo.Constraint(m.T, m.ST, rule = EB_SH_Balance)

        def EB_DHW_Balance(m,t, st):
            return m.q_eb_dhw[t] == m.y_eb_dhw[t]*m.Eff['EB']
        m.EB_DHW_Balance = pyo.Constraint(m.T, m.ST, rule = EB_DHW_Balance)

        def BB_SH_Balance(m,t, st):
            return m.q_bb_sh[t] == m.bf_sh[t]*m.A_i['BB']*m.Eff['BB']
        m.BB_SH_Balance = pyo.Constraint(m.T, m.ST, rule=BB_SH_Balance)

        def BB_DHW_Balance(m,t, st):
            return m.q_bb_dhw[t] == m.bf_dhw[t]*m.A_i['BB']*m.Eff['BB']
        m.BB_DHW_Balance = pyo.Constraint(m.T, m.ST, rule=BB_DHW_Balance)


        #---Zero emission/energy constraints
        def ZE_Balance(m):
            if m.A_co2==1:
                print('ACTIVE ZEB-carbon RESTRICTION')
                if m.lastT == 8735:
                    return sum(m.y_imp[t]*m.G_el + m.bf_sh[t]*m.G_bf + m.bf_dhw[t]*m.G_bf for t in m.T) <= sum(m.y_exp[t]*m.G_el for t in m.T)    #what about CO2-factors for DH-import? Diverse mix, assuming spillover heat from industry with factor 0 here.
                ''' quarterly balancing
                if m.lastT == 8735:
                    for i in range(4):
                        sum(m.y_imp[t]*m.G_el - m.y_exp[t]*m.G_el for t in range(i*2184,(i+1)*2184)) <= m.G_ref*m.gamma
                '''

                if m.lastT == 4367:
                    return 2*sum(m.y_imp[t]*m.G_el - m.y_exp[t]*m.G_el + m.bf[t]*m.G_bf for t in m.T) <= m.G_ref*m.gamma
                elif m.lastT == 727:
                    return 12*sum(m.y_imp[t]*m.G_el - m.y_exp[t]*m.G_el + m.bf[t]*m.G_bf for t in m.T) <= m.G_ref*m.gamma
                elif m.lastT == 671:
                    return sum(m.y_imp[t]*m.G_el + m.bf_sh[t]*m.G_bf + m.bf_dhw[t]*m.G_bf for t in m.T) <= sum(m.y_exp[t]*m.G_el for t in m.T)
                elif m.lastT == 287:
                    return (8736/288)*sum(m.y_imp[t]*m.G_el - m.y_exp[t]*m.G_el + m.bf[t]*m.G_bf for t in m.T) <= m.G_ref*m.gamma
                elif m.lastT == 23:
                    return (8736/24)*sum(m.y_imp[t]*m.G_el - m.y_exp[t]*m.G_el + m.bf[t]*m.G_bf for t in m.T) <= m.G_ref*m.gamma
            elif m.A_pe == 1:
                print('ACTIVE ZEB-pef RESTRICTION')
                if m.lastT == 8735:
                    return sum(m.y_imp[t]*m.PE_imp - m.y_exp[t]*m.PE_exp + m.bf[t]*m.PE_bf for t in m.T) <= m.PE_ref*m.gamma
                elif m.lastT == 671:
                    return 13*sum(m.y_imp[t]*m.PE_imp - m.y_exp[t]*m.PE_exp + m.bf[t]*m.PE_bf for t in m.T) <= m.PE_ref*m.gamma
            else:
                print('NO ZEB RESTRICTION')
                return pyo.Constraint.Skip
        m.ZE_Balance = pyo.Constraint(rule=ZE_Balance)


    #Subscription power pricing Constraint
        def pty_volume(m, t): #counting all power within one hour exceeding max limit
            if m.A_ps == 1:
                return m.y_imp[t] - m.Y_max <= m.y_pty[t]
            else:
                return m.y_pty[t] ==0
        m.pty_volume = pyo.Constraint(m.T, rule = pty_volume)

        def pty_volume2(m, t):
                return 0 <= m.y_pty[t]
        m.pty_volume2 = pyo.Constraint(m.T, rule = pty_volume2)

    #   OBJECTIVE FUNCTION  ##################################################

        def cost_Investments_rule(m, st):
            expr = self.npv_cost_Investments(m, st)
            return expr
        m.cost_Investments = pyo.Expression(m.ST, rule = cost_Investments_rule)

        def cost_Operation_rule(m, st):
            expr = self.npv_cost_Operations(m, st)
            return expr
        m.cost_Operations=  pyo.Expression(m.ST, rule = cost_Operation_rule)

        def objective_TotalCost(m):
            expr = pyo.summation(m.cost_Investments) + pyo.summation(m.cost_Operations)
            return expr
        m.objective_TotalCost = pyo.Objective(rule = objective_TotalCost, sense =
         ↪  pyo.minimize)

        return m

    def createConcreteModeltwoStage(self, data):
            '''Function creating instance from input data'''
            #data = self.writeStochasticProblem(data)
            concretemodel = self.abstractmodel.create_instance(data = {'mymodel':data},
             ↪  namespace = 'mymodel')
            return concretemodel
```

The following code, Loading_Data.py, contains functions for the waterborne model loading the relevant data from an excel spreadsheet and writing the results to another spreadsheet.

```python
import xlrd
import pandas as pd
import time
from selection import selection

def createModelData(path_in, year):
    #Creates all input data for reduced model
    wb = xlrd.open_workbook(path_in)
    ws1 = wb.sheet_by_name('Input_Tech')
    ws2 = wb.sheet_by_name('Input_Parameters')

    di = {}
    #Sets:
    di['T'] = {None: list(range(0, int(ws2.cell(16, 2).value) + 1))}
    di['lastT'] = {None: (ws2.cell(16, 2).value)}
    di['ST'] = {None:[1, 2]}

    for row in (15, 16, 17, 18, 19, 20, 21, 22, 26, 27, 28, 29, 30, 31, 32, \
                33, 34, 35, 36, 37, 39, 40, 41, 42, 45, 46, 47, 52, 53, \
                54, 57, 59, 60, 61):
        if row in [15, 16, 21, 22, 26, 30, 33, 39]:
            di[ws2.cell(row,1).value] = {None: int(ws2.cell(row,2).value)}

        else:
            di[ws2.cell(row,1).value] = {None: ws2.cell(row,2).value}

    #important parameters:
    I = []
    for row in range(11):
        I.append(ws1.cell(4+row,1).value)
    di['I'] = {None:I}

    for col in (2, 3, 4, 5, 6, 7, 8, 9):
        data = {}
        for row in range(11):
            if col in [2,6]:
                data[I[row]] = int(ws1.cell(4+row,col).value)
            else:
                data[I[row]] = ws1.cell(4+row,col).value
            di[ws1.cell(2, col).value] = data

    if ws2.cell(16, 2).value == 8735:
        df = pd.read_excel('Scenario_data.xlsx', str(year), usecols =
         ↪  [3,4,5,6,7,8,18,19,20,21]) #Now with separated COPs for DHW and SH
        df = df.drop([0])
        df = df.drop(range(8737,8761))
        new_df = pd.DataFrame(data=df.values, columns =
         ↪  ['D_el','D_sh','D_dhw','P_spot','Temp','Y_pv','COP_ashp_sh',
         ↪  'COP_ashp_wh','COP_gshp_sh', 'COP_gshp_wh'], index = range(8736))
        data = new_df.to_dict()

        di['D_el'] = data['D_el']
        di['D_sh'] = data['D_sh']
        di['D_dhw'] = data['D_dhw']
        di['P_spot'] = data['P_spot']
        di['Temp'] = data['Temp']
        di['Y_pv'] = data['Y_pv']
        di['COP_ashp_sh'] = data['COP_ashp_sh']
        di['COP_ashp_dhw'] = data['COP_ashp_wh']
        di['COP_gshp_sh'] = data['COP_gshp_sh']
        di['COP_gshp_dhw'] = data['COP_gshp_wh']

        return di

    elif ws2.cell(16, 2).value == 671:
        #select 4 weeks for each season surrounding the highest space heating load, week
         ↪  size is 168 hours
        df = pd.read_excel('Scenario_data.xlsx', str(year), usecols =
         ↪  [3,4,5,6,7,8,18,19,20,21]) #Now with separated COPs for DHW and SH
        df = df.drop([0])
        df = df.drop(range(8737,8761))
        select_dict = selection(str(year), 'Scenario_data.xlsx')
        df = df.drop(range(1,select_dict['winter_start']))
        df = df.drop(range(select_dict['winter_start'] + 168 ,select_dict['spring_start']))
        df = df.drop(range(select_dict['spring_start'] + 168 ,select_dict['summer_start']))
        df = df.drop(range(select_dict['summer_start'] + 168 ,select_dict['fall_start']))
        df = df.drop(range(select_dict['fall_start'] + 168 , 8737))

        new_df = pd.DataFrame(data=df.values, columns =
         ↪  ['D_el','D_sh','D_dhw','P_spot','Temp','Y_pv','COP_ashp_sh',
         ↪  'COP_ashp_wh','COP_gshp_sh', 'COP_gshp_wh'], index = range(672))
        data = new_df.to_dict()

        di['D_el'] = data['D_el']
        di['D_sh'] = data['D_sh']
        di['D_dhw'] = data['D_dhw']
        di['P_spot'] = data['P_spot']
        di['Temp'] = data['Temp']
        di['Y_pv'] = data['Y_pv']
        di['COP_ashp_sh'] = data['COP_ashp_sh']
        di['COP_ashp_dhw'] = data['COP_ashp_wh']
        di['COP_gshp_sh'] = data['COP_gshp_sh']
        di['COP_gshp_dhw'] = data['COP_gshp_wh']

        return di

def saveDeterministicResults(m, result_file):

    from openpyxl import load_workbook

    wb = load_workbook(result_file)
    ser = wb['Series']
    val = wb['Values']
    if m.lastT == 23:
        f = 8736/24
    elif m.lastT == 287:
        f = 8736/288
    elif m.lastT == 8735:
        f = 1
    elif m.lastT==671:
        f = 13

    for t in m.T:
        for i, elem in enumerate(['D_el', 'y_imp', 'y_exp', 'y_pty', 'y_ch', 'y_dch', \
                                  'z_ba', 'y_pv', 'y_ashp_sh', 'y_ashp_dhw', 'y_gshp_sh',\
                                  'y_gshp_dhw', 'y_eb_sh','y_eb_dhw', 'D_sh', 'D_dhw', \
                                  'q_hs', 'q_ashp_sh','q_ashp_dhw',
                                   ↪  'q_gshp_sh','q_gshp_dhw',\
                                  'q_eb_sh', 'q_eb_dhw', 'z_hs', 'q_ss', 'z_ss','z_ds',
                                   ↪  'z_hwt', 'q_hwt', 'q_bb_sh', 'q_bb_dhw', 'Flow']):
            ser.cell(row = 1, column = i + 1, value = elem)

        ser.cell(row= t+2, column=1, value=m.D_el[t])
        ser.cell(row= t+2, column=2, value=m.y_imp[t].value)
        ser.cell(row= t+2, column=3, value=m.y_exp[t].value)
        ser.cell(row= t+2, column=4, value=m.y_pty[t].value)

        ser.cell(row= t+2, column=5, value=m.y_ch[t].value)
        ser.cell(row= t+2, column=6, value=m.y_dch[t].value)
        ser.cell(row= t+2, column=7, value=m.z_ba[t].value)


        ser.cell(row= t+2, column=8, value=m.y_pv[t].value)
        ser.cell(row= t+2, column=9, value=m.y_ashp_sh[t].value)
        ser.cell(row= t+2, column=10, value=m.y_ashp_dhw[t].value)
        ser.cell(row= t+2, column=11, value=m.y_gshp_sh[t].value)
        ser.cell(row= t+2, column=12, value=m.y_gshp_dhw[t].value)
        ser.cell(row= t+2, column=13, value=m.y_eb_sh[t].value)
        ser.cell(row= t+2, column=14, value=m.y_eb_dhw[t].value)
        ser.cell(row= t+2, column=15, value=m.D_sh[t])
        ser.cell(row= t+2, column=16, value=m.D_dhw[t])
        ser.cell(row= t+2, column=17, value=m.q_hs[t].value) #HS output
        ser.cell(row= t+2, column=18, value=m.q_ashp_sh[t].value)
        ser.cell(row= t+2, column=19, value=m.q_ashp_dhw[t].value)
        ser.cell(row= t+2, column=20, value=m.q_gshp_sh[t].value)
        ser.cell(row= t+2, column=21, value=m.q_gshp_dhw[t].value)
        ser.cell(row= t+2, column=22, value=m.q_eb_sh[t].value)
        ser.cell(row= t+2, column=23, value=m.q_eb_dhw[t].value)
        ser.cell(row= t+2, column=24, value=m.z_hs[t].value)
        ser.cell(row= t+2, column=25, value=m.q_ss[t].value)
        ser.cell(row= t+2, column=26, value=m.z_ss[t].value)
        ser.cell(row= t+2, column=27, value=m.z_ds[t].value)
        ser.cell(row= t+2, column=28, value=m.z_hwt[t].value)
        ser.cell(row= t+2, column=29, value=m.q_hwt[t].value)
        ser.cell(row= t+2, column=30, value=m.q_bb_sh[t].value)
        ser.cell(row= t+2, column=31, value=m.q_bb_dhw[t].value)
        ser.cell(row= t+2, column=32, value=m.Flow[t].value)

    #Invested capacities
    val['A1'] = 'Time'
    val['B1'] = time.ctime() #Time and date
    #val['C4'] = time.time() - start_time #Code run time
    val['A2'] = 'Scenario name'
    val['B2'] = 'NOR'
    val['A3'] = 'gamma'
    val['B3'] = m.gamma.value
    val['A4'] = 'A_co2'
    val['B4'] = m.A_co2.value
    val['A5'] = 'A_pe'
    val['B5'] = m.A_pe.value

    val['B6'] = m.x['PV'].value
    val['B7'] = m.x['BA'].value
    val['B8'] = m.x['DH'].value
    val['B9'] = m.x['EB'].value
    val['B10'] = m.x['BB'].value
    val['B11'] = m.x['GSHP'].value
    val['B12'] = m.x['ASHP'].value
    val['B13'] = m.x['HS'].value
    val['B14'] = m.SS_cap.value
    val['B15'] = m.DS_cap.value
    val['B16'] = m.x['HWT'].value
    #val['B15'] = m.x['SS'].value

    val['A6'] = 'pv'
    val['A7'] = 'ba'
    val['A8'] = 'dh'
    val['A9'] = 'eb'
    val['A10'] = 'bb'
    val['A11'] = 'gshp'
    val['A12'] = 'ashp'
    val['A13'] = 'hs'
    val['A14'] = 'ss'
    val['A15'] = 'ds'
    val['A16'] = 'hwt'
    #val['A15'] = 'ss'

    val['C5'] = 'Tech inv cost'
    val['C6'] = (m.C_spe['PV']*m.x['PV'].value+ m.C_fxd['PV']*m.a_i['PV'].value)
    val['C7'] = (m.C_spe['BA']*m.x['BA'].value+ m.C_fxd['BA']*m.a_i['BA'].value)
    val['C8'] = (m.C_spe['DH']*m.x['DH'].value+ m.C_fxd['DH']*m.a_i['DH'].value)
    val['C9'] = (m.C_spe['EB']*m.x['EB'].value+ m.C_fxd['EB']*m.a_i['EB'].value)
    val['C10'] = (m.C_spe['BB']*m.x['BB'].value+ m.C_fxd['BB']*m.a_i['BB'].value)
    val['C11'] = (m.C_spe['GSHP']*m.x['GSHP'].value+ m.C_fxd['GSHP']*m.a_i['GSHP'].value)
    val['C12'] = (m.C_spe['ASHP']*m.x['ASHP'].value+ m.C_fxd['ASHP']*m.a_i['ASHP'].value)
    val['C13'] = (m.C_spe['HS']*m.x['HS'].value+ m.C_fxd['HS']*m.a_i['HS'].value)
    val['C14'] = 0
    val['C15'] = 0
    val['C16'] = (m.C_spe['HWT']*m.x['HWT'].value+ m.C_fxd['HWT']*m.a_i['HWT'].value)

    val['D1'] = m.objective_TotalCost()
    val['D2'] = m.cost_Investments[1]()
    val['D3'] = m.cost_Operations[2]()

    val['C1'] = 'tot_cost'
    val['C2'] = 'inv_cost'
    val['C3'] = 'op_cost'

    y_imp_tot = 0
    y_exp_tot = 0
    bf_tot = 0
```

```python
    for t in m.T:
        y_imp_tot += m.y_imp[t].value
        y_exp_tot += m.y_exp[t].value
        bf_tot += m.bf_sh[t].value + m.bf_dhw[t].value

    G_tot = m.G_bf*bf_tot + m.G_el*y_imp_tot

    val['F1'] = 'y_imp_tot'
    val['F2'] = 'y_exp_tot'
    val['F3'] = 'bf_tot'
    val['F4'] = 'G_tot'

    val['G1'] = y_imp_tot
    val['G2'] = y_exp_tot
    val['G3'] = bf_tot
    val['G4'] = G_tot

    val['F8'] = 'BITES-params'
    val['F9'] = 'K_flow'
    val['F10'] = 'K_shallow'
    val['F11'] = 'K_deep'

    val['G9'] = m.K_flow.value
    val['G10'] = m.K_shallow.value
    val['G11'] = m.K_deep.value

    y_imp_max = 0
    for t in m.T:
        if m.y_imp[t].value > y_imp_max:
            y_imp_max = m.y_imp[t].value

    val['F13'] = 'peak load el'
    val['G13'] = y_imp_max

    wb.save(result_file)
    print('Deterministic result file successfully created')
```

The following code is script running the optimization for the waterborne model, which loads the two previous code fragments as modules:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 12 20:17:29 2019

@author: marius
"""

import MODEL_wb_SH as model
import Loading_Data as ld
import pyomo.environ as pyo
import xlrd
import time
from pyomo.environ import SolverFactory

year = 2012
path_in = 'allData.xlsx'
path_out = 'results/results_' + str(year) + '.xlsx'

abstract = model.ZEBModel()
data = ld.createModelData(path_in, year)
data['SS_cap'] = {None: 12.5}
data['DS_cap'] = {None: 90}

instance = abstract.abstractmodel.create_instance(data={'mymodel':data},namespace='mymodel')
opt = pyo.SolverFactory('gurobi')
#opt.options['TimeLimit'] = 3600
#opt.options['BestObjStop'] = 65000
results = opt.solve(instance, tee=True, #to stream the solver output
                                                keepfiles=True, #print the LP file for
                                                 ↪  examination
                             symbolic_solver_labels=False) # use human readable names
ld.saveDeterministicResults(instance, path_out)
print('Objective value', instance.objective_TotalCost())
```

Appendix B

This appendix contains the code for the point-source model. The following python script, MODEL_ph_SH.py, defines the point-source model:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 12 20:17:29 2019

@author: marius
"""

import pyomo.environ as pyo
from math import floor

class ZEBModel():

    def __init__(self, M_const = 1000):
        """Create Abstract Pyomo model for ZEB
        """
        # Deterministic two-stage model
        self.abstractmodel = self.createTWOSTAGEMODEL()

        self.M_const = M_const

    def disco(self, n, r):
        '''Discounting factor'''
        return 1/((1+r)**n)

    def annui(self, n, r):
        '''Annuity factor'''
        return r/(1-(1+r)**(-n))

    def capit(self, n, r): # Capitalization factor
        '''Capitalization factor'''
        return (1-(1+r)**(-n))/r

    def cost(self, Yn, cost, l, r):
        '''For the two-stage model: calculating forced reinvestment costs'''
        Kn = pyo.floor(Yn/(l*1))
        n = Yn-l*Kn
        Tn = Yn-n
        return cost*(self.annui(l,r)*self.capit(n,r)*self.disco(Tn, r) \
                 + sum(self.disco(k*l,r) for k in range(0,Kn)))


    def npv_cost_Investments(self, m, st):
        '''Investment function for two-stage/deterministic, including o&m costs'''
        investments = 0
        if st==1:
            for i in m.I:
                investments +=  (m.C_spe[i]*m.x[i]+ m.C_fxd[i]*m.a_i[i])
                '''Operations and maintenance costs included in investment costs:'''
                #investments +=  (m.C_run[i]*m.C_spe_0[i]*m.x[i])*self.capit(m.YRN, m.R)*self.disco(1, m.R)
        else:
            investments = 0
        return investments

    def npv_cost_Operations(self, m, st):
        '''Operational costs for  two-stage/deterministic
        Summation of yearly costs for all years in YRN'''
        techrun = 0
        runcosts = 0
        gridtariff = 0
        operations = 0
        if m.lastT == 4367:
            f = 2
        elif m.lastT == 23:
            f = 8736/24
        elif m.lastT == 287:
            f = 8736/288
        elif m.lastT == 671:
            '''multiplication factor will be 13 if reduced model'''
            f = 13
        elif m.lastT == 727:
            f = 12
        elif m.lastT == 8735:
            f = 1

        if st == 2:
            for i in m.I:
                techrun +=  m.C_run[i]*m.C_spe_0[i]*m.x[i]
            if m.A_ep: #Grid tariff model (includes VAT): Energy pricing
                ''' Nettleie: fastledd=8.61 EUR/mnd, energiledd = 0.05 EUR/kWh (inkluderer enova-avgift samnt moms)'''
                gridtariff = 12*m.C_fxd_ep + m.C_spe_ep*sum(f*m.y_imp[t] for t in m.T)
            elif m.A_ps == 1: #Grid tariff model(includes VAT): Power subscription

                gridtariff =  12*(m.C_fxd_ps*(1+m.Y_max)) + m.C_pty_ps*sum(f*m.y_pty[t] for t in m.T) + m.C_spe_ps*sum(f*m.y_imp[t] for t in m.T)

            VAT = 1.25
            runcosts  = sum(m.f_fp[t]*m.C_wo + VAT*m.y_imp[t]*m.P_spot[t] - m.y_exp[t]*m.P_spot[t]*m.A_exp for t in m.T)
            operations = (f*runcosts + gridtariff + techrun)*self.capit(m.YRN, m.R)*self.disco(1, m.R)
        else:
            operations = 0
        return operations


    def createTWOSTAGEMODEL(self):
        m = pyo.AbstractModel()
        m.name = 'ZEB stochastic two-stage model'

    # SETS ###########################################################
        m.T  = pyo.Set(doc = 'Set of all hours, full model: 8736, reduced model: 672')
        m.M = pyo.Set(doc = 'Set of all months')
        m.I = pyo.Set(doc = 'Set of all technologies')
        m.ST = pyo.Set(initialize = [1, 2], doc='STAGE')

    # PARAMETERS ###########################################################

        m.lastT = pyo.Param(within=m.T, doc="Last time step")

    #---Technology costs
        m.C_fxd_0 = pyo.Param(m.I,within=pyo.NonNegativeReals, default = 0, doc='Fixed
         ↪  investment cost for all techs, EUR in year t = 0')

        m.C_spe_0 = pyo.Param(m.I,within=pyo.NonNegativeReals, default = 0,doc='Investment
         ↪  costs dependent on installed capacity, EUR/kW (EUR/kWh) in t= 0')
        m.C_run = pyo.Param(m.I,within=pyo.NonNegativeReals,default = 0,doc='Yearly running
         ↪  cost of each tech, given from investment costs EUR/kW installed')

    #---Grid Tariff pricing
        #Energy pricing
        m.A_ep = pyo.Param(within=pyo.Binary,default = 0,doc = 'Activation of energy
         ↪  pricing')
        m.C_fxd_ep = pyo.Param(within=pyo.NonNegativeReals,default = 0,doc='Fixed charge
         ↪  part of grid tariff for ep')
        m.C_spe_ep = pyo.Param(within=pyo.NonNegativeReals,default = 0,doc='Specific energy
         ↪  charge part of grid tariff for ep')

        #Power Subscription pricing
        m.A_ps = pyo.Param(within = pyo.Binary, default = 0,doc = 'Activation of power
         ↪  subscription pricing')
        m.C_fxd_ps = pyo.Param(within=pyo.NonNegativeReals,default = 0, doc='Subscriptopn
         ↪  charge for pp')
        m.C_pty_ps = pyo.Param(within=pyo.NonNegativeReals,default = 0,doc='Penalty charge
         ↪  for pp')
        m.C_spe_ps = pyo.Param(within=pyo.NonNegativeReals,default = 0,doc='Energy charge
         ↪  charge for pp')
        m.Y_max = pyo.Param(within=pyo.NonNegativeReals,default = 0,doc = 'Subscription
         ↪  limit')

        #Peak Power Pricing
        m.A_pp = pyo.Param(within=pyo.Binary)
        m.C_pp = pyo.Param(within=pyo.NonNegativeReals)
        m.C_fxd_pp = pyo.Param(within=pyo.NonNegativeReals)
        m.C_spe_pp = pyo.Param(within=pyo.NonNegativeReals)

    #---#Reference System

        #CO2-Factors
        m.A_co2 = pyo.Param(within=pyo.Binary,doc = 'Activation of co2 crediting system')
        m.G_ref = pyo.Param(within=pyo.NonNegativeReals,doc='CO2 reference emissions')
        m.G_el_imp = pyo.Param(within=pyo.NonNegativeReals,doc='gCO2 eq. per kWh
         ↪  imported/exported')
        m.G_el_exp = pyo.Param(within=pyo.NonNegativeReals,doc='gCO2 eq. per kWh
         ↪  imported/exported')
        m.G_bf = pyo.Param(within=pyo.NonNegativeReals,doc='gCO2 eq. per kWh for technology
         ↪  i, i.e BB')
        m.G_wo = pyo.Param(within=pyo.NonNegativeReals,doc='gCO2 eq. per kWh for wood
         ↪  technology')
        m.G_tot = pyo.Param(within=pyo.NonNegativeReals,doc='gCO2 total for noZEB')

        #Primary Energy Factors
        m.A_pe = pyo.Param(within=pyo.Binary,doc = 'Activation of primary energy crediting
         ↪  system')
        m.PE_ref = pyo.Param(within=pyo.NonNegativeReals,doc='CO2 reference emissions')
        m.PE_imp = pyo.Param(within=pyo.NonNegativeReals,doc='PE per kWh imported
         ↪  electricity')
        m.PE_exp = pyo.Param(within=pyo.NonNegativeReals,doc='PE per kWh exported
         ↪  electricity')
        m.PE_bf = pyo.Param(within=pyo.NonNegativeReals,doc='PE per kWh for technology i,
         ↪  i.e BB')

    #---Technologies
        m.A_i = pyo.Param(m.I,  within=pyo.Binary,doc='Pre-activation of each tech')

        m.Eff = pyo.Param(m.I,within=pyo.NonNegativeReals, doc='Technology efficiency')
        m.Eff_ba_ch = pyo.Param(within=pyo.NonNegativeReals,doc='Battery charging
         ↪  efficiency')
        m.Eff_ba_dch = pyo.Param(initialize = 1, doc='Battery discharge efficiency')
        m.Beta_ba = pyo.Param(within=pyo.NonNegativeReals,doc='Charging/discharging rate')
        m.Beta_hs = pyo.Param(within=pyo.NonNegativeReals, doc='identical charging rate for
         ↪  heat storage')

        m.L = pyo.Param(m.I,  within=pyo.NonNegativeIntegers, doc='Lifetime of technology
         ↪  i')
        m.X_min = pyo.Param(m.I,  within=pyo.NonNegativeReals, doc='Max possible installed
         ↪  capacity of technology ')
        m.X_max = pyo.Param(m.I, within=pyo.NonNegativeReals, doc='Min possible installed
         ↪  capacity of technology ')

        m.Temp = pyo.Param(m.T, within=pyo.Reals, doc='Ambient temperature of certain hour')
        m.Y_pv = pyo.Param(m.T, within=pyo.NonNegativeReals,doc='Possible PV output at time
         ↪  t')
        m.COP_a2a = pyo.Param(m.T, within=pyo.NonNegativeReals, doc='Air-to-air heat pump
         ↪  performance at time t w/r SH')


        #BITES-params
        m.SS_cap = pyo.Param(within=pyo.NonNegativeReals, doc='Shallow storage capacity of
         ↪  house, fully determined by area, thus a parameter instead of a variable')
        m.DS_cap = pyo.Param(within=pyo.NonNegativeReals, doc='Deep storage capacity of
         ↪  house, ---=---')
        m.K_shallow = pyo.Param(doc='loss factor for BITES, shallow part')
        m.K_deep = pyo.Param(doc='loss factor for BITES, deep part')
        m.K_flow = pyo.Param(doc='flow factor for BITES, cross-node flow')

    #---Energy Demand
```

```
        m.D_el = pyo.Param(m.T,  within=pyo.NonNegativeReals, doc='Hourly building
         ↪  electricity demand')
        m.D_sh = pyo.Param(m.T, within=pyo.NonNegativeReals, doc='Hourly building space
         ↪  heating demand')
        m.D_dhw = pyo.Param(m.T, within=pyo.NonNegativeReals, doc='Hourly building domestic
         ↪  hot water demand')

    #---Grid
        m.P_spot = pyo.Param(m.T,  within=pyo.NonNegativeReals, doc='Hourly price of
         ↪  imported electricity EUR/kWh including certificates')
        m.X_max_imp = pyo.Param(within=pyo.NonNegativeReals, doc='Maximum grid import')
        m.X_max_exp = pyo.Param( within=pyo.NonNegativeReals,doc='Maximum grid export')
        m.A_imp = pyo.Param( within=pyo.Binary, doc='Binary: Import is activated, 1/0')
        m.A_exp = pyo.Param( within=pyo.Binary,doc='Binary: Export is activated, 1/0')
        m.C_bf = pyo.Param(within=pyo.NonNegativeReals, doc='Constant price of biofuel')
        m.C_wo = pyo.Param(within=pyo.NonNegativeReals, doc='Constant price of wood')

    #---Control
        m.gamma = pyo.Param( within=pyo.NonNegativeReals, doc='=0 for strictly ZEB')
        m.R = pyo.Param(within=pyo.NonNegativeReals, doc='Chosen discount Rate')
        m.YRN = pyo.Param( within=pyo.NonNegativeIntegers, doc='Total years in modelling
         ↪  period')

        def npv_inv_spe(m, i):
            return self.cost(m.YRN, m.C_spe_0[i], m.L[i], m.R)
        m.C_spe = pyo.Param(m.I, rule = npv_inv_spe)

        def npv_inv_fxd(m, i):
            return self.cost(m.YRN, m.C_fxd_0[i], m.L[i], m.R)
        m.C_fxd = pyo.Param(m.I, rule = npv_inv_fxd)

    #VARIABLES ##########################################################

        # 1 STAGE : STRATEGIC VARIABLES
        m.x = pyo.Var(m.I,within = pyo.NonNegativeReals,
         ↪  doc='Optimal installed capacity (storage size), semi-continous, kW (kWh)')
        m.a_i = pyo.Var(m.I, within = pyo.Binary,
         ↪  doc='Activation binary decision for technology i, 1/0')


        #2 STAGE : OPERATIONAL VARIABLES
        m.q_hs = pyo.Var(m.T, domain = pyo.Reals,
         ↪  doc='Keeping track of HS discharge')
        m.q_eb_dhw = pyo.Var(m.T,domain = pyo.NonNegativeReals,
         ↪  doc='Net heat supplied from (DHW) electric boiler at time t, kWh/h')
        m.q_po_sh = pyo.Var(m.T,domain = pyo.NonNegativeReals,
         ↪  doc='Net heat supplied from (SH) paneloven at time t, kWh/h')
        m.q_a2a_sh = pyo.Var(m.T,  domain= pyo.NonNegativeReals,
         ↪  doc='Net heat supplied from air-to-air heat pump at time t, kWh/h')
        m.q_fp_sh = pyo.Var(m.T,  domain=pyo.NonNegativeReals,
         ↪  doc='Net heat supplied from bio boiler at time t, kWh/h')
        m.f_fp = pyo.Var(m.T,  domain= pyo.NonNegativeReals,
         ↪  doc='Wood input to the fireplace at time t kWh/h')

        m.z_hs = pyo.Var(m.T,  domain = pyo.NonNegativeReals,
         ↪  doc='Content in heat storage at the end of time t, kWh')
        m.z_ss = pyo.Var(m.T,  domain = pyo.NonNegativeReals, doc='Content in shallow BITES at the end of time t, kWh')
        m.z_ds = pyo.Var(m.T,  domain = pyo.NonNegativeReals, doc='Content in deep BITES at the end of time t, kWh')
        m.z_ba = pyo.Var(m.T,  domain = pyo.NonNegativeReals, doc='Content of battery at the end of time t, kWh')

        #charging variables for BITES, shallow part
        #m.z_ss_ch = pyo.Var(m.T,  domain = pyo.NonNegativeReals, doc='Maximum charging rate, shallow BITES at the end of time t, kWh')
        #m.z_ss_dch = pyo.Var(m.T,  domain = pyo.NonNegativeReals, doc='Maximum discharging rate, shallow BITES at the end of time t, kWh')
        m.q_ss = pyo.Var(m.T,  domain = pyo.Reals,                      doc='Energy released from shallow BITES at the end of time t, kWh')
        m.Flow = pyo.Var(m.T,  domain = pyo.Reals, doc='Cross-node flow of two-node BITES model at time t, kWh')


        m.y_imp = pyo.Var(m.T, domain = pyo.NonNegativeReals, doc='Electricity imported from grid at time t, kWh')
        m.y_exp = pyo.Var(m.T,  domain = pyo.NonNegativeReals, doc='Electricity exported to grid at time t, kWh')
        m.y_pv = pyo.Var(m.T, domain = pyo.NonNegativeReals,                      doc='PV production at time t, kWh/h')
        m.y_eb_dhw = pyo.Var(m.T, domain = pyo.NonNegativeReals, doc='Electricity drawn from (DHW) electric boiler at time t, kWh/h')
        m.y_po_sh = pyo.Var(m.T, domain = pyo.NonNegativeReals, doc='Electricity drawn for (SH) paneloven at time t, kWh/h')
        m.y_a2a_sh = pyo.Var(m.T,  domain=pyo.NonNegativeReals, doc='Electricity drawn for air-to-air heat pump at time t, kWh/h')
        m.y_ch = pyo.Var(m.T, domain = pyo.NonNegativeReals, doc='Amount of electricity to battery (charging) at time t, kWh/h')
        m.y_dch = pyo.Var(m.T, domain = pyo.NonNegativeReals, doc='Amount of electricity discharge from battery at time t, kWh/h')
        m.y_pty = pyo.Var(m.T, domain = pyo.Reals,                                  doc = 'Penalty volume')


        m.y_max = pyo.Var(m.M, domain = pyo.NonNegativeReals,                      doc='max power for every month')

        m.a_ch = pyo.Var(m.T,  domain = pyo.Binary,                                 doc ='Charging activation inward time t, 1 = activated')
        m.a_dch = pyo.Var(m.T,  domain= pyo.Binary,                                 doc ='Discharging activation inward time t, 1=activated')


    # CONSTRAINTS ##########################################################

    # 1 STAGE : INVESTMENTS

        #---Activation and boundary constraints
        def Tech_active(m, i, st):
            return m.x[i] <= m.a_i[i]*self.M_const
        m.Tech_active = pyo.Constraint(m.I, m.ST, rule = Tech_active)

```

```python
        def Tech_Min(m, i, st):
            return m.X_min[i]*m.a_i[i] <= m.x[i]
        m.Tech_Min = pyo.Constraint(m.I, m.ST, rule= Tech_Min)

        def Tech_Max(m, i, st):
            return m.x[i] <= m.X_max[i]*m.A_i[i]
        m.Tech_Max = pyo.Constraint(m.I, m.ST, rule= Tech_Max)

    #2 STAGE : OPERATIONS

        #---Balacing constraints

        def El_Balance(m, t, st):
            return m.D_el[t] == m.y_imp[t] + m.y_pv[t] - m.y_exp[t] + m.y_dch[t] - m.y_ch[t]
             ↪  - m.y_a2a_sh[t] - m.y_po_sh[t] - m.y_eb_dhw[t]
        m.El_Balance = pyo.Constraint(m.T, m.ST, rule = El_Balance)


        def SH_balance(m, t, st):
            return m.D_sh[t] == m.q_ss[t] + m.q_po_sh[t] + m.q_a2a_sh[t] + m.q_fp_sh[t]
        m.SH_balance = pyo.Constraint(m.T, m.ST, rule = SH_balance)

        def DHW_balance(m, t, st):
            if t == 0:
                return m.D_dhw[t] + m.z_hs[t] == m.z_hs[m.lastT]*m.Eff['HS'] + m.q_eb_dhw[t]
            else:
                return m.D_dhw[t] + m.z_hs[t] == m.z_hs[t-1]*m.Eff['HS'] + m.q_eb_dhw[t]
        m.DHW_balance = pyo.Constraint(m.T, m.ST, rule = DHW_balance)

        #---Capacity
        #split heat pumps in two here as well
        #point heat, no heat pumps
        def PO_SH_Restriction(m,t, st):
            return m.q_po_sh[t] <= m.x['PO_SH']
        m.PO_SH_Restriction = pyo.Constraint(m.T, m.ST, rule = PO_SH_Restriction)

        def FP_SH_Restriction(m,t, st):
            i = t - floor(t/24)*24
            if i >= 16 and i <= 24:
                return m.q_fp_sh[t] <= m.x['FP_SH']
            else:
                return m.q_fp_sh[t] == 0
        m.FP_SH_Restriction = pyo.Constraint(m.T, m.ST, rule = FP_SH_Restriction)

        def A2A_SH_Restriction(m,t, st):
            return m.q_a2a_sh[t] <= m.x['A2A_SH']
        m.A2A_SH_Restriction = pyo.Constraint(m.T, m.ST, rule = A2A_SH_Restriction)

        def A2A_SH_Restriction_1(m,t, st):
            return m.q_a2a_sh[t] <= 0.4*m.D_sh[t]
        m.A2A_SH_Restriction_1 = pyo.Constraint(m.T, m.ST, rule = A2A_SH_Restriction_1)

        def EB_DHW_Restriction(m,t, st):
            return m.q_eb_dhw[t] <= m.x['EB_DHW']
        m.EB_DHW_Restriction = pyo.Constraint(m.T, m.ST, rule = EB_DHW_Restriction)

        #---Grid equations
        def Grid_Import(m,t, st):
            return m.y_imp[t] <= m.X_max_imp
        m.Grid_Import = pyo.Constraint(m.T, m.ST,  rule=Grid_Import)

        def Grid_Export(m,t, st):
            return m.y_exp[t] <= m.X_max_exp
        m.Grid_Export = pyo.Constraint(m.T, m.ST,  rule=Grid_Export)

        #---Storage equations

        def HS_Restriction(m, t, st):
            return m.z_hs[t] <= m.x['HS']
        m.HS_Restriction = pyo.Constraint(m.T, m.ST, rule=HS_Restriction)

        def SS_Restriction(m, t, st):
            return m.z_ss[t] <= m.x['SS']*m.A_i['SS']
        m.SS_Restriction = pyo.Constraint(m.T, m.ST, rule=SS_Restriction)

        def DS_Restriction(m, t, st):
            return m.z_ss[t] <= m.x['DS']*m.A_i['DS']
        m.DS_Restriction = pyo.Constraint(m.T, m.ST, rule=DS_Restriction)

        def SS_charge_active(m,t, st):
            return m.q_ss[t] <= m.z_ss[t]
        m.SS_charge_active = pyo.Constraint(m.T, m.ST, rule=SS_charge_active)

        def SS_discharge_active(m,t, st):
            return - m.z_ss[t] <= m.q_ss[t]
        m.SS_discharge_active = pyo.Constraint(m.T, m.ST, rule=SS_discharge_active)

        def HS_charge_active(m,t, st):
            return m.q_hs[t] <= m.z_hs[t]
        m.HS_charge_active = pyo.Constraint(m.T, m.ST, rule=HS_charge_active)

        def HS_discharge_active(m,t, st):
            return - m.z_hs[t] <= m.q_hs[t]
        m.HS_discharge_active = pyo.Constraint(m.T, m.ST, rule=HS_discharge_active)
        '''
        def SS_charge_rate_max(m, t, st):
            if m.Temp[t] >= 15:
                return m.q_ss[t] >= -m.x['SS']
            elif m.Temp[t] < 15 and m.Temp[t] > -15:
                return m.q_ss[t] <= m.x['SS'] * (1 - (15 - m.Temp[t])/30)
            else:
                return m.q_ss[t] >= 0
        m.SS_charge_rate_max = pyo.Constraint(m.T, m.ST, rule = SS_charge_rate_max)

        def SS_discharge_rate_min(m, t, st):
            if m.Temp[t] >= 15:
                return m.q_ss[t] <= 0
            elif m.Temp[t] < 15 and m.Temp[t] > -15:
                return m.q_ss[t] >= -(m.x['SS'] * ((15 - m.Temp[t])/30))
            else:
                return m.q_ss[t] <= m.x['SS']
        m.SS_discharge_rate_min = pyo.Constraint(m.T, m.ST, rule = SS_discharge_rate_min)
```

```
'''

def HS_Balance_ch(m,t,st):
    if t == 0:
        return m.q_hs[t] == m.z_hs[m.lastT] - m.z_hs[t]
    else:
        return m.q_hs[t] == m.z_hs[t-1] - m.z_hs[t]
m.HS_Balance_ch = pyo.Constraint(m.T, m.ST, rule = HS_Balance_ch)

'''
def SS_balance_ch(m, t, st):
    if t == 0:
        return m.q_ss[t] == m.z_ss[m.lastT] - m.z_ss[t]
    else:
        return m.q_ss[t] == m.z_ss[t-1] - m.z_ss[t]
m.SS_balance_ch = pyo.Constraint(m.T, m.ST, rule = SS_balance_ch)
'''
def Flow_Constraint(m,t,st):
    return m.Flow[t] == m.K_flow*((m.z_ss[t]/m.SS_cap) - (m.z_ds[t]/m.DS_cap))
m.Flow_Constraint = pyo.Constraint(m.T, m.ST, rule = Flow_Constraint)

def SS_balance_ch(m, t, st):
    if t == 0:
        return m.q_ss[t] == m.z_ss[m.lastT] - m.z_ss[t] - m.Flow[t] - m.z_ss[t]*(1 -
         ↪  m.K_shallow)
    else:
        return m.q_ss[t] == m.z_ss[t-1] - m.z_ss[t] - m.Flow[t] - m.z_ss[t]*(1 -
         ↪  m.K_shallow)
m.SS_balance_ch = pyo.Constraint(m.T, m.ST, rule = SS_balance_ch)

def DS_balance_ch(m, t, st):
    if t == 0:
        return m.z_ds[t] == m.z_ds[m.lastT] + m.Flow[t] - m.z_ds[t]*(1 - m.K_deep)
    else:
        return m.z_ds[t] == m.z_ds[t-1] + m.Flow[t] - m.z_ds[t]*(1 - m.K_deep)
m.DS_balance_ch = pyo.Constraint(m.T, m.ST, rule = DS_balance_ch)
'''
def SS_discharge_rate_min(m,t, st):
       return -m.x['SS'] <= m.q_ss[t]
m.SS_discharge_rate_min = pyo.Constraint(m.T, m.ST, rule=SS_discharge_rate_min)

def SS_discharge_rate_max(m,t, s):
    return m.q_ss[t] <= m.x['SS']
m.SS_discharge_rate_max = pyo.Constraint(m.T, m.ST, rule=SS_discharge_rate_max)
'''


def HS_discharge_rate_min(m,t, st):
       return -m.x['HS']*m.Beta_hs <= m.q_hs[t]
m.HS_discharge_rate_min = pyo.Constraint(m.T, m.ST, rule=HS_discharge_rate_min)

def HS_discharge_rate_max(m,t, s):
    return m.q_hs[t] <= m.x['HS']*m.Beta_hs
m.HS_discharge_rate_max = pyo.Constraint(m.T, m.ST, rule=HS_discharge_rate_max)

def BA_Restriction(m,t, st):
    return m.z_ba[t] <= m.x['BA']
```

```python
        m.BA_restriction = pyo.Constraint(m.T, m.ST, rule=BA_Restriction)

        def BA_Balance(m,t, st):
            if t == 0:
                return m.z_ba[t] == m.z_ba[m.lastT] - m.y_dch[t]*(1/m.Eff_ba_dch) + m.y_ch[t]*m.Eff_ba_ch
            else:
                return m.z_ba[t] == m.z_ba[t-1] - m.y_dch[t]*(1/m.Eff_ba_dch) + m.y_ch[t]*m.Eff_ba_ch
        m.BA_Balance = pyo.Constraint(m.T, m.ST, rule=BA_Balance)

        def BA_Charge_Balance(m, t, st):
            if t == 0:
                return m.y_ch[t] <= (m.x['BA'] - m.z_ba[m.lastT])*m.A_i['BA']*(1/m.Eff_ba_ch)
            else:
                return m.y_ch[t] <= (m.x['BA'] - m.z_ba[t-1])*m.A_i['BA']*(1/m.Eff_ba_ch)
        m.BA_Charge_Balance = pyo.Constraint(m.T, m.ST, rule=BA_Charge_Balance)

        def BA_Discharge_Balance(m,t, st):
            if t == 0:
                return m.y_dch[t] <= m.z_ba[m.lastT]*m.A_i['BA']*m.Eff_ba_dch
            else:
                return m.y_dch[t] <= m.z_ba[t-1]*m.A_i['BA']*m.Eff_ba_dch
        m.BA_Discharge_Balance = pyo.Constraint(m.T, m.ST, rule=BA_Discharge_Balance)

        def BA_charge_active(m,t, st):
            return m.y_ch[t] <= m.X_max_imp*m.a_ch[t]
        m.BA_charge_active = pyo.Constraint(m.T, m.ST, rule=BA_charge_active)

        def BA_discharge_active(m,t, st):
            return m.y_dch[t] <= m.X_max_imp*m.a_dch[t]
        m.BA_discharge_active = pyo.Constraint(m.T, m.ST, rule=BA_discharge_active)

        def Battery_Balance(m,t, st):
            return m.a_ch[t] + m.a_dch[t] <= 1
        m.Battery_Balance = pyo.Constraint(m.T, m.ST, rule=Battery_Balance)

        def BA_charge_rate(m,t, st):
            return m.y_ch[t] <= m.x['BA']*m.Beta_ba
        m.BA_charge_rate = pyo.Constraint(m.T, m.ST, rule=BA_charge_rate)

        def BA_discharge_rate(m,t, st):
            return m.y_dch[t] <= m.x['BA']*m.Beta_ba
        m.BA_discharge_rate = pyo.Constraint(m.T, m.ST, rule=BA_discharge_rate)

        #---Production constraints for generating technologies

        def PV_Balance(m,t, st):
            return m.y_pv[t] == m.x['PV']*m.Y_pv[t]
        m.PV_Balance = pyo.Constraint(m.T, m.ST, rule=PV_Balance)

        def A2A_SH_Balance(m,t, st):
            return m.q_a2a_sh[t] == m.y_a2a_sh[t]*m.COP_a2a[t]
        m.A2A_SH_Balance = pyo.Constraint(m.T, m.ST, rule = A2A_SH_Balance)

        def PO_SH_Balance(m,t, st):
            return m.q_po_sh[t] == m.y_po_sh[t]*m.Eff['PO_SH']
        m.PO_SH_Balance = pyo.Constraint(m.T, m.ST, rule = PO_SH_Balance)

        def FP_SH_Balance(m,t, st):
            return m.q_fp_sh[t] == m.f_fp[t]*m.Eff['FP_SH']*m.A_i['FP_SH']
        m.FP_SH_Balance = pyo.Constraint(m.T, m.ST, rule = FP_SH_Balance)

        def EB_DHW_Balance(m,t, st):
            return m.q_eb_dhw[t] == m.y_eb_dhw[t]*m.Eff['EB_DHW']
        m.EB_DHW_Balance = pyo.Constraint(m.T, m.ST, rule = EB_DHW_Balance)


        #---Zero emission/energy constraints
        def ZE_Balance(m):
            if m.A_co2==1:
                print('ACTIVE ZEB-carbon RESTRICTION')
                if m.lastT == 8735:
                    return sum(m.y_imp[t]*m.G_el_imp + m.f_fp[t]*m.G_wo for t in m.T) <= sum(m.y_exp[t]*m.G_el_exp for t in m.T)
                if m.lastT == 4367:
                    return 2*sum(m.y_imp[t]*m.G_el - m.y_exp[t]*m.G_el + m.bf[t]*m.G_bf for t in m.T) <= m.G_ref*m.gamma
                elif m.lastT == 727:
                    return 12*sum(m.y_imp[t]*m.G_el - m.y_exp[t]*m.G_el + m.bf[t]*m.G_bf for t in m.T) <= m.G_ref*m.gamma
                elif m.lastT == 671:
                    return sum(m.y_imp[t]*m.G_el_imp + m.f_fp[t]*m.G_wo for t in m.T) <= sum(m.y_exp[t]*m.G_el_exp for t in m.T)
                elif m.lastT == 287:
                    return (8736/288)*sum(m.y_imp[t]*m.G_el - m.y_exp[t]*m.G_el + m.bf[t]*m.G_bf for t in m.T) <= m.G_ref*m.gamma
                elif m.lastT == 23:
                    return (8736/24)*sum(m.y_imp[t]*m.G_el - m.y_exp[t]*m.G_el + m.bf[t]*m.G_bf for t in m.T) <= m.G_ref*m.gamma
            elif m.A_pe == 1:
                print('ACTIVE ZEB-pef RESTRICTION')
                if m.lastT == 8735:
                    return sum(m.y_imp[t]*m.PE_imp - m.y_exp[t]*m.PE_exp + m.bf[t]*m.PE_bf for t in m.T) <= m.PE_ref*m.gamma
                elif m.lastT == 671:
                    return 13*sum(m.y_imp[t]*m.PE_imp - m.y_exp[t]*m.PE_exp + m.bf[t]*m.PE_bf for t in m.T) <= m.PE_ref*m.gamma
            else:
                print('NO ZEB RESTRICTION')
                return pyo.Constraint.Skip
        m.ZE_Balance = pyo.Constraint(rule=ZE_Balance)

    #Subscription power pricing Constraint
        def pty_volume(m, t): #counting all power within one hour exceeding max limit
            if m.A_ps == 1:
                return m.y_imp[t] - m.Y_max <= m.y_pty[t]
            else:
                return m.y_pty[t] == 0
        m.pty_volume = pyo.Constraint(m.T, rule = pty_volume)

        def pty_volume2(m, t):
                return 0 <= m.y_pty[t]
        m.pty_volume2 = pyo.Constraint(m.T, rule = pty_volume2)

    #   OBJECTIVE FUNCTION  #################################################

        def cost_Investments_rule(m, st):
            expr = self.npv_cost_Investments(m, st)
            return expr
        m.cost_Investments = pyo.Expression(m.ST, rule = cost_Investments_rule)

        def cost_Operation_rule(m, st):
            expr = self.npv_cost_Operations(m, st)
            return expr
        m.cost_Operations=  pyo.Expression(m.ST, rule = cost_Operation_rule)


        def objective_TotalCost(m):
            expr = pyo.summation(m.cost_Investments) + pyo.summation(m.cost_Operations)
            return expr
        m.objective_TotalCost = pyo.Objective(rule = objective_TotalCost, sense =
         ↪  pyo.minimize)

        return m

    def createConcreteModeltwoStage(self, data):
            '''Function creating instance from input data'''
            #data = self.writeStochasticProblem(data)
            concretemodel = self.abstractmodel.create_instance(data = {'mymodel':data},
             ↪  namespace = 'mymodel')
            return concretemodel
```

The following code, Loading_Data.py, contains two functions for the point-source model, one loading the relevant data from an excel spreadsheet, and another writing the results to a spreadsheet.

```python
import xlrd
import pandas as pd
import time
from openpyxl import load_workbook
from selection import selection

def createModelData(path_in, year):
    #Creates all input data for reduced model.
    wb = xlrd.open_workbook(path_in)
    ws1 = wb.sheet_by_name('Input_Tech')
    ws2 = wb.sheet_by_name('Input_Parameters')

    di = {}
    #Sets:
    di['T'] = {None: list(range(0, int(ws2.cell(16, 2).value) + 1))}
    di['lastT'] = {None: (ws2.cell(16, 2).value)}
    di['ST'] = {None:[1, 2]}

    for row in (15, 16, 17, 18, 19, 20, 21, 22, 26, 27, 28, 29, 30, 31, 32, \
                33, 34, 35, 36, 37, 39, 40, 41, 42, 43, 45, 46, 47, 48, 52, 53, \
                54, 57, 59, 60, 61):
        if row in [15, 16, 21, 22, 26, 30, 33, 39]:
            try:
                di[ws2.cell(row,1).value] = {None: int(ws2.cell(row,2).value)}
            except ValueError:
                print(row)
        else:
            di[ws2.cell(row,1).value] = {None: ws2.cell(row,2).value}

    #         important parameters imported from here:
    I = []
    for row in range(9):
        I.append(ws1.cell(4+row,1).value)
    di['I'] = {None:I}

    for col in (2, 3, 4, 5, 6, 7, 8, 9):
        data = {}
        for row in range(9):
            if col in [2,6]:
                data[I[row]] = int(ws1.cell(4+row,col).value)
            else:
                data[I[row]] = ws1.cell(4+row,col).value
            di[ws1.cell(2, col).value] = data



    #elif (ws2.cell(16, 2).value) == 8735:
    df = pd.read_excel('Scenario_data.xlsx', str(year), usecols =  [3,4,5,6,7,8,22]) #Now
    ↪ with separated COPs for DHW and SH
    df = df.drop([0])
    df = df.drop(range(8737,8761))

    if ws2.cell(16, 2).value == 8735:
        new_df = pd.DataFrame(data=df.values, columns =
         ↪  ['D_el','D_sh','D_dhw','P_spot','Temp','Y_pv','COP_a2a'], index = range(8736))
        data = new_df.to_dict()

        di['D_el'] = data['D_el']
        di['D_sh'] = data['D_sh']
        di['D_dhw'] = data['D_dhw']
        di['P_spot'] = data['P_spot']
        di['Temp'] = data['Temp']
        di['Y_pv'] = data['Y_pv']
        di['COP_a2a'] = data['COP_a2a']

        return di

    elif ws2.cell(16, 2).value == 671:
        #select 4 weeks for each season surrounding the highest space heating load, week
         ↪  size is 168 hours
        select_dict = selection(str(year), 'Scenario_data.xlsx')
        df = df.drop(range(1,select_dict['winter_start']))
        df = df.drop(range(select_dict['winter_start'] + 168 ,select_dict['spring_start']))
        df = df.drop(range(select_dict['spring_start'] + 168 ,select_dict['summer_start']))
        df = df.drop(range(select_dict['summer_start'] + 168 ,select_dict['fall_start']))
        df = df.drop(range(select_dict['fall_start'] + 168 , 8737))

        new_df = pd.DataFrame(data=df.values, columns =
         ↪  ['D_el','D_sh','D_dhw','P_spot','Temp','Y_pv','COP_a2a'], index = range(672))
        data = new_df.to_dict()
        di['D_el'] = data['D_el']
        di['D_sh'] = data['D_sh']
        di['D_dhw'] = data['D_dhw']
        di['P_spot'] = data['P_spot']
        di['Temp'] = data['Temp']
        di['Y_pv'] = data['Y_pv']
        di['COP_a2a'] = data['COP_a2a']

        return di


def saveDeterministicResults(m, result_file):

    wb = load_workbook(result_file)
    ser = wb['Series']
    val = wb['Values']
    if m.lastT == 23:
        f = 8736/24
    elif m.lastT == 287:
        f = 8736/288
    elif m.lastT == 8735:
        f = 1
    elif m.lastT==671:
        f = 13

    for i, elem in enumerate(['D_el', 'y_imp', 'y_exp', 'y_pty', 'y_ch', 'y_dch', \
                              'z_ba', 'y_pv', 'y_a2a_sh', 'D_dhw', 'q_eb_dhw', 'z_hs',
                               ↪  'q_hs', 'D_sh',\
                              'q_a2a_sh','q_fp_sh',\
                                'q_po_sh', 'z_ss', 'z_ds', 'q_ss', 't_amb', 'P_spot',
                                 ↪  'y_po_sh', 'y_eb_dhw']):
        ser.cell(row = 1, column = i + 1, value = elem)
    for t in m.T:
        ser.cell(row= t+2, column=1, value=m.D_el[t])
        ser.cell(row= t+2, column=2, value=m.y_imp[t].value)
        ser.cell(row= t+2, column=3, value=m.y_exp[t].value)
        ser.cell(row= t+2, column=4, value=m.y_pty[t].value)
        ser.cell(row= t+2, column=5, value=m.y_ch[t].value)
        ser.cell(row= t+2, column=6, value=m.y_dch[t].value)
        ser.cell(row= t+2, column=7, value=m.z_ba[t].value)
        ser.cell(row= t+2, column=8, value=m.y_pv[t].value)
        ser.cell(row= t+2, column=9, value=m.y_a2a_sh[t].value)
        ser.cell(row= t+2, column=10, value=m.D_dhw[t])
        ser.cell(row= t+2, column=11, value=m.q_eb_dhw[t].value)
        ser.cell(row= t+2, column=12, value=m.z_hs[t].value)
        ser.cell(row= t+2, column=13, value=m.q_hs[t].value)
        ser.cell(row= t+2, column=14, value=m.D_sh[t])
        ser.cell(row= t+2, column=15, value=m.q_a2a_sh[t].value)
        ser.cell(row= t+2, column=16, value=m.q_fp_sh[t].value)
        ser.cell(row= t+2, column=17, value=m.q_po_sh[t].value)
        ser.cell(row= t+2, column=18, value=m.z_ss[t].value)
        ser.cell(row= t+2, column=19, value=m.z_ds[t].value)
        ser.cell(row= t+2, column=20, value=m.q_ss[t].value)
        ser.cell(row= t+2, column=21, value=m.Temp[t])
        ser.cell(row= t+2, column=22, value=m.P_spot[t])
        ser.cell(row= t+2, column=23, value=m.y_po_sh[t].value)
        ser.cell(row= t+2, column=24, value=m.y_eb_dhw[t].value)

    #print('test output:', m.q_ashp[0].value, m.q_ashp[110].value )
    #Invested capacities
    val['A1'] = 'Time'
    val['B1'] = time.ctime() #Time and date
    #val['C4'] = time.time() - start_time #Code run time
    val['A2'] = 'Scenario name'
    val['B2'] = 'NOR'
    val['A3'] = 'gamma'
    val['B3'] = m.gamma.value
    val['A4'] = 'A_co2'
    val['B4'] = m.A_co2.value
    val['A5'] = 'A_pe'
    val['B5'] = m.A_pe.value

    val['B6'] = m.x['PV'].value
    val['B7'] = m.x['BA'].value
    val['B8'] = m.x['EB_DHW'].value
    val['B9'] = m.x['A2A_SH'].value
    val['B10'] = m.x['FP_SH'].value
    val['B11'] = m.x['PO_SH'].value
    val['B12'] = m.x['HS'].value
    val['B13'] = m.x['SS'].value

    val['A6'] = 'pv'
    val['A7'] = 'ba'
    val['A8'] = 'eb_dhw'
    val['A9'] = 'a2a_sh'
    val['A10'] = 'fp_sh'
    val['A11'] = 'po_sh'
    val['A12'] = 'hs'
    val['A13'] = 'ss'

    y_imp_tot = 0
    y_exp_tot = 0
    wo_tot = 0
    for t in m.T:
        y_imp_tot += m.y_imp[t].value
        y_exp_tot += m.y_exp[t].value
        wo_tot += m.f_fp[t].value

    G_tot = m.G_bf*wo_tot + m.G_el_imp*y_imp_tot

    val['F1'] = 'y_imp_tot'
    val['F2'] = 'y_exp_tot'
    val['F3'] = 'wo_tot'
    val['F4'] = 'G_tot'

    val['G1'] = y_imp_tot
    val['G2'] = y_exp_tot
    val['G3'] = wo_tot
    val['G4'] = G_tot

    val['F8'] = 'BITES-params'
    val['F9'] = 'K_flow'
    val['F10'] = 'K_shallow'
    val['F11'] = 'K_deep'

    val['G9'] = m.K_flow.value
    val['G10'] = m.K_shallow.value
    val['G11'] = m.K_deep.value

    y_imp_max = 0
    for t in m.T:
        if m.y_imp[t].value > y_imp_max:
            y_imp_max = m.y_imp[t].value

    val['F13'] = 'peak load el'
    val['G13'] = y_imp_max

    val['C5'] = 'Tech inv cost'
    val['C6'] = (m.C_spe['PV']*m.x['PV'].value+ m.C_fxd['PV']*m.a_i['PV'].value)
    val['C7'] = (m.C_spe['BA']*m.x['BA'].value+ m.C_fxd['BA']*m.a_i['BA'].value)
    val['C8'] = (m.C_spe['EB_DHW']*m.x['EB_DHW'].value+
     ↪  m.C_fxd['EB_DHW']*m.a_i['EB_DHW'].value)
    val['C9'] = (m.C_spe['A2A_SH']*m.x['A2A_SH'].value+
     ↪  m.C_fxd['A2A_SH']*m.a_i['A2A_SH'].value)
    val['C10'] = (m.C_spe['FP_SH']*m.x['FP_SH'].value+
     ↪  m.C_fxd['FP_SH']*m.a_i['FP_SH'].value)
    val['C11'] = (m.C_spe['PO_SH']*m.x['PO_SH'].value+
     ↪  m.C_fxd['PO_SH']*m.a_i['PO_SH'].value)
    val['C12'] = (m.C_spe['HS']*m.x['HS'].value+ m.C_fxd['HS']*m.a_i['HS'].value)
    val['C13'] = 0

    val['D1'] = m.objective_TotalCost()
    val['D2'] = m.cost_Investments[1]()
    val['D3'] = m.cost_Operations[2]()

    val['C1'] = 'tot_cost'
    val['C2'] = 'inv_cost'
    val['C3'] = 'op_cost'

    wb.save(result_file)
    print('Deterministic result file successfully created')
```

The following code is script running the optimization for the point-source model, which loads the two previous code fragments as modules:

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 12 20:17:29 2019

@author: marius
"""


import MODEL_ph_SH as model
import Loading_Data_ph as ld
import pyomo.environ as pyo
#import pandas as pd
#import numpy as np
#import pickle
#import xlrd
#import time
#from pyomo.environ import SolverFactory

year = 2012
path_in = 'allData.xlsx'
path_out = 'results/results_' + str(year) + '.xlsx'

abstract = model.ZEBModel()
data = ld.createModelData(path_in, year)
data['SS_cap'] = {None: 12.5}
data['DS_cap'] = {None: 90}

instance = abstract.abstractmodel.create_instance(data={'mymodel':data},namespace='mymodel')
opt = pyo.SolverFactory('gurobi')
#opt.options['TimeLimit'] = 1800
results = opt.solve(instance, tee=True, #to stream the solver output
                                                keepfiles=True, #print the LP file for
                                                 ↪  examination
                            symbolic_solver_labels=False) # use human readable names
ld.saveDeterministicResults(instance, path_out)
print('Objective value', instance.objective_TotalCost())
```